On an MPI Rank/Node Layout Utility for Improving Performance of Communications Intensive Heterogeneous MPI Applications on SGI Altix ICE 8200 Systems

Bracy H. Elton
High Performance Technologies Group, Dynamics Research Corporation
AFRL/RCM, DRC HPTG
Wright-Patterson AFB, Ohio, USA
E-mail: belton@drc.com

***Abstract*—In heterogeneous MPI applications different MPI processes (ranks)a may have different responsibilities and characteristics in terms of communication and computation. MPI launch facilities generally offer little in how to lay out such MPI applications onto systems, particularly in conjunction with batch submissions and being increasingly cumbersome with increasing numbers of nodes. Within an application it is possible to orchestrate desired layout arrangements; however, it typically leaves the application less portable. We present an MPI rank/compute node layout utility, targeted for SGI Altix ICE 8200 systems, such as HPCMP DSRC and DHPI systems, that can be applied externally to an application or internally in such a way so as to maintain portability without affecting performance. The utility helps in assigning MPI ranks to nodes on a system. Performance might be enhanced, for example, by putting sending and receiving MPI ranks on the same node. Additionally, the tool can help in specifying varying ratios of MPI ranks to compute node for separate groups of MPI ranks. It also provides the capability to group MPI ranks in blocked or interleaved fashions across compute nodes. Complex layouts for large numbers of MPI ranks can be expressed in a concise syntax. When used internally, the utility can also be made to work on Cray XE6 systems.**

I. Introduction

When running MPI jobs on distributed memory systems, typically MPI processes are mapped to compute nodes by using the default behavior of the launch mechanism or by giving general instructions, e.g., how many MPI ranks (processes) per compute node. For compute intensive problems, usually this means using the same number of MPI ranks per compute node as there are cores on a node. In heterogeneous MPI applications, different MPI ranks (processes) may have different responsibilities and characteristics in terms of communication, computation, and I/O. Examples of such applications include signal processing and some coupled models (fluids/structures, fluids/structures/chemistry, ocean/weather). Signal processing applications may involve a series of processing pipeline stages, each stage with its own computational characteristics and each interface between stages with its own communications characteristics. Additionally, some stages may be paralleized differently than other stages, e.g., distributed memory vs. shared memory. In coupled models, each model may involve its own computational, communications, and implementation characteristics, and the interfaces between the models will have their own characteristics as well.

By using "multiple binary launch", where a single MPI launch command starts up potentially different executables on different groups of compute nodes, it is possible to have different ratios of MPI ranks to compute nodes. This may help in applications that involve coupling a pure MPI part of a program with one that uses both distributed memory (MPI) and shared memory (pThreads, OpenMP) parallelism. By having the same node listed in one group and in another group of MPI ranks, it is possible to have non-adjacently numbered ranks mapped to the same compute node. This may improve inter-process communication if two non-adjacently numbered MPI ranks would otherwise communicate over the inter-node communications network. And generally, by specifying the mapping between MPI processes and compute nodes, it may be possible to significantly improve performance and/or accommodate various resource constraints. However, orchestrating this for a large job on batch systems, such as those in the United States Department of Defense (DOD) High Performance Computing Modernization Program (HPCMP) at its DOD Supercomputing Resources Centers (DSRCs), may prove to be painstaking and difficult, for the nodes allocated to jobs are unknown until the batch job begins. Furthermore, if an application is complicated and it is not readily known what arrangement is best, searching for a performance enhancing MPI rank/compute node mappings can be daunting.

Ideally, we would like a job to start running and then have each MPI rank determine, in an effort to optimize communications, its job based on the layout of the nodes assigned to the job. However, given the complexities of systems and

This work performed under the United States Department of Defense (DoD) High Performance Computing Modernization Program User Productivity Enhancement and Technology Transfer activities through High Performance Technologies, Inc. under contract No. GS04T09DBC0017. The opinions expressed herein are those of the authors and do not necessarily reflect the views of the US Department of Defense or the employer of the author.

of programs this may not be feasible. What we can do is to provide a utility that allows users to specify in a concise syntax how they want their job organized across the nodes that are assigned to the job. By combining knowledge of the application and of the system, users can experiment with various layouts to determine what works best. Without changing the code itself, it is possible to rearrange the nodes to the desired mapping between nodes and MPI ranks. We introduce a utility that addresses such difficulties and enables users to layout their jobs on SGI ALtix ICE 8200 systems, such as the Army Research Laboratory (ARL) Harold and ERDC DSRC Diamond DSRC systems, with complex mappings in a concise syntax. The MPI rank/node layout utility introduced here attempts to address this problem by providing mechanisms to allow users to have increased control over how their jobs are laid out on a system. There are two modes of usage. One provides some capability, though not necessarily optimal, without changing the user's MPI application; this works only on SGI Altix ICE systems. The second mode requires some code modification; this works on SGI Altix ICE and Cray XE6 systems. (The code modification regards abstracting MPI ranks so that there are logical and physical ranks and tables to translate between them.)

II. MPI Rank/Node Layout Utility

Specific features of the MPI rank/node layout utility include the following:

- Sorting of the input node list (provided from the batch system) according to rack, IRU, and node within an IRU. Sorting can be ascending, descending, or random within each category. Sorting is optional.
- Calculating the number of required nodes for a particular layout.
- Listing and reporting details of the processing. This aides verifying desired behavior of the utility.
- Specifying numbered ranks or a number of ranks with specific rank numbers determined automatically as needed (in ascending order from 0).
- Listing multiple layout specifications for groups of ranks.
- Setting default behavior for all specifications.
- Specifying two-dimensional (2-D) blocked layouts. This indicates MPI ranks spread across in a 2-D blocked fashion across compute nodes. You can indicate the number of MPI ranks per node.
- Specifying two groups of MPI ranks interleaved layouts. Here, two groups of MPI ranks are dispersed across a collection of compute nodes, in such a way that MPI ranks from each group can be assigned to the same compute node. This can help facilitate inter-node communications for two different sets of MPI ranks. For example, MPI rank 1 and 100 could be on the same node. If these ranks communicate, then it would be on the same node and perhaps improved. You can also specify the MPI rank density (MPI ranks per compute node).
- Specifying the number of ranks per node for each layout specification. Some groups of MPI ranks may need more memory than others, in which case different groups may have different MPI rank densities (MPI ranks per compute node). Also, an MPI application that has groups of MPI ranks that also employ threading, e.g., via POSIX pThreads or OpenMP, may need to have different MPI rank densities for each group.
- Allowing for gaps in the consumption of nodes. That is, the utility allows for skipping nodes. This might be useful in spreading an application out on a system in a way so as to not consume all the inter-node communications bandwidth.

In lieu of space to cover the utility's syntax and ocomplete semantics, we provide an illustrative example. Consider four groups (1–4) of nodes, say, from an signal processing applications where the groups constitute stages of a processing pipeline, where communication goes from stage 1 to stage 2 to stage 3 to stage 4. Suppose group 1 requires a lot of memory and little computing but stage 2 requires little memory and a lot of computing. Suppose stage 3 requires a lot of computing and moderate memory while stage 4 employs multithreading via OpenMP. Suppose stage 1, 2, 3, and 4 comprise 8, 32, 256, and 1 MPI ranks, respectively. Then

```
#!/bin/csh
uniq < ${PBS_NODEFILE} > nodefile
set layout = \
  "'interleave2d(8,1,32,4):5,block2d(256,16,16,8,8):8,2:1'"
layout -num_nodes -consolidate -layout ${layout}
layout -nodefile nodefile -consolidate \
  -layout ${layout} > mynodelist
set cmd = "mpirun -f mynodelist my_app.exe
${cmd}
```

results in 42 nodes being used for 298 MPI ranks. The first 40 (8+32) MPI ranks will have 5 ranks each, one from stage 1 and 4 from stage 2, each (the first node will have MPI ranks 0, 8, 9, 10, 11). The next 256 MPI ranks will be spread across 256/8 = 32 nodes in a 2×2 array of 8×8 subblocks. Two nodes (one MPI rank each) comprise stage 4. Changing the "`:8`" in the "`block2d[...]:8`" to "4" results in the 2-D blocked array having four MPI ranks per node, resulting in employing 74 nodes for (still) 298 MPI ranks.

While the specific syntax for the utility remains unspecified here, the above example illustrates that it is easy to devise complex mappings of MPI ranks onto compute nodes. Its compact and simple syntax readily supports parametric studies over different layouts.

You can find more information on this utility by running the following command on the ARL DSRC Harold and ERDC DSRC Diamond systems:

```
$PET_HOME/cta/sip/layout-1.0/bin/layout -help
```

III. Summary

We have briefly introduced a new utility to help with experimenting with various mappings between MPI ranks (processes) and compute nodes on SGI Altix ICE systems, such as the ARL DSRC Harold, ERDC DSRC Diamond, and AFRL Desch DHPI systems. The utility helps balance resources across MPI ranks. Examples where this may be beneficial in improved MPI application throughput include pipelined signal processing applications and coupled models (climate/ocean, structures/fluid dynamics).