Topology aware Cartesian grid mapping with MPI

Christoph Niethammer
High-Performance Computing Center Stuttgart
Stuttgart, Germany
niethammer@hlrs.de

Rolf Rabenseifner
High-Performance Computing Center Stuttgart
Stuttgart, Germany
rabenseifner@hlrs.de

CCS CONCEPTS

• **Computing methodologies → Parallel programming languages**;

KEYWORDS

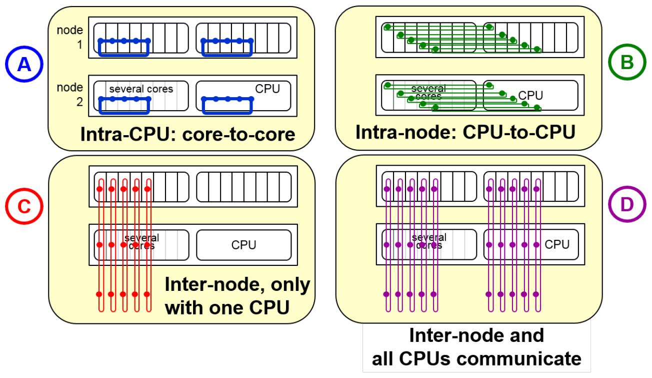
MPI, distributed programming, performance, grid, mapping

ACM Reference Format:
Christoph Niethammer and Rolf Rabenseifner. 2018. Topology aware Cartesian grid mapping with MPI. In *Proceedings of EuroMPI 2018*. ACM, New York, NY, USA, 3 pages.

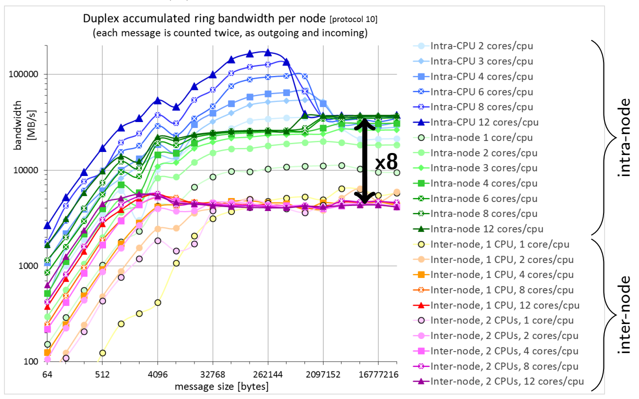
1 INTRODUCTION

Many scientific applications perform computations on a Cartesian grid. The common approach for the parallelization of these applications with MPI is domain decomposition. To help developers with the mapping of MPI processes to subdomains, the MPI standard provides the concept of process topologies. For Cartesian type topologies two useful functions are `MPI_Dims_create` and `MPI_Cart_create`. While the first function helps finding a factorization for a Cartesian process grid from a given number of processes, the second function creates an MPI Cartesian communicator from a given Cartesian process grid. However, this interface requires care in its usage as neither `MPI_Dims_create` takes into account the application topology nor `MPI_Cart_create` takes care of the underlying network topology and node architecture of the system. This becomes a problem for today's multi node NUMA systems because of the limited communication bandwidth at the different hardware levels, namely inter-node, inter-socket and inter-core.

Figure 1 shows different communication schemes (a) and the corresponding duplex accumulated ring bandwidth per node (b) for varying number of processes per node and process placements. Obviously, the limit of accumulated intra-CPU and intra-node bandwidth (green and blue) is 8x larger than the limit of accumulated node-to-node bandwidth (red and purple). To achieve good performance it is therefore essential to make a better (or the best) usage of the available bandwidth at the different levels: The inter-node communication must be reduced in favor of the intra-node communication.



(a) Communication schemes



(b) Duplex accumulated bandwidth per node

Figure 1: Duplex accumulated bandwidth benchmark: Messages are send bidirectionally in rings.

At this point, there are two problems with the MPI API: First, `MPI_Dims_create` computes only a grid with dimensions as close together as possible, e.g., based on the algorithm in [4]. If the underlying grid of a simulation is not close to a quadratic shape, the decomposition becomes non-optimal[1]. Figure 2 shows an example of a 1800×580 grid, which shall be distributed on 12 processors. `MPI_Dims_create` will suggest a 4×3 decomposition, however, a 6×2 decomposition would be optimal, as it leads to close to quadratic subdomains.

The second problem arises from not optimized implementations of `MPI_Cart_create`, which map processes linearly to the processor grid as shown in the left of Figure 3. This leads to unnecessary high numbers of slow inter-node communication. For comparison, an optimized mapping, which takes into account the network topology and system architecture,

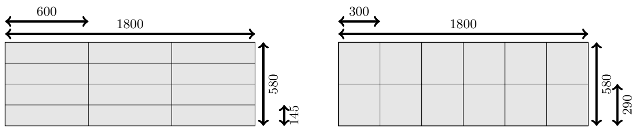


Figure 2: Decomposition of a grid with 1800×590 grid points onto 12 processes. Left: Suggestion from `MPI_Dims_create`, Right: Optimal distribution.

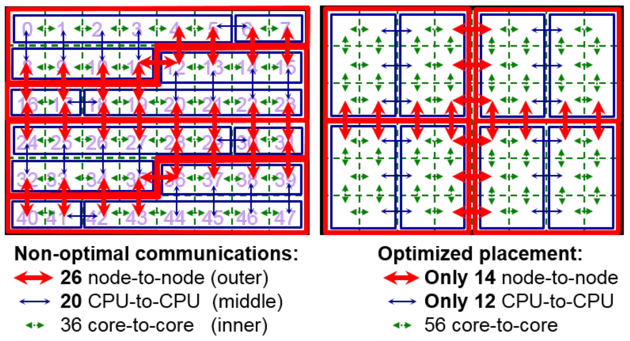


Figure 3: Process mapping to nodes, sockets and cores. Left: Non-optimal default mapping behaviour of `MPI_Cart_create`, Right: Optimized placement suggested by the multi level decomposition in this work.

is shown in the right. Generally, this mapping problem is non-trivial as shown in [3].

Both problems are closely coupled and therefore have to be solved together.

2 MAPPING STRATEGY

Hereinafter, the application topology is given as a d-dimensional Cartesian grid with a total of $T = \prod_{i=1}^{d} t_i$ elements, where t_i are the grid dimensions in directions $i \in [1, d]$. The target system shall consist of nodes with P cores per node.

To achieve an optimal domain decomposition and MPI process mapping for a given hardware topology, we present in the following a new multi level optimization approach. The benchmark in Figure 1 shows that the node-to-node communication is dominant for the whole communication overhead. Therefore, this communication must be minimized first, i.e., the approach starts with the node level and ends with the core level.

2.1 Grid decomposition at the node level

The application shall be run on N nodes. For the domain decomposition nodes, it will form a d-dimensional Cartesian node topology with n_i nodes in the i-th dimension. It is

$$N = \prod_{i=1}^{d} n_i \ . \tag{1}$$

The communication of the application requires for each node the exchange of halo data with it's neighbours. The amount of data to be transferred depends on the subdomain surface determined by it's dimensions, so that the communication costs c can be described as

$$c = 2 \sum_{i=1}^{d} \prod_{\substack{j=1 \\ j \neq i}}^{d} \frac{t_j}{n_j} = 2 \frac{T}{N} \sum_{i=1}^{d} \frac{n_i}{t_i} \ . \tag{2}$$

The goal is now to reduce the inter node communication costs. This is achieved by finding a set (n_i), which minimizes the sum in (2) under the condition (1).

2.2 Grid decomposition at the core level

After achieving an optimized node mapping, the same approach is applied at the next hardware topology level for the subdomains. The subgrid at the new level has the grid dimensions $t'_i = t_i / n_i$. Taking the core level as the next level, we construct a Cartesian core topology with dimensions p_i. The communication costs c' at this level are now given by

$$c' = 2 \frac{T'}{P} \sum_{i=1}^{d} \frac{p_i}{t'_i} = 2 \frac{T}{NP} \sum_{i=1}^{d} \frac{n_i p_i}{t_i} \quad \text{with} \quad P = \prod_{i=1}^{d} p_i \ . \tag{3}$$

Again, we search a set (p_i) that minimizes the term in (3).

2.3 Multi level decomposition

The approach shown so far for the node and core level can be applied hierarchically to any number of hardware topology levels by subsequentially minimizing the communication costs

$$c^{(l)} = 2 \frac{T}{\prod_{k=1}^{l} N^{(k)}} \sum_{i=1}^{d} \frac{\prod_{k=1}^{l} n_i^{(k)}}{t_i} \ \text{with} \ N^{(l)} = \prod_{i=1}^{d} n_i^{(l)} \ , \tag{4}$$

starting from node level $l = 1$. A result is shown in Figure 3.

Based on this decomposition then a rank reordering can be preformed to create a new optimized MPI communicator.

A first implementation[1] showed execution times in the order of Jesper Träff's algorithm [4]. The implementation of the rank mapping and the creation of the Cartesian communicator can be implemented according to [2]

3 SUMMARY

In this work, we present a new approach to optimize the domain decomposition in MPI applications, which perform computations on a Cartesian grid. Our approach takes into account the computational grid as well as the hardware topology to optimize inter-process communication. The optimization criteria is the minimization of slow bandwidth communication and is applied hierarchically to the different hardware layers. Based on our generalized approach, we intend to propose the addition of a new API to the MPI standard, which helps to create communicators for such scenarios.

[1]Our poster and a free implementation in a form suitable for MPI libraries is available at fs.hlrs.de/projects/par/mpi/EuroMPI2018-Cartesian

REFERENCES

[1] Pavan Balaji et al. 2009-2012. Topology awareness in MPI_Dims_create. https://github.com/mpi-forum/mpi-forum-historic/issues/195. Accessed 2018-07-19.

[2] Bill Gropp. 2018. Using Node Information to Implement MPI Cartesian Topologies. In *Proceedings of the 25th European MPI Users' Group Meeting (EuroMPI '18)*. ACM, New York, NY, USA.

[3] T. Hoefler and M. Snir. 2011. Generic Topology Mapping Strategies for Large-scale Parallel Architectures. In *Proceedings of the 2011 ACM International Conference on Supercomputing (ICS'11)*. ACM, 75–85.

[4] Jesper Larsson Träff and Felix Donatus Lübbe. 2015. Specification Guideline Violations by MPI_Dims_Create. In *Proceedings of the 22nd European MPI Users' Group Meeting (EuroMPI '15)*. ACM, New York, NY, USA, Article 19, 2 pages.