Toplogy aware
Cartesian grid mapping with MPI

Christoph Niethammer  Rolf Rabenseifner
niethammer@hlrs.de   rabenseifner@hlrs.de

High Performance Computing Center (HLRS), University of Stuttgart, Germany

HLRS, Stuttgart, Sep 25, 2018
Slides for the poster at EuroMPI 2018
For further information, see https://fs.hlrs.de/projects/par/mpi/EuroMPI2018-Cartesian/
The Problems of MPI_DIMS_CREATE + MPI_CART_CREATE

- The factorization of a given amount of MPI processes must be
  - Application topology aware [1]
  - Hardware topology aware
- Current definition of MPI_DIMS_CREATE is not prepared for this
- Extreme differences in latency and accumulated bandwidth between inter-node and intra-node communication
- The reordering by MPI_CART_CREATE:
  - Many implementations do nothing
  - A perfect reordering may require complex domain decomposition algorithms (e.g. Metis) [3]

We propose a new and fast algorithm, which is application and hardware topology aware

[1, 3] see References on last slide
Examples

- **Application topology awareness**
  - 2-D example with 12 MPI processes and gridsize 1800x580
    - `MPI_Dims_create` → 4x3
    - Grid aware → 6x2 processes
      - Boundary of a subdomain = 2(450+194) = 1288
      - Boundary of a subdomain = 2(300+290) = 1180

- **Hardware topology awareness**
  - 2-D example with 25 nodes x 24 cores and gridsize 3000x3000
    - `MPI_Dims_create` → 25 x 24
    - Hardware aware
      - (5 nodes x 6 cores) X (5 nodes x 4 cores)
      - Accumulated communication per node
        - O(10x120+12x125) = O(2700)
      - Accumulated communication per node
        - O(4x600) = O(2400)
Ring Benchmarks for Inter- and Intra-node Communication

Benchmark halo_irecv_send_multiplelinks_toggle.c
- Varying message size,
- number of *communication cores per CPU*, and
- four communication schemes (example with 5 *communicating cores per CPU*)

![Diagram of communication schemes](http://www.hlrs.de/training/par-prog ws/ Practical MPI.tar.gz subdirectory MPI/course/C/1sided/)

---

Topography aware MPI process grid mapping 2018
Niethammer, Rabenseifner
Duplex accumulated ring bandwidth per node

(each message is counted twice, as outgoing and incoming)

2 Haswell Intel Xeon E5-2680v3, each with 12 cores. Cray XC40 Aries dragonfly network

What is important?

The limit of accumulated intra-CPU and intra-node bandwidth is 8x larger than the limit of accumulated node-to-node bandwidth

Measurement with halo_irecv_send_multiplexlinks_toggle.c on 4 nodes of Cray XC40 hazelhen.hww.de, June 15, 2018, HLRS, by Rolf Rabenseifner (protocol 10)
Duplex accumulated ring bandwidth per node – scaling vs. asymptotic behavior

Core-to-core: Linear scaling for small to medium size messages due to caches

Node-to-node: One duplex link by one core already fully saturates the network

Core-to-core & CPU-to-CPU: Long messages: Same asymptotic limit through memory bandwidth

Result: The limit of accumulated intra-CPU and intra-node bandwidth is 8x larger than the limit of accumulated node-to-node bandwidth
Re-numbering on a cluster of SMPs (cores / CPUs / nodes)

- Example with 48 cores on:
  - 4 ccNUMA nodes
  - each node with 2 CPUs
  - each CPU with 6 cores

- 2-dim application with 6000 x 8080 gridpoints
  - Minimal communication with 2-dim domain composition
    with 1000 x 1010 gridpoints/core
    (shape as quadratic as possible → minimal circumference
    → minimal halo communication)
  - virtual 2-dim process grid: 6 x 8

- How to locate the MPI processes on the hardware?
  - Using sequential ranks in MPI_COMM_WORLD
  - Optimized placement
  → Proposed algorithm in slides 7-15

Non-optimal communications:
- 26 node-to-node (outer)
- 20 CPU-to-CPU (middle)
- 36 core-to-core (inner)

Optimized placement:
- Only 14 node-to-node
- Only 12 CPU-to-CPU
- 56 core-to-core

Order of the new ranks:
Last coordinate is running contiguously
→ Perfect basis for MPI_Cart_create() without reorder, i.e. with reorder==0 / .FALSE.
Proposed Mapping Algorithm

- To keep the algorithm small (and fast):
  - Use multi-level Cartesian subdomains
- Based on the benchmark results:
  - The first and major optimization goal is minimizing the inter-node communication volume
  - Using the algorithm from [4] for the multi-dimensional factorization of the number of nodes, but with a modified optimization goal that is application topology aware
- Using the same principles for each further hardware level to minimize
  - intra-node (i.e., CPU-to-CPU) communication
  - intra-CPU (i.e., core-to-core) communication
The Optimization Algorithm – First level

**Given:** $d$-dimensional Cartesian grid with a total of $T = \prod_{i=1}^{d} t_i$ elements

Level 1 (= outer level = node level) on $N$ nodes:

- Factorization of $N$ into factors $(n_i)_{i=1,d}$ with $N = \prod_{i=1}^{d} n_i$
- Communication cost in both directions of each dimension (example for $d = 3$):
  
  \[
  2 \frac{t_2t_3}{n_2n_3}, \quad 2 \frac{t_1t_3}{n_1n_3}, \quad 2 \frac{t_1t_2}{n_1n_2}
  \]
- Minimizing the communication costs $c$

\[
  c_{(level=1)} = 2 \sum_{i=1}^{d} \prod_{j=1 \atop j \neq i}^{d} \frac{t_j}{n_j} = 2 \frac{T}{N} \sum_{i=1}^{d} \frac{n_i}{t_i}
\]

**Summary of Level 1:** One must search factors $(n_i)_{i=1,d}$

- that factorize $N$ with $N = \prod_{i=1}^{d} n_i$
- and minimize the term $\sum_{i=1}^{d} \frac{n_i}{t_i}$
Second Level Optimization

Level 2:

- Each node has $P$ processors (or cores)
- Factorization of $P$ into factors $p_i$ with $P = \prod_{i=1}^{d} p_i$
- Communication costs in both directions of each dimension:
  
  $$2 \frac{t_2 t_3}{n_2 p_2 n_3 p_3}, \ 2 \frac{t_1 t_3}{n_1 p_1 n_3 p_3}, \ 2 \frac{t_1 t_2}{n_1 p_1 n_2 p_2}$$

- Minimizing the communication costs $c$,

  $$c^{(level=2)} = 2 \sum_{i=1}^{d} \prod_{j=1, j\neq i}^{d} \frac{t_j}{n_j p_j} = 2 \frac{T}{NP} \sum_{i=1}^{d} \frac{n_i p_i}{t_i}$$

Summary of Level 2:

One must search factors $(p_i)_{i=1,d}$

- that factorize $P$ with $P = \prod_{i=1}^{d} p_i$
- and minimize the term $\sum_{i=1}^{d} \frac{n_i p_i}{t_i}$
Topology aware MPI process grid mapping

Hierarchical Cartesian Domain Decomposition

Primary and main optimization goal:
Whole communication from each node to all of its neighbors must be minimized!

Second and minor optimization goal:
Whole intra-node communication must be minimized!
Example

- **Given:** $d=3$ dimensions, $N=625$ nodes, $P=24$ cores, and all $t_i$ are identical
- $\rightarrow$ in all formulas, the $1/t_i$ can be ignored
- **Level 1:**
  - Search $(n_i)_{i=1,3}$ that $n_1 n_2 n_3 = 625$ and $\sum_{i=1}^{3} n_i$ minimal
  - Result: $(n_i)_{i=1,3} = (25,5,5)$ with $\sum_{i=1}^{3} n_i = 35$
- **Level 2:**
  - Search $(p_i)_{i=1,3}$ that $p_1 p_2 p_3 = 24$ and $\sum_{i=1}^{3} n_i p_i$ minimal
  - Result: $(p_i)_{i=1,3} = (1,6,4)$ with $\sum_{i=1}^{3} n_i p_i = 25 + 30 + 20 = 75$
- **Optimized result:**
  - with $(nodes \times cores)$ in each dimension: $(25 \times 1) \times (5 \times 6) \times (5 \times 4)$
  - Total core numbers as used for MPI_Cart_create: $25 \times 30 \times 20$
- **Comparison with existing MPI_Dims_create:**
  - MPI_Dims_create would result in $25 \times 25 \times 24$
  $\rightarrow$ complex mapping to the hardware is needed, see also slide 2 or no reordering is done $\rightarrow (25 \times 1) \times (25 \times 1) \times (1 \times 24)$
  with $\sum_{i=1}^{3} n_i = 51 (\text{instead of } 35)$, $\rightarrow 46\%$ more inter-node communication!
Generalized multi-level optimization

**Given:** $d$-dimensional Cartesian grid with a total of $T = \prod_{i=1}^{d} t_i$ elements

Number of hardware levels $L$

and for each level the number of processors $N^{(l)}$, $l = 1, L$

Communication costs on each level $l$:

$$c^{(l)} = 2 \frac{T}{\prod_{k=1}^{l} N^{(k)}} \sum_{i=1}^{d} \frac{\prod_{k=1}^{l} n^{(k)}_i}{t_i}$$

for a factorization $N^{(l)} = \prod_{i=1}^{d} n^{(l)}_i$

Search:

- On each level $l = 1..L$, one must search a factorization $\left( n^{(l)}_i \right)_{i=1,d}$ of $N^{(l)}$
- with $N^{(l)} = \prod_{i=1}^{d} n^{(l)}_i$
- and minimal sum $\sum_{i=1}^{d} \frac{\prod_{k=1}^{l} n^{(k)}_i}{t_i}$,

  i.e., with minimal sum $\sum_{i=1}^{d} a^{(l)}_i n^{(l)}_i$ with $a^{(l)}_i = \frac{\prod_{k=1}^{l-1} n^{(k)}_i}{t_i}$
Generalization with weighted communication in each direction, e.g. by different halo width

- If the communication cost in each direction $i = 1, d$ is multiplied with a halo width $w_i$, e.g., $2 \frac{t_2 t_3}{n_2 n_3} w_1$
- On each $l = 1..L$, the sum to be minimized is
  $$\sum_{i=1}^{d} \frac{\prod_{k=1}^{l} n_i^{(k)}}{(t_i/w_i)}$$
- i.e., $\sum_{i=1}^{d} a_i^{(l)} n_i^{(l)}$ with $a_i^{(l)} = \frac{\prod_{k=1}^{l-1} n_i^{(k)}}{(t_i/w_i)}$
Using weighted MPI\_Dims\_create for application topology awareness

- **Task:** In general, on each hardware topology level \( l \) with \( l = 1, L \) and for given \( N^{(l)} \), find factorizations \( \binom{n^{(l)}_i}{i = 1..d} \) with \( N^{(l)} = \prod_{i=1}^{d} n^{(l)}_i \) and the following sum is minimal: \( \sum_{i=1}^{d} a^{(l)}_i n^{(l)}_i \) with \( a^{(l)}_i = \frac{\prod_{k=1}^{l-1} n^{(k)}_i}{(t_i/w_i)} \)

- **Algorithm** on each level \( l = 1 \ldots L \)
  - Sorting indexes \( i = 1 \ldots d \) so that the \( \binom{a_i'}{i' = 1..d} \) to be non-decreasing
  - Calculate all divisors of \( N^{(l)} \) i.e. with \( n_i \geq n_{i+1} \)
  - Loop over all non-increasing factorizations and find optimum according to
    - 1\(^{st}\) criterion: a factorization is better if \( \sum_{i=1}^{d} a^{(l)}_i n^{(l)}_i \) is smaller
    - 2\(^{nd}\) criterion: if the \( \sum_{i=1}^{d} a^{(l)}_i n^{(l)}_i \) is the same then a factorization is better if \( \Delta = n^{(l)}_1 - n^{(l)}_d \) is smaller
    - 3\(^{rd}\) criterion: if \( \sum_{i=1}^{d} n_i \) and \( \Delta = n^{(l)}_1 - n^{(l)}_d \) are the same then a factorization is better if \( n^{(l)}_1 \) is smaller
  - Revert the index mapping \( \binom{n^{(l)}_i}{i' = 1..d} \rightarrow \binom{n^{(l)}_i}{i = 1..d} \)

Re-mapping of all process ranks according to \( \binom{n^{(l)}_i}{i = 1..d, l = 1..L} \)

and creation of the new Cartesian communicator according to algorithm in [2]

Additional tricks:
- Calculate divisors only up to \( \sqrt{N} \), calculate the rest by reciprocal values.
- Loop over divisors from highest to smallest, recursively over \( i = 1, d \),
- Start value for \( n_{i+1} \) is next real divisor equal or smaller then \( n_i \)
Rank mapping is based on:
• Node level: 625 = 5 x 25 x 5
• CPU level: 2 = 2 x 1 x 1
• Core level: 12 = 3 x 1 x 4
Result (product): 1 30 x 25 x 20

The Cartesian communicator reflects this result: 30 x 25 x 20

Proposed Interfaces

MPI_Cart_ml_create_from_types (MPI_Comm comm_old, int ntype_levels, int type_levels[ntype_levels], int ndims, double dim_weights[ndims], int periods[ndims], MPI_Info info, /*OUT*/ int dims[ndims], MPI_Comm *comm_cart);

MPI_Cart_ml_create_from_comms (int nlevels, MPI_Comm level_comms[nlevels], int ndims, double dim_weights[ndims], int periods[ndims], MPI_Info info, /*OUT*/ int dims[ndims], MPI_Comm *comm_cart);

MPI_Dims_weighted_create (int nnodes, int ndims, double dim_weights[ndims], /*OUT*/ int dims[ndims]);

MPI_Dims_ml_create (int nnodes, int ndims, double dim_weights[ndims], int nlevels, int sizes[nlevels], /*OUT*/ int dims_ml[ndims][nlevels]);
Conclusion and Outlook

Conclusions

• We developed a new algorithm to minimize the total communication time in a cluster of ccNUMA nodes with multi-core CPUs.
• It is needed, due to the significant bandwidth differences between inter- and intra-node communication.
• It can be implemented based on the algorithm in [4], but with a modified optimization goal, and repeated for each hardware level.

Outlook

• We plan to provide a portable implementation, and
• compare it with existing solutions with MPI_Dims_create + MPI_Cart_create.
• We plan to propose an appropriate interface for the next MPI standard, because MPI libraries may internally have faster access to the hardware topology information for a given communicator.
References


Appendix

- Additional material that is not part of the poster
Renumbering in a 2-dim example with 3 levels (nodes / CPUs / cores)

\[ \text{dim1} = \text{inner}_d1 \times \text{mid}_d1 \times \text{outer}_d1 = 3 \times 2 \times 3 = 18 \]

Old ranks in MPI_COMM_WORLD. The ranks must be sequential in the hardware, i.e., first through the CPUs, then through the ccNUMA nodes, and then through the cluster.

New ranks in optimized communicator. → New global communicator with minimal node-to-node & optimal intra-node communication.

Order of the new ranks: last coordinate is running contiguously → Perfect basis for MPI_Cart_create() with reorder==0 / .FALSE.

Number of communication links:

Without re-numbering:
- 150 outer
- 72 mid
- 180 inner

With new ranks:
- 60 outer
- 90 mid
- 252 inner

2-D example on hierarchical hardware with 9 nodes x 4 CPUs x 6 cores resulting in 12*18 Cartesian processes.
Renumbering in a 2-dim example with 3 levels (nodes / CPUs / cores) – the code

Product = number of cores/CPU  number of CPUs/node  number of nodes

/*Input: */

\[
\begin{align*}
\text{inner}_d_0 &= \ldots; \\
\text{mid}_d_0 &= \ldots; \\
\text{outer}_d_0 &= \ldots; \\
\text{inner}_d_1 &= \ldots; \\
\text{mid}_d_1 &= \ldots; \\
\text{outer}_d_1 &= \ldots;
\end{align*}
\]

dim0=inner_d0*mid_d0*outer_d0;  \quad dim1=inner_d1*mid_d1*outer_d1;
idim=inner_d0*inner_d1;  \quad mdim=mid_d0*mid_d1;  \quad odim=outer_d0*outer_d1;
whole_size=dim0*dim1  /* or  =idim*mdim*odim */;
ranks = malloc(whole_size*sizeof(int));

for (oc0=0; oc0<outer_d0; oc0++) /*any sequence of the nested loops works*/
for (mc0=0; mc0<mid_d0; mc0++)
for (ic0=0; ic0<inner_d0; ic0++)
for (oc1=0; oc1<outer_d1; oc1++)
for (mc1=0; mc1<mid_d1; mc1++)
for (ic1=0; ic1<inner_d1; ic1++)
{
    \textbf{old\_rank} = ic1 + inner_d1*ic0 + (mc1 + mid_d1 *mc0)*idim \\
    + (oc1 + outer_d1*oc0)*idim*mdim;
    c0 = ic0 + inner_d0*mc0 + inner_d0*mid_d0*oc0;
    c1 = ic1 + inner_d1*mc1 + inner_d1*mid_d1*oc1;
    new\_rank = c1 + dim1*c0;
    ranks[new\_rank] = old\_rank;
}

/* Establishing new\_comm with the new ranking in an array "ranks": */

MPI_Comm_group(MPI_COMM_WORLD, &world_group);

MPI_Group_incl(world_group, world_size, ranks, &new_group); free(ranks);

MPI_Comm_create(MPI_COMM_WORLD, new_group, &new_comm);

dims[0] = dim0; dims[1] = dim1;

MPI_Cart_create(new_comm, 2, dims, periods, 0 /*=false*/, &comm_cart);

For an alternative with MPI_Comm_split, see MPI-3.1, Sec. 7.5.8, page 313, lines 7-13.
MPI_Dims_create optimizing $\sum_{i=1}^{d} n_i$

- Based on Jesper Träff’s algorithm *tuwdims.c* developed for [4]
  - Jesper’s optimization criterion:
    - Minimizing $\Delta = n_1 - n_d$
- New optimization criterion (three criteria)
  - 1\(^{st}\) criterion: a factorization is better if $\sum_{i=1}^{d} n_i$ is smaller
  - 2\(^{nd}\) criterion: if the $\sum_{i=1}^{d} n_i$ is the same then
    a factorization is better if $\Delta = n_1 - n_d$ is smaller
  - 3\(^{rd}\) criterion: if $\sum_{i=1}^{d} n_i$ and $\Delta = n_1 - n_d$ are the same then
    a factorization is better if $n_1$ is smaller
- Both algorithms have nearly same execution time: $\sim O(10\mu s/call)$
- For $n=2..10,000,000$ and ndims=2..10
  - 4630x different factorization: with same $\Sigma$ and $\Delta$ and smaller $\textit{dims}[0]$
    - e.g., ndims=3: N=360 = (old) 10x6x6 = (new) 9x8x5 (both $\Sigma=22$ and $\Delta=4$)
  - 6066x different factorization: with better $\Sigma$ and mostly worse $\Delta$
    - e.g., ndims=3: N=35200 = (old) 40x40x22 ($\Sigma=102$, $\Delta=18$) = (new) 44x32x25 ($\Sigma=101$, $\Delta=19$)
    - N=37044 = (old) 42x42x21 ($\Sigma=105$, $\Delta=21$) = (new) 49x28x27 ($\Sigma=104$, $\Delta=22$)

Next below 10,000 are:
- ndims=3:
  - 22x14x12 = 21x16x11
  - 21x16x15 = 20x18x14
  - 26x16x15 = 24x20x13
- ndims=4:
  - 10xFx6x6 = 9xFxFx5 with F=6,7,8,9

All other larger than 50,000