Topology aware Cartesian grid mapping with MPI

Christoph Niethammer, Rolf Rabenseifner
High Performance Computing Center Stuttgart, University of Stuttgart, Germany
{niethammer,rabenseifner}@hlrs.de

For further information, see https://fs.hlrs.de/projects/par/mpi/EuroMPI2018-Cartesian/
Poster at EuroMPI 2018, Barcelona, Spain, Sep 23-26, 2018

1 Adressed problem

The Problems of MPI_Dims_create + MPI_Cart_create

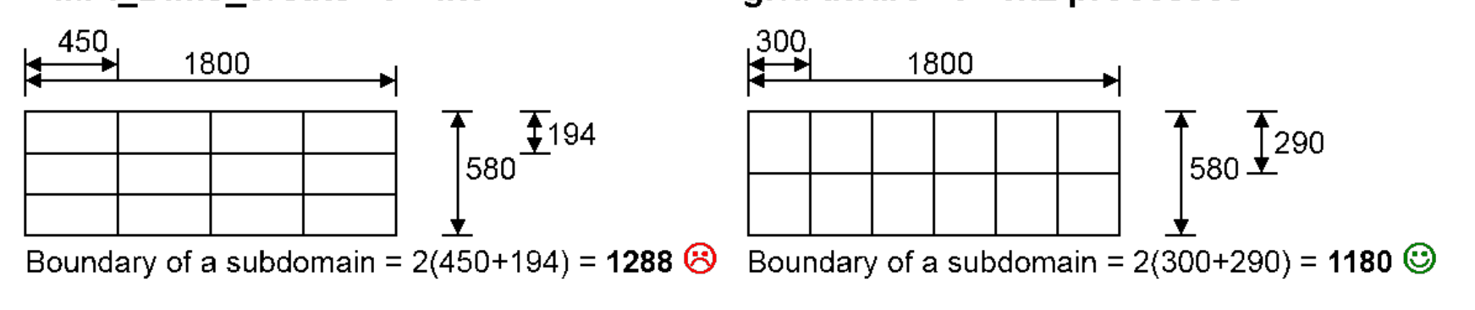
- The factorization of a given amount of MPI processes must be
  - Application topology aware [1]
  - Hardware topology aware
- Current definition of MPI_Dims_create is not prepared for this
  (Boxes 1+2)
- Extreme differences in latency and accumulated bandwidth between **inter**-node and **intra**-node communication (Boxes 3-5)
- The reordering by MPI_Cart_create:
  - Many implementations do nothing (Box 6)
  - A perfect reordering may require complex domain decomposition algorithms (e.g. Metis) [2]

We propose a new and fast algorithm, which is application and hardware topology aware (Boxes 7-15)

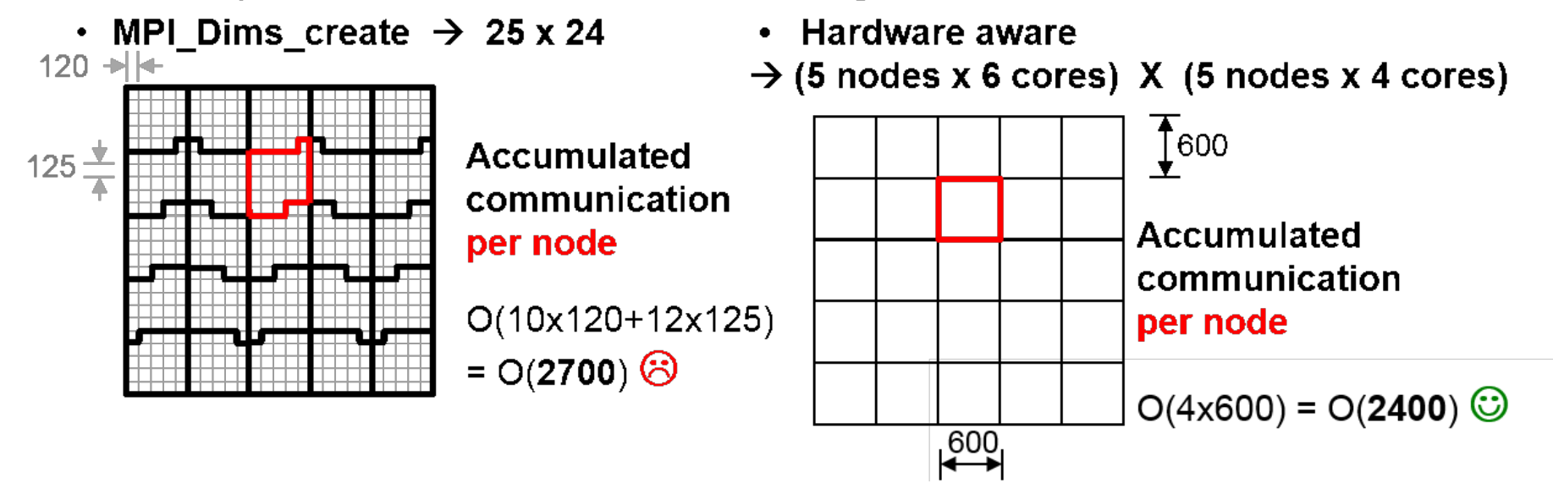
[1, 2] see References in last box (Box 17)

2 Examples

- Application topology awareness
  - 2-D example with 12 MPI processes and gridsize 1800x580
    - **MPI_Dims_create → 4x3**
    - **grid aware → 6x2 processes**


450
1800
194
580
Boundary of a subdomain = 2(450+194) = 1288 ☹
300
1800
290
580
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)**


120
125
Accumulated communication per node
O(10x120+12x125) = O(2700) ☹
600
Accumulated communication per node
O(4x600) = O(2400) ☺
600

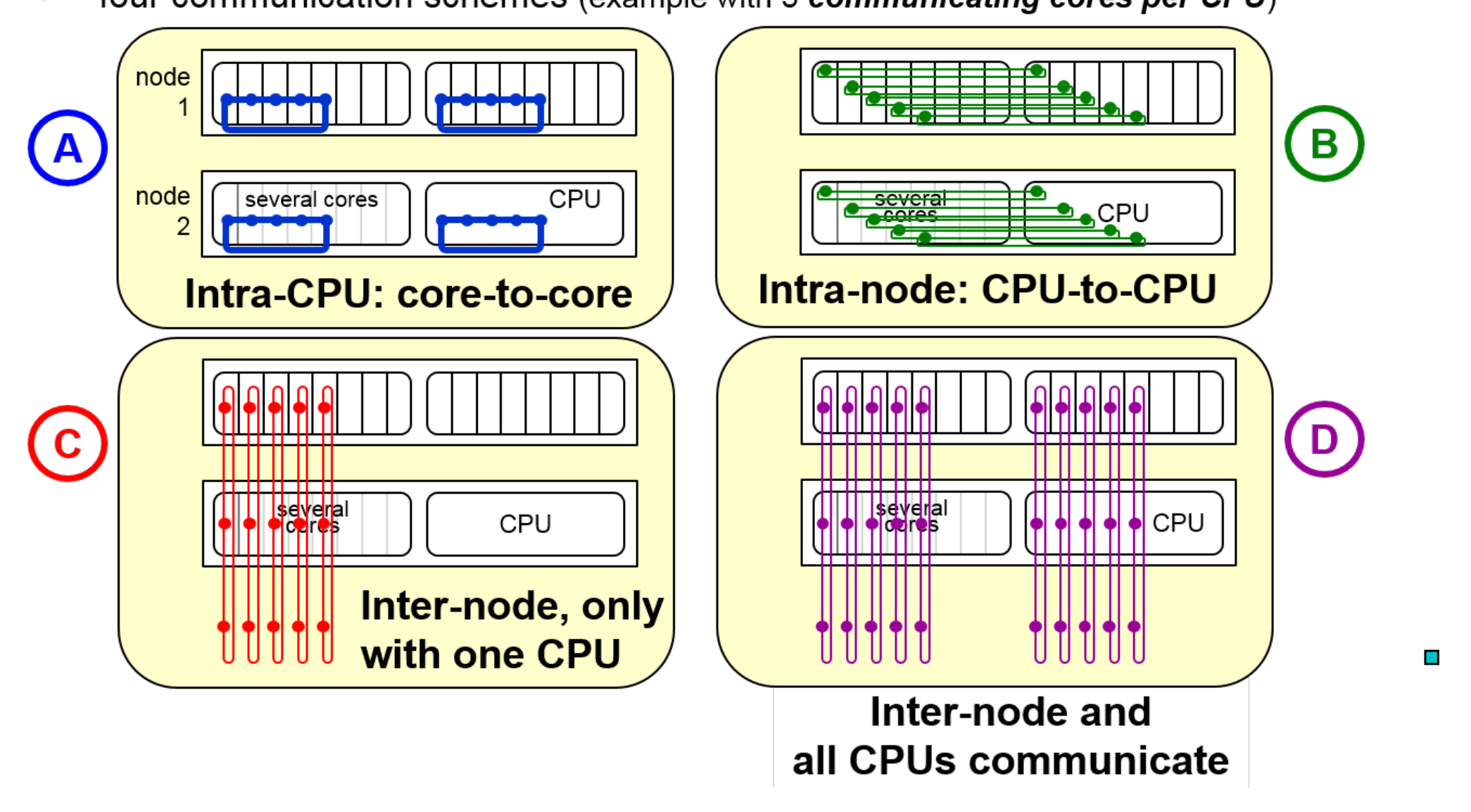
3 Ring Benchmark - Description

Ring Benchmarks for Inter- and Intra-node Communication

Benchmark halo_irecv_send_multiplelinks_toggle.c

See HLRS online courses http://www.hlrs.de/training/par-prog-ws/ → Practical → MPI.tar.gz → subdirectory MPI/course/C/1sided/

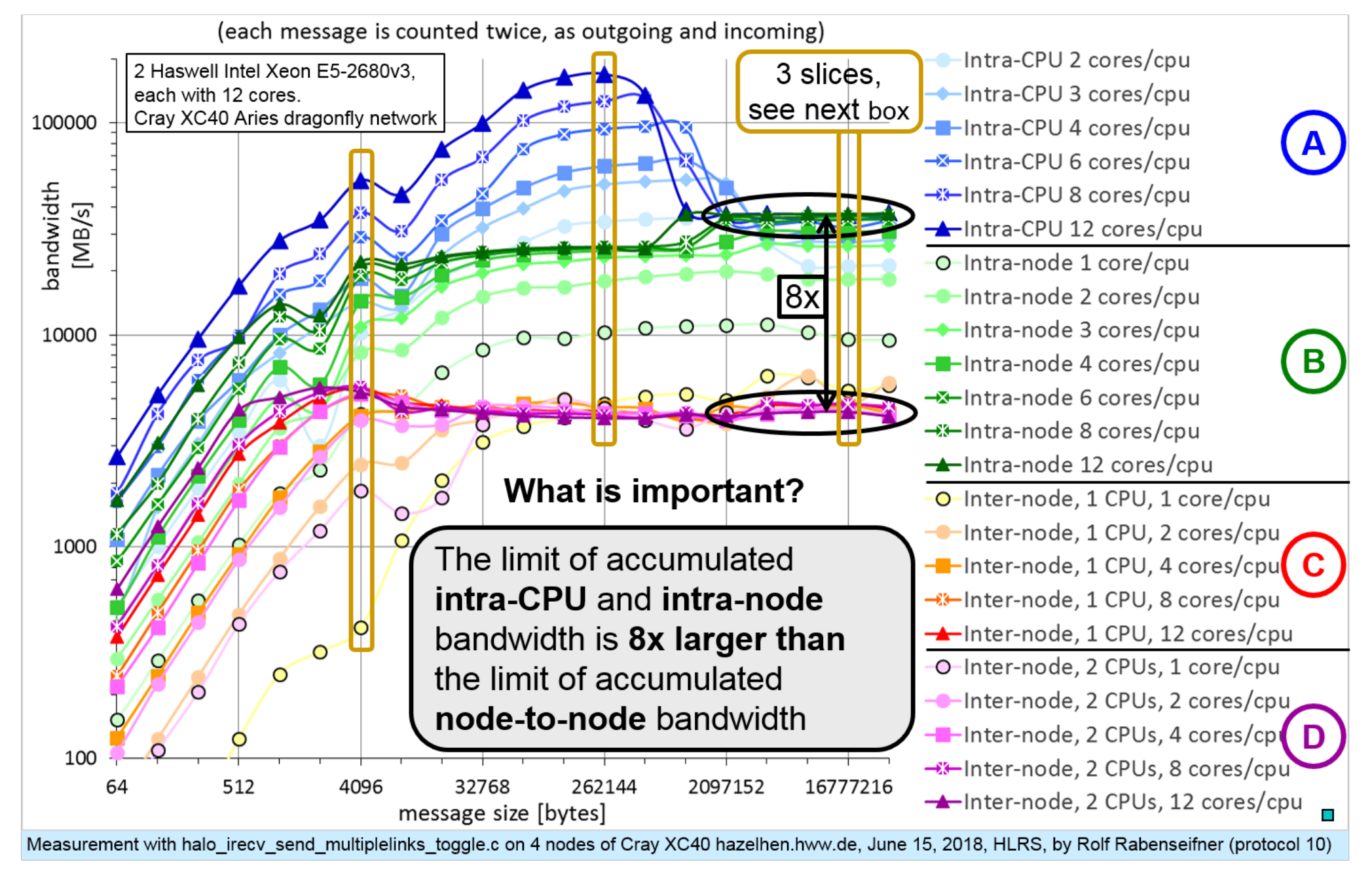
- Varying message size,
- number of ***communication cores per CPU***, and
- four communication schemes (example with 5 ***communicating cores per CPU***)



4 Ring Benchmark - Results I

See HLRS online courses http://www.hlrs.de/training/par-prog-ws/ → Practical → MPI.tar.gz → subdirectory MPI/course/C/1sided/

Duplex accumulated ring bandwidth per node



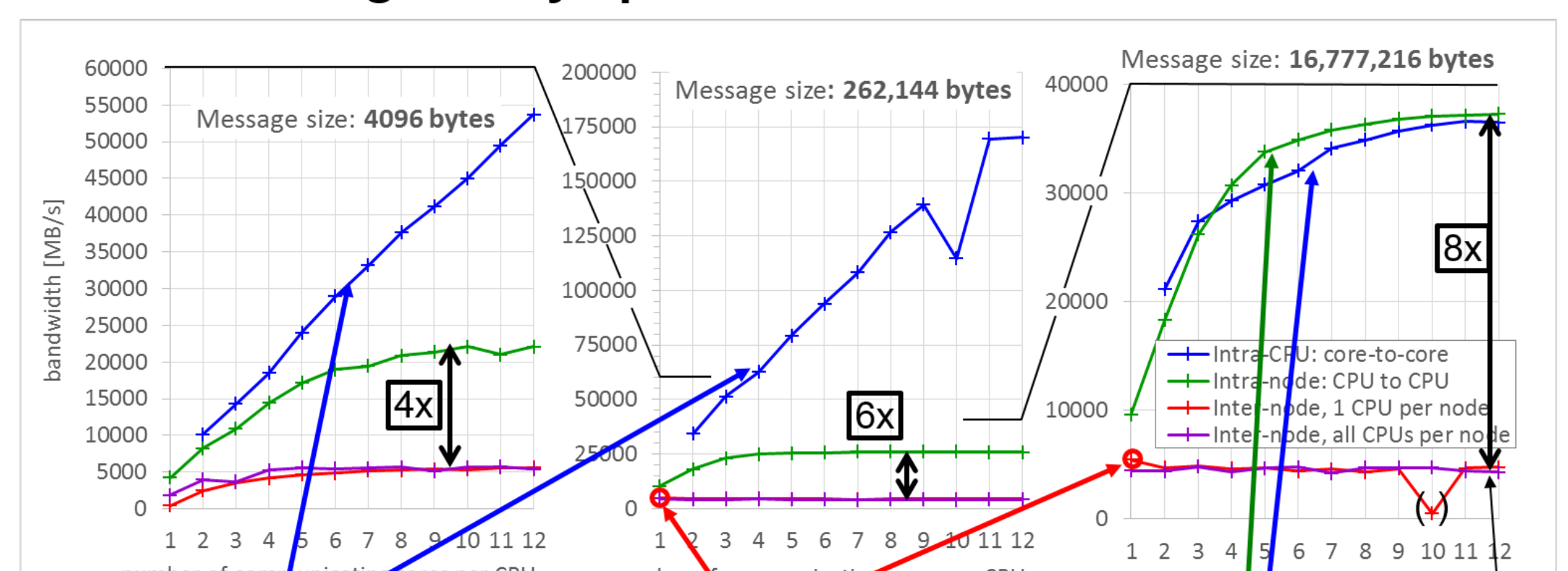
What is important?

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_multiplelinks_toggle.c on 4 nodes of Cray XC40 hazelhen.hww.de, June 15, 2018, HLRS, by Rolf Rabenseifner (protocol 10)

5 Ring Benchmark - Results II

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

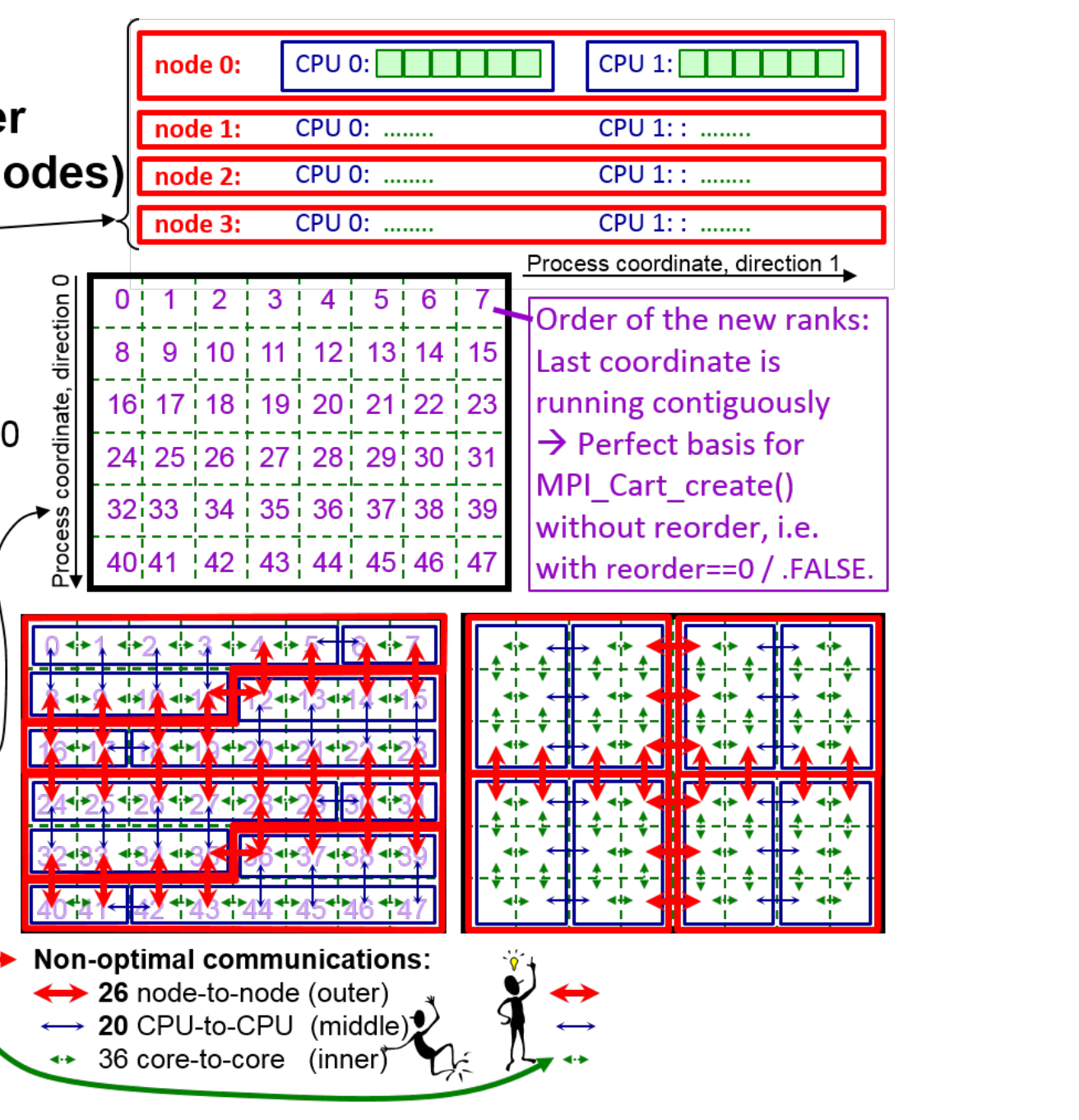
Core-to-core & CPU-to-CPU: **Long messages**: Same asymptotic limit through **memory bandwidth**

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

6 Rank Reordering Problem

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 Boxes 7-15**


node 0:
CPU 0:
CPU 1:
node 1:
node 2:
node 3:
Process coordinate, direction 1
Process coordinate, direction 0
Order of the new ranks: Last coordinate is running contiguously → Perfect basis for MPI_Cart_create() without reorder, i.e. with reorder==0 / .FALSE.
Non-optimal communications:
26 node-to-node (outer)
20 CPU-to-CPU (middle)
36 core-to-core (inner)

7 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

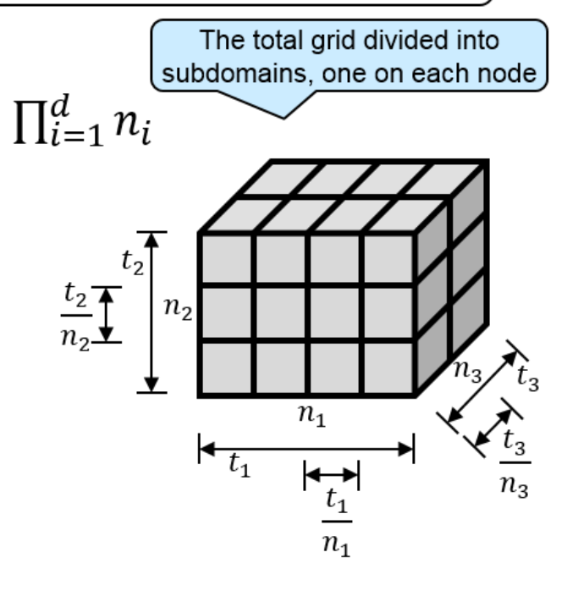
8 The Optimization Algorithm

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

Level 1 (= outer level = node level) on $\boldsymbol{N}$ 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 $d = 3$):
  $2\frac{t_2 t_3}{n_2 n_3}$, $2\frac{t_1 t_3}{n_1 n_3}$, $2\frac{t_1 t_2}{n_1 n_2}$
- Minimizing the communication costs c

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



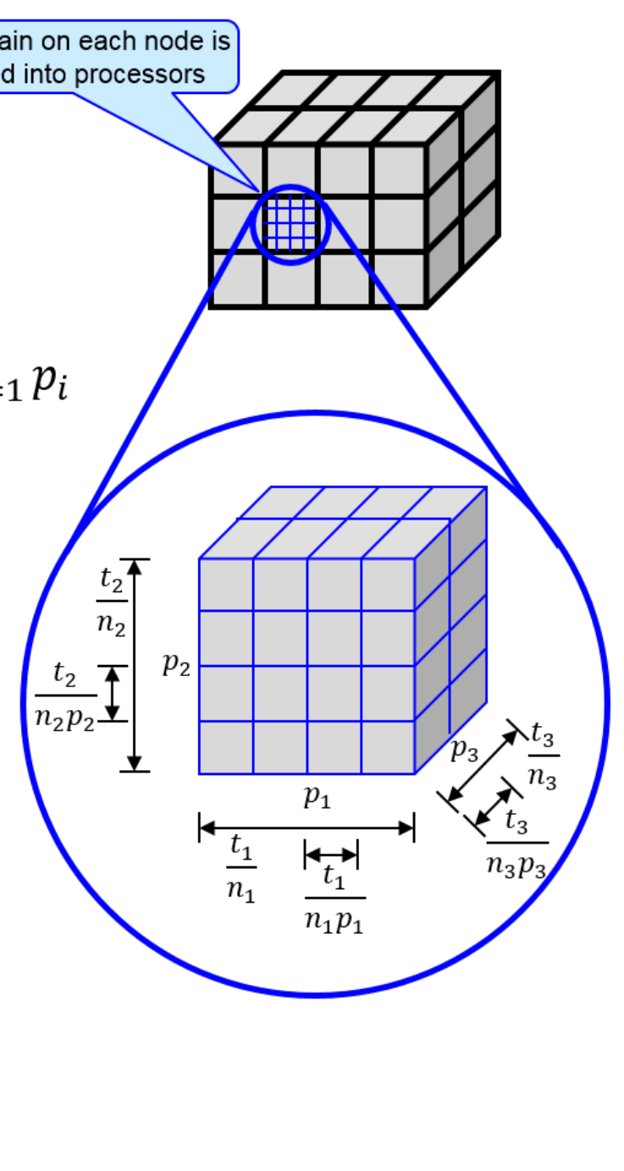
Summary of Level 1: One must search factors $(n_i)_{i=1,d}$
- that factorize N with $N = \prod_{i=1}^{d} n_i$
- and minimize the term $\sum_{i=1}^{d}\frac{n_i}{t_i}$

9 Second Level Optimization

Second Level Optimization

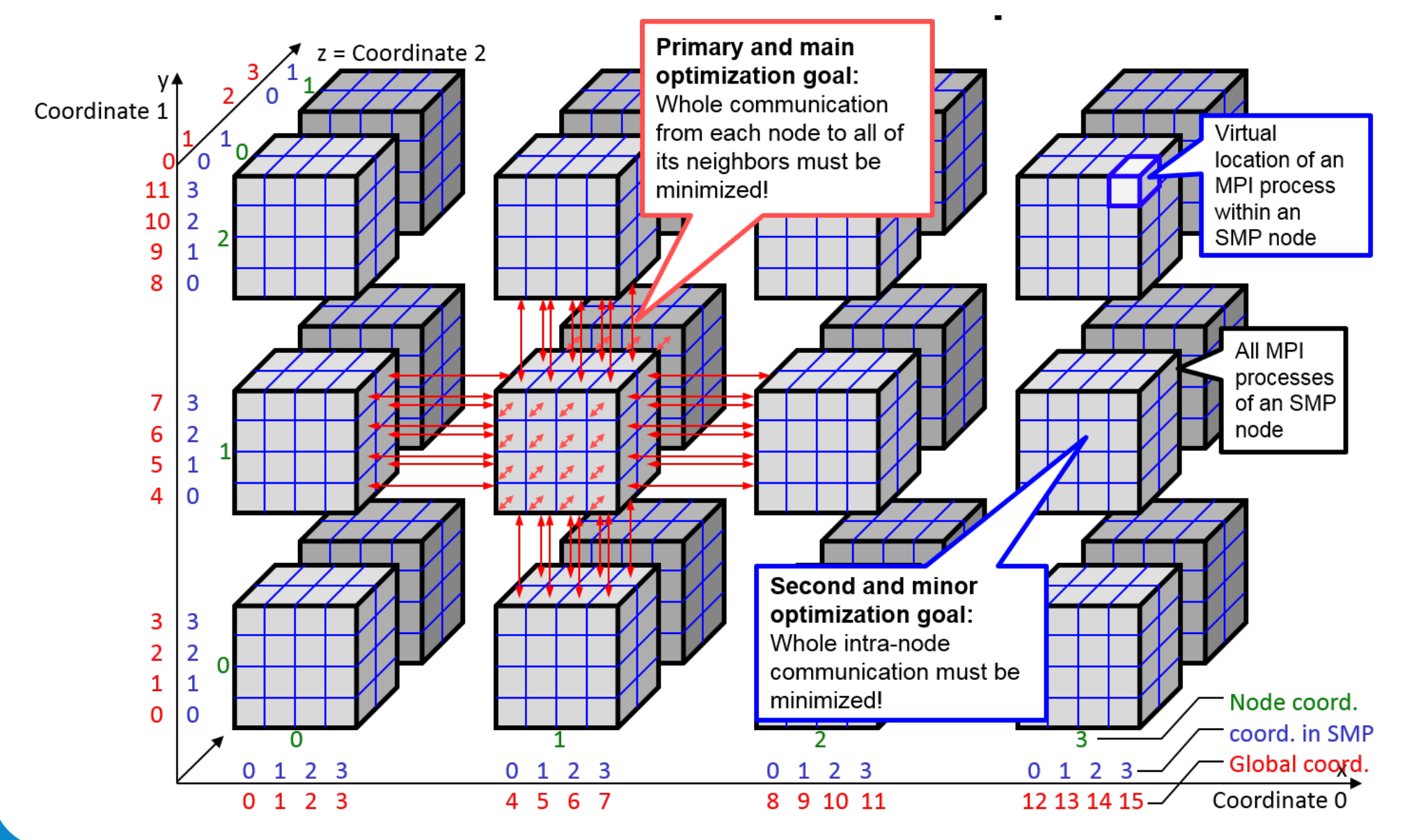
Level 2:
- Each node has $\boldsymbol{P}$ processors (or cores)
- Factorization of P into factors p_i with $P = \prod_{i=1}^{d} p_i$
- Communication costs in both directions of each dimension:
  $2\frac{t_2 t_3}{n_2 p_2 n_3 p_3}$, $2\frac{t_1 t_3}{n_1 p_1 n_3 p_3}$, $2\frac{t_1 t_2}{n_1 p_1 n_2 p_2}$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

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



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 $\sum_{i=1}^{d}\frac{n_i p_i}{t_i}$

10 Hierarchical Cartesian Domain Decomposition


Coordinate 1
z = Coordinate 2
Primary and main optimization goal: Whole communication from each node to all of its neighbors must be minimized!
Virtual location of an MPI process within an SMP node
All MPI processes of an SMP node
Second and minor optimization goal: Whole intra-node communication must be minimized!
Node coord.
coord. in SMP
Global coord.
Coordinate 0

11 Example

- **Given:** $\boldsymbol{d}$=3 dimensions, $\boldsymbol{N}$=625 nodes, $\boldsymbol{P}$=24 cores, and all $\boldsymbol{t_i}$ are identical
- → 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 $(nodes\ x\ cores)$ in each dimension: (25x1) X (5x6) X (5x4)
  - Total core numbers as used for MPI_Cart_create: 25 X 30 X 20
- Comparison with existing MPI_Dims_create:
  - MPI_Dims_create would result in 25 x 25 x 24
  - → complex mapping to the hardware is needed, see also slide 3, or no reordering is done → (25x1) X (25x1) X (1x24) with $\sum_{i=1}^{3} n_i = 51$ ☹ (instead of 35 ☺), → **46%** more **inter-node** communication!

12 Generalized Multi-Level Optimization

Given: $\boldsymbol{d}$-dimensional Cartesian grid with a total of $T = \prod_{i=1}^{d} \boldsymbol{t_i}$ elements
Number of hardware levels $\boldsymbol{L}$
and for each level the number of processors $\boldsymbol{N^{(l)}}$, $l = 1, L$

Communication costs on each level l:

$$c^{(l)} = 2\frac{T}{\prod_{k=1}^{l} N^{(k)}}\sum_{i=1}^{d}\frac{\prod_{k=1}^{l} n_i^{(k)}}{t_i} \qquad \text{for a factorization } N^{(l)} = \prod_{i=1}^{d} n_i^{(l)}$$

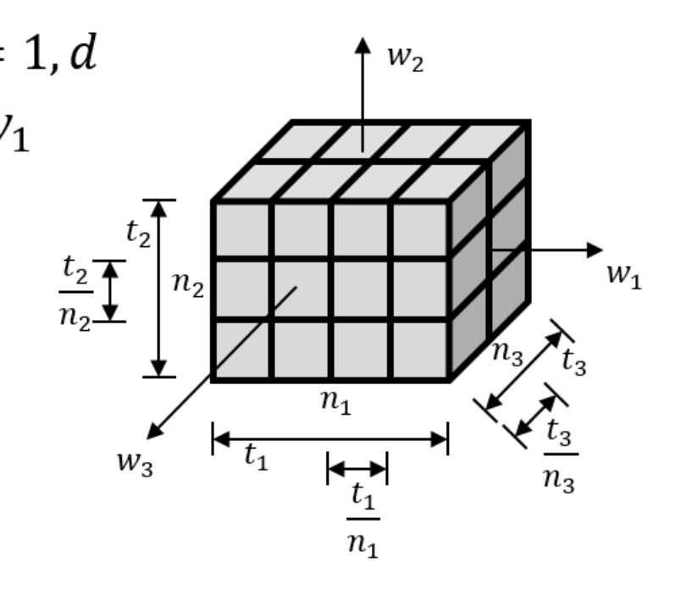
Search:
- On each level $l = 1..L$, one must search a factorization $\left(n_i^{(l)}\right)_{i=1,d}$ of $N^{(l)}$
- with $N^{(l)} = \prod_{i=1}^{d} n_i^{(l)}$
- and minimal sum $\sum_{i=1}^{d}\frac{\prod_{k=1}^{l} n_i^{(k)}}{t_i}$,

  i.e., with minimal sum $\sum_{i=1}^{d} a_i^{(l)} n_i^{(l)}$ with $a_i^{(l)} = \frac{\prod_{k=1}^{l-1} n_i^{(k)}}{t_i}$

13 Weighted Communication Volumes

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 $\boldsymbol{w_i}$, 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 $a_i^{(l)} = \frac{\prod_{k=1}^{l-1} n_i^{(k)}}{(t_i/w_i)}$



14 Using Weighted MPI_Dims_create

- Task: In general, on each hardware topology level l with $l = 1, \boldsymbol{L}$ and for given $\boldsymbol{N^{(l)}}$, find factorizations $\left(n_i^{(l)}\right)_{i=1,d}$ with $N^{(l)} = \prod_{i=1}^{d} n_i^{(l)}$ **and** the following sum is minimal: $\sum_{i=1}^{d} a_i^{(l)} n_i^{(l)}$ with $a_i^{(l)} = \frac{\prod_{k=1}^{l-1} n_i^{(k)}}{(t_i/w_i)}$
- Algorithm on each level $l = 1..L$

New MPI_Dims_weighted_create
- Sorting indexes $\boldsymbol{i = 1..d}$ so that the $(a_{i'})_{i'=1..d}$ to be non-decreasing
- Calculate all divisors of $\boldsymbol{N^{(l)}}$
- Loop over all non-increasing factorizations and find optimum according to (i.e. with $n_i \geq n_{i+1}$)
  - **1st criterion: a factorization is better if $\sum_{i=1}^{d} a_i^{(l)} n_i^{(l)}$ is smaller**
  - **2nd criterion: if the $\sum_{i=1}^{d} a_i^{(l)} n_i^{(l)}$ is the same then a factorization is better if $\Delta = n_1^{(l)} - n_d^{(l)}$ is smaller**
  - **3rd criterion: if $\sum_{i=1}^{d} n_i$ and $\Delta = n_1^{(l)} - n_d^{(l)}$ are the same then a factorization is better if $n_1^{(l)}$ is smaller**
- Revert the index mapping $\left(n_{i'}^{(l)}\right)_{i'=1..d} \rightarrow \left(n_i^{(l)}\right)_{i=1..d}$

Additional tricks:
- Calculate divisors only upto 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

Re-mapping of all process ranks according to $\left(n_i^{(l)}\right)_{i=1..d,l=1,L}$ and creation of the new Cartesian communicator according to algorithm in [2]

15 Proposed Interface

Substitute for / enhancement to existing MPI-1
MPI_Dims_create (size_of_comm_old, ndims, *dims*);
MPI_Cart_create (comm_old, ndims, dims, periods, reorder, **comm_cart*);

```
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] );
```

e.g., with 25*25*24 = **15000 processes** on **625 ccNUMA nodes** with **2 CPUs/node** and **12 cores/CPU**

e.g., { MPI_COMM_TYPE_SHARED, MPI_COMM_TYPE_NUMA }

e.g., 3 dimensions with a data grid with 1000 x 1100 x 950 elements → dim_weights[] = { 1.0/1000, 1.0/1100, 1.0/950 }

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): **30 x 25 x 20**

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

Next steps:
MPI_Comm_rank (comm_cart, &my_rank);
MPI_Cart_coords (comm_cart, my_rank, ndims, *coords*)

e.g., level_comms[0] is comm_old, level_comms[1 and 2] are the result recursively called MPI_Comm_split_type with the type_levels from above.

Same as above

Same as above

Same as above

16 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.

17 References

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

[2] Bill Gropp. 2018. Using Node Information to Implement MPI Cartesian Topologies. In *Proceedings of the 25nd European MPI Users' Group Meeting (EuroMPI '18)*, September 23–26, 2018, Barcelona, Spain. ACM, New York, NY, USA, 9 pages.

[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.