Specification Guideline Violations by MPI_Dims_create*

Jesper Larsson Träff
traff@par.tuwien.ac.at

Felix Donatus Lübbe
luebbe@par.tuwien.ac.at

Vienna University of Technology (TU Wien)
Faculty of Informatics, Institute of Information Systems
Research Group Parallel Computing

1. OBSERVATIONS

In benchmarking a library providing alternative functionality for structured, so-called isomorphic, sparse collective communication [4], we found use for the `MPI_Dims_create` functionality of MPI [3] for suggesting a balanced factorization of a given number p (of MPI processes) into d factors that can be used as the dimension sizes in a d-dimensional Cartesian communicator. Much to our surprise, we observed that a) different MPI libraries can differ quite significantly in the factorization they suggest, and b) the produced factorizations can sometimes be quite far from balanced, indeed, for some composite numbers p some MPI libraries sometimes return trivial factorizations (p as factor). This renders the functionality, as implemented, useless. In this poster abstract, we elaborate on these findings[1].

2. SPECIFICATION GUIDELINE FOR MPI_DIMS_CREATE

The `MPI_Dims_create` function, when called with a number of processes p (confusingly named `nnodes`) and a number of dimensions d, returns a factorization of p into d factors $p_0, p_1, \ldots p_{d-1}$ with $p_0 \geq p_1 \geq \ldots \geq p_{d-1}$ that must be "set to be as close to each other as possible" [3, Section 7.5.2, p. 293]. It is not further explained what "close... as possible" means, perhaps on purpose since factorization is a hard problem. One interpretation of "balanced" would be that the factorization must *minimize* the difference between any two factors, in particular between the largest and the smallest factor $p_0 = \max_{i=0,\ldots,d-1} p_i$ and $p_{d-1} = \min_{i=0,\ldots,d-1} p_i$. This is a well-defined optimization problem. It would be natural to pose as a *guideline* for an implementation of `MPI_Dims_create` that deviation from this balanced optimum be zero, or at least small.

*This work was supported by the Austrian FWF project "Verifying self-consistent MPI performance guidelines" (P25530).

[1]Unbeknownst to the authors, this and other obvious issues with `MPI_Dims_create` have been discussed at length at the MPI Forum [2].

EuroMPI '15 September 21-23, 2015, Bordeaux , France

ACM ISBN 978-1-4503-3795-3/15/09.

DOI: http://dx.doi.org/10.1145/2802658.2802677

```
int dims2create(const int p, int dims[]) {
  int f = (int)floor(sqrt((double)p));
  while (f>1&&p%f!=0) f--;
  dims[0] = f; dims[1] = p/f;
}
```

Figure 1: Finding a balanced factorization of p into $d = 2$ factors.

To check the extent to which standard MPI library implementations fulfill this guideline, we first implemented the straight-forward, optimal 2-dimensional balanced factorization algorithm shown in Figure 1. It suffices to check for divisors of p smaller than or equal to $\lfloor\sqrt{p}\rfloor$, since the largest such divisor f and its complement p/f will be the required, balanced factorization with $p/f \geq f$. This algorithm takes $O(\sqrt{p})$ operations. Note that this is *not* a polynomial-time algorithm, since the input to the algorithm is the number p which can be represented in $\lceil\log_2 p\rceil$ bits.

In Table 1 we report the results from executing `MPI_Dims_create` for four different MPI libraries available to us for *all* process counts p from 1 up to the indicated maximum number and varying dimensions d. We count the total number of *violations* of the specification guideline where the difference between largest and smallest factor of the MPI generated factorization deviates from the optimum, and the number of *strong violations* which are the instances where a trivial factorization is returned for a non-prime number. The latter are particularly extreme violations of the specification guideline, and likely to be highly undesirable for the application programmer.

We see that all four libraries from some number of processes violate the optimality performance guideline. Obviously, the MPI libraries apply heuristics in their implementation. The NEC MPI library fares the best in terms of violations[2] and exhibits no strong violations. The OpenMPI library applies a slow heuristic, and from $p = 10^4$ it was not feasible to wait for the results. The times required for the libraries and our optimal, enumerative method are shown in Table 1. The durations indicate that requiring an optimally balanced factorization and thus fulfilling the specification guideline is indeed feasible.

For $d = 2$ the first violation with MPICH occurs for 28 processes, which is factorized into 14×2 instead of 7×4. Its first strong violation occurs for $p = 361$, which is factorized

[2]Since submission of the poster, a bug that made the NEC MPI library crash at $p = 994008$ and caused violations even for $d = 2$ was fixed.

d	p	MPICH 3.1.4/MVAPICH2 2.1		NEC MPI 1.3.1		Open MPI 1.8.6		Optimal
		Vio. (strong)	Dur. [s]	Vio. (strong)	Dur. [s]	Vio. (strong)	Dur. [s]	Dur. [s]
2	10^2	7 (0)	0.000028	0 (0)	0.000084	1 (0)	0.000253	0.000022
	10^4	3683 (37)	0.005492	0 (0)	0.010020	279 (0)	46.959451	0.005980
	10^6	475258 (1483)	2.882608	0 (0)	2.077307	—	—	6.190137
3	10^2	0 (0)	0.000031	0 (0)	0.000076	0 (0)	0.000256	0.000037
	10^4	1558 (37)	0.005664	55 (0)	0.008977	124 (0)	46.959950	0.013031
	10^6	251823 (1483)	2.900862	7396 (0)	1.853388	—	—	10.871350
4	10^2	0 (0)	0.000033	0 (0)	0.000064	0 (0)	0.000258	0.000038
	10^4	543 (37)	0.005877	30 (0)	0.007487	34 (0)	46.960326	0.014489
	10^6	113196 (1483)	2.921363	4819 (0)	1.602826	—	—	11.594636
5	10^2	0 (0)	0.000035	0 (0)	0.000056	0 (0)	0.000259	0.000036
	10^4	168 (37)	0.006108	16 (0)	0.006365	1 (0)	46.959648	0.014040
	10^6	45245 (1483)	2.945329	2027 (0)	1.421701	—	—	11.756804
6	10^2	0 (0)	0.000037	0 (0)	0.000028	0 (0)	0.000261	0.000036
	10^4	78 (37)	0.006462	13 (0)	0.005535	0 (0)	46.961428	0.012988
	10^6	17095 (1483)	2.983855	698 (0)	1.301888	—	—	11.471130
10	10^2	0 (0)	0.000053	0 (0)	0.000030	0 (0)	0.000273	0.000036
	10^4	38 (37)	0.008319	0 (0)	0.004680	0 (0)	46.963084	0.011367
	10^6	1656 (1483)	3.180510	21 (0)	1.142547	—	—	10.364155
20	10^2	0 (0)	0.000097	0 (0)	0.000041	0 (0)	0.000314	0.000038
	10^4	37 (37)	0.012790	0 (0)	0.005362	0 (0)	46.966152	0.011620
	10^6	1483 (1483)	3.638120	0 (0)	1.168740	—	—	10.301055

Table 1: Number of guideline violations (Vio.) and duration (Dur.) [seconds, measured on an AMD Opteron 6134] for d-dimensional factorization for different maximum number of processes p. Strong violations are trivial factorizations of composite p. Column "Optimal" lists the time for optimally balanced factorization.

into 361×1 instead of the optimal 19×19: The heuristic is not able to factorize a square into its roots! The largest p in the considered range causing a strong violation is $p = 524137$, which should be factorized into 941×557. The first violation with Open MPI occurs for 72 processes, which is factorized into 12×6 instead of 9×8, the last one for $p = 9945 = 153 \times 65$ instead of 117×85.

3. HIGHER DIMENSIONS

In order to gauge the quality of the solutions delivered by the MPI libraries also for higher dimensions, we implemented an exhaustive, enumerative method that generates *optimally balanced factorizations* in d dimensions[3]. For given p and dimension d it tries all divisors of p up to $\lfloor \sqrt[d]{p} \rfloor$, and applies the factorization recursively on dimension $d-1$, recording the so far best balanced factorization at the base when $d = 1$. A very rough estimate of the total number of steps not using any number theoretic properties on the number of factors of p is $O(\sqrt{p}^{\log d})$ which is exponential in both $\log_2 p$ and $\log d$. Table 1 indicates that the actual time may be much better and not dependent on d. Note that the number of essentially distinct factorizations is small, namely at most $p/\log p$ for $p \neq 144$ [1]. Optimally balanced d-factorization seems quite feasible for the small p relevant for MPI.

All four MPI libraries show violations for $d > 2$. Strong violations, independent of d, are only produced by MPICH. For multiples of 2^{10} no library shows strong violations, whereas some square numbers violate strongly. The greatest p causing a violation is 10^6, with NEC MPI for $d \in \{4, 5\}$ and MPICH for $d = 10$.

[3]Our implementation in a form suitable for MPI libraries is available at www.par.tuwien.ac.at/Downloads/TUWMPI/tuwdims.c

4. SUMMARY

The MPI standard seems reasonably clear what is expected from `MPI_Dims_create`, but common library implementations obviously deviate from the requirements, sometimes badly (strong violations). Whether the "closeness" requirement of the current MPI standard [3] is the most sensible is indeed debatable [2]; our factorization implementation can easily accommodate other cost functions depending on the d factors. Since libraries apply different heuristics, and thus may deliver factorizations of different quality, portability of applications that rely on `MPI_Dims_create` may suffer.

We note finally that `MPI_Dims_create`, as one of few MPI functions, does not take a communicator argument, and thus has no possibility of suggesting factorizations that fit well with the context in which the corresponding Cartesian communicator will be created. This is peculiar.

5. REFERENCES

[1] J. F. Hughes and J. O. Shallit. Estimating the number of multiplicative partitions. *Rocky Mountain Journal of Mathematics*, 17:797–813, 1987.

[2] MPI Forum. Ticket #195: Topology awareness in `MPI_Dims_create`, 2009. Accessible from https://svn.mpi-forum.org/trac/mpi-forum-web/ticket/195, last visited 1.7.2015.

[3] MPI Forum. *MPI: A Message-Passing Interface Standard. Version 3.1*, June 4th 2015. www.mpi-forum.org.

[4] J. L. Träff, F. Lübbe, A. Rougier, and S. Hunold. Isomorphic, sparse MPI-like collective communication operations for parallel stencil computations. In *Recent Advances in the Message Passing Interface (EuroMPI)*, 2015. To appear.