

# PE mapping and the contention problem on the T3E

Matthias Müller\* and Michael M. Resch†

October 21, 1998

## Abstract

Widespread communication patterns found in most parallel programs with domain decomposition can lead to contention and thus to performance loss. We show that improved mapping of the communication topology on the physical processing elements (PEs) is one possible solution to the problem.

## 1 Introduction

Benchmarks are widely used to measure performance. On one hand they should be specific enough to serve as a guideline for improvements, on the other hand they should reflect the demand of real applications. The Message Passing Interface [1] (MPI) is a widespread library for communication on parallel computers. Most MPI benchmarks concentrate on point to point and global communications [2]. Some benchmarks also use kernels of applications or even complete applications [3, 4].

In the course of profiling and improving some of our parallel applications we encountered some problems that were not represented in benchmarks results available to us so far. Because we think that some of these problems are of general interest we try here to present the results and make proposals for possible new benchmarks.

One situation that occurs in many parallel application is that the algorithm performs a *compute - communicate - compute* loop. In this case all PEs communicate at the same time. This can overburden the communication network, the so called contention occurs.

---

\*Institute of Computer Applications 1, University of Stuttgart, Pfaffenwaldring 27, D-70550 Stuttgart, Germany, e-mail: matthias@ica1.uni-stuttgart.de

†High Performance Computing Center Stuttgart (HLRS), Allmandring 30, D-70550 Stuttgart, Germany, e-mail: resch@hlrs.de

## 2 PE mapping and MPI

MPI identifies a PE with its *rank*. In the first place there is no information about a topology associated with those ranks. On the T3E the PEs are numbered in a way that ranks with a small difference are likely to be close in the communication network. It is clear that it is impossible to provide a good solution for all situations.

MPI therefore offers to create special process topologies. One possibility is to create an n-dimensional cartesian grid with a call to `MPI_Cart_create`. One option allows the ranks to be **reordered**, this gives the implementation the possibility to remap the PEs to get the best performance with the specific hardware. The PEs of the T3E are physically organized in a bidirectional three dimensional torus. This is well suited for the cartesian grids found in applications with domain decomposition.

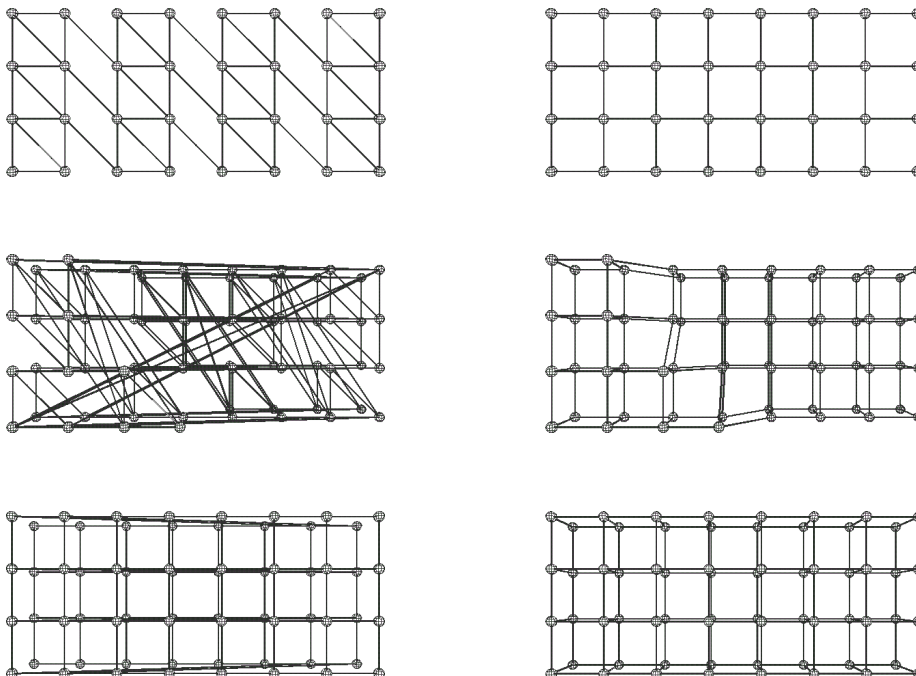


Figure 1: Different mappings onto PEs. Left: generated with `MPI_Cart_create`. Right: generated with `MPI_Cart_create` that performs reordering.

During our investigations we found that MPI on the Cray does no reordering of PEs. We therefore tried to do an optimized reordering of PEs based on the information that we get from the *sysconf* call. The algorithm chosen for optimization is to sort physical coordinates that we get from *sysconf* according to x,y and z-coordinates. There are 6 permutations of the (x,y,z) triplet. Each

of them is tested with respect to average hop count and the optimum ordering is chosen. Results for different cartesian communicators are shown in Fig. 1.

The result of the optimization measured in reduction of average hop counts is given in Tab. 1 for varying dimensions of the cartesian communicator.

Grid	without reordering	with reordering
2x2x2	1.3	1.3
3x3x3	2.5	1.8
4x4x1	2	1
4x4x4	2.8	1.8
8x4x1	2.2	1
8x4x2	1.5	1

Table 1: Average number of hops a message has to travel.

### 3 Benchmarks

#### 3.1 Pair communication

##### 3.1.1 Description:

For contention to occur several PE have to communicate simultaneously. We choose a communication pattern where the PEs communicate pairwise with each other (see Fig. 2).

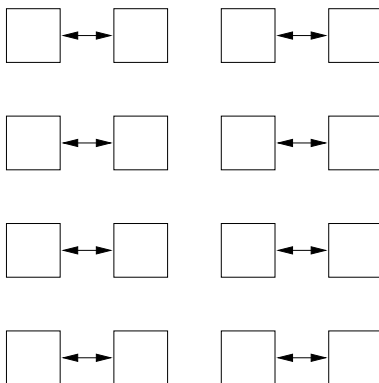


Figure 2: Topology of the benchmark problem

It's clear that this does not reflect any real application but the design goal was to have completely independent communications. The messages are sent with `MPI_Isend` and received with `MPI_Irecv` (see Fig. 3). Because the hardware of the T3E is capable of bidirectional communication the messages could be sent

and received at the same time. We could have used `MPI_Sendrecv` but the use of the immediate send and receives reflects the use in application where some computations are done between the send/receive calls and the `MPI_Waitany`.

The topology is created with a call to `MPI_Cart_create` and `MPI_Cart_shift`. This gives the underlying MPI implementation the possibility to perform an optimized mapping onto the hardware.

```
MPI_Isend(sendfield,size,MPI_DOUBLE,partner,tag,comm,&request[0]);
MPI_Irecv(recvfield,size,MPI_DOUBLE,partner,tag,comm,&request[1]);
MPI_Waitall(2,request,status);
```

Figure 3: Code segement of the benchmark code.

We define the bandwidth as the amount of data that can be sent by one PE per second.

### 3.1.2 Result:

From the observed maximum bandwidth of about 200MB/s with 2 PEs (see Fig. 4) we conclude that at least to some extent bidirectional communication is performed. One link of a T3E-900 has a bandwidth of about 300MB/s for MPI.

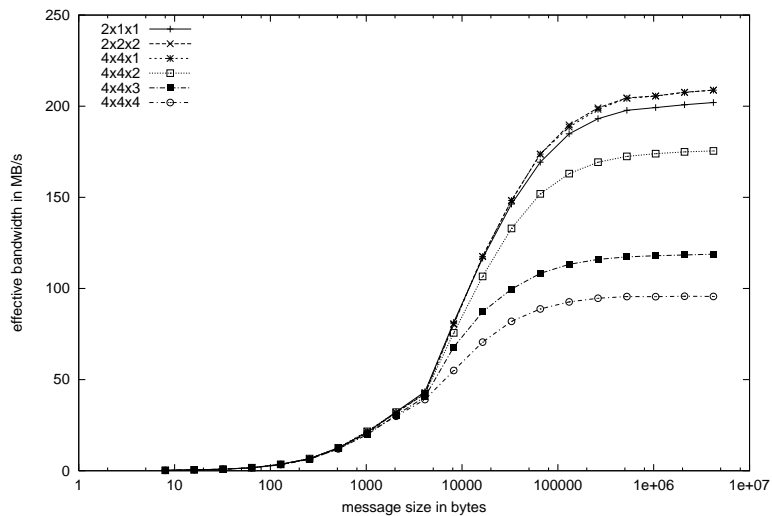


Figure 4: Bandwith depending of message size and number of PEs. The topology is created with `MPI_Cart_create`.

The contention is obvious in Fig. 4. The bandwidth for message sizes above

10.000 bytes drops with increasing number of PEs. For 64 PEs the bandwidth for large messages drops to less than half of the uncongested value.

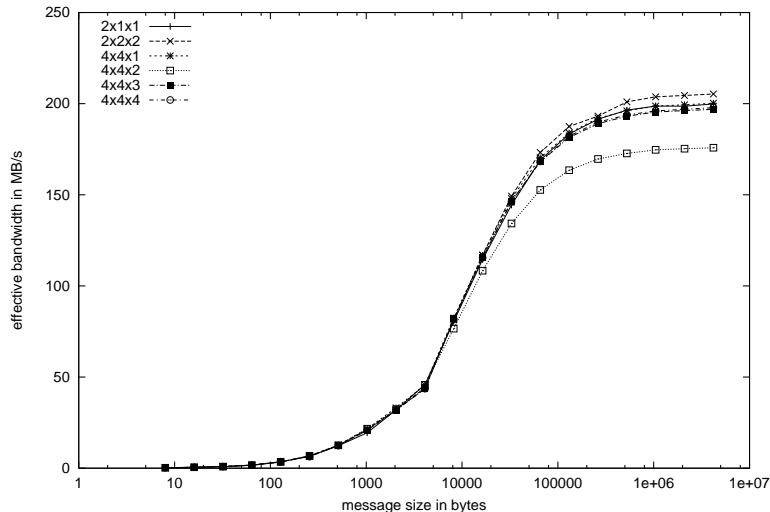


Figure 5: Bandwidth depending of message size and number of PEs. The topology is created with an optimized mapping.

Using the optimized mapping we can reduce congestion dramatically. Fig. 5 shows that nearly for all cases we can get better results.

### 3.1.3 Extension to cyclic communication:

To find out about the behaviour of the toroidal communication network we extended the example. Now communication is done between all neighbouring nodes in the x-direction. Each process has to talk to a left and a right neighbour. In principal we would assume that all communication can be done within one single communication step. This should also hold for cyclic communication patterns assuming that all processes are equal with respect to network topology.

For our test example we created a 8x4x1 cartesian communicator. On the one side we did tests with a periodic communicator and compared these results to tests done with a non-periodic. The tests with a periodic communicator include cyclic communication. For the non-periodic communicator no cyclic communication is done. A first guess would be that the additional communication should cost more. This would be true on machines with a standard network. However, the toroidal mesh of the T3E should show no difference. All tests were done using a fixed message size of 4MB. All measurements were done 100 times and the accumulated times are given. Without setting `MPI_BUFFER_MAX` one communication should take about 2.7 seconds.

First results showed a significant difference which was independent of the ordering method chosen. The overhead for cyclic communication was significant for both kinds of communicators as can be seen in Tab. 2.

	without reordering	with reordering
non-periodic	8.2 sec	5.9 sec
periodic	8.7 sec	6.8 sec

Table 2: Time for twosided exchange communication.

The striking point here is that even for a periodic communicator we would expect that all communication could be done within two communication cycles. This should take 5.4 seconds. For the Cray T3E it takes more than 3 communication cycles. For our own method there is still some overhead.

To find out what could be done about that problem we did some more tests with different communication patterns. The result was that once cyclic communication is involved the performance of the network decreases. Further investigation made us think that we should optimize our communication pattern to make it easier for the system to cope with a fully loaded network. So we explicitly decoupled the communication and did it in two steps. For this, every second processor row started with the communication to the right neighbour. Every other second row did the same for the left one. In the next step all processors switched to the other neighbour. This way we thought that within two communication steps all communication should be finished. The results are given in Tab. 3.

	without reordering	with reordering
standard	8.7 sec	6.8 sec
new	9.2 sec	5.4 sec

Table 3: Time for twosided exchange communication using a two step communication scheme.

Although the decoupling of communication should make the Cray MPI version faster it actually slows it down even more. We suspect that by prescribing the communication pattern we loose the possibility to optimize the communication for the bad mapping of processes. For our own mapping strategy we find that we actually can achieve peak bandwidth for that communication pattern.

In this chapter we have seen that completely independent messages can interfere with each other: the communication gets slower because the network of the parallel computer gets congested. In the next chapter we will discuss a communication pattern that can be found in real applications and show how our modified mapping can provide a solution to the contention problem.

## 3.2 Application benchmark

### 3.2.1 Description

The topology in this case is a three dimensional cartesian grid. This corresponds to a standard domain decomposition that can be found in many applications. First the communication between the neighbours of one dimension is performed. After it is completed the communication for the next dimension is started. The topology is again created with a call to `MPI_Cart_create`.

We defined the bandwidth as the amount of data a PE can send. Because there are two sends done simultaneously in different directions the theoretical peak bandwidth is 600MB/s for MPI.

### 3.2.2 Result

The observed bandwidth is always below 200MB/s (see Fig. 6). There seems to be just one send active at a time.

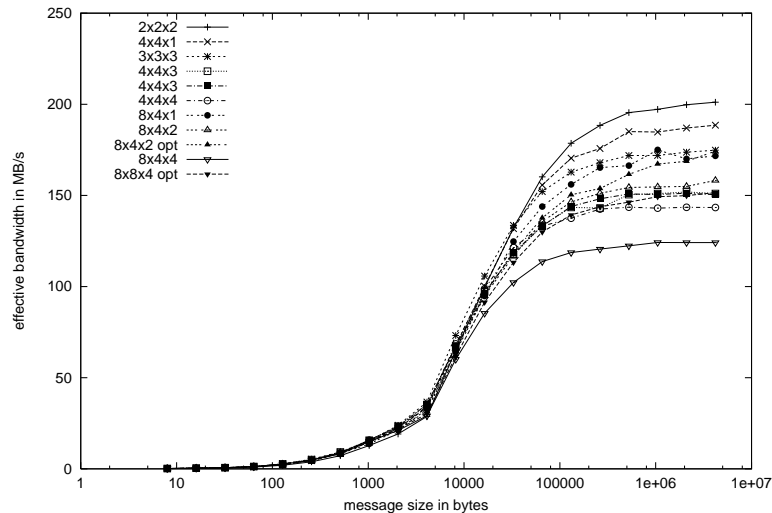


Figure 6: Bandwidth depending of message size without reordering of PEs.

Again we see the congestion. In this case the bandwidth decreased from 200MB/s to 124MB/s for 128 PEs (8x4x4). The decrease is less pronounced than in the benchmark case because the different directions are effected to different degrees. A poor performance of one direction is compensated by a good performance of another one. In principle there is no reason for a congestion, the layout of the T3E is a bidirectional three dimensional torus[5, 6]. Two neighbours in a cartesian grid should have a direct connection.

However the operating system does not know whether a request for 64 PEs results in a 4x4x4 or 8x8 grid. It guarantees just a connected area in physical

PE space. This provides a first approach to local communication between PEs.

But even if the layout of the physical PEs corresponds to the requested grid contention occurs. `MPI_Cart_create` does not reorder the ranks to optimize the mapping.

Instead of calling the original `MPI_Cart_create` we used our own version that performs reordering as described above. The result is shown in Fig. 7. The bandwidth drops only to 154MB/s. We checked the quality of a mapping by looking at the average number of hops a message has to travel. The hop count of the grids created by our mapping was between 1 and 1.8 ( see Table. 1). Whenever the layout of the physical PEs corresponded to the requested grid the hop count was one. The performance is between 2 and 40 percent better than the unoptimized mapping for PE numbers larger than 8. Contention only occurs when the topology is strongly disturbed. In the case of 8x8x4 PEs the domain was practically separated into two partitions with 128 PEs. Small distortions are handled well as can be seen in Fig. 1.

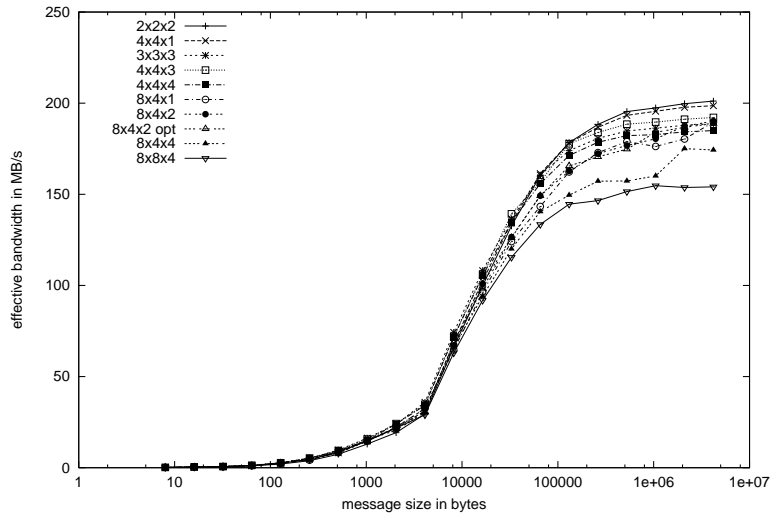


Figure 7: Bandwidth depending of message size with reordering of PEs.

## 4 Conclusion

We have shown that the mapping of communication topology onto the underlying hardware is important for benchmarks reflecting communication patterns found in real applications. This opens a wide field of optimization. Our first finding was that the application partition of a T3E should be configured to be undisturbed by OS or CMD PEs in its area. This will not only provide better

mapping for the standard programs but will also enable optimized mapping algorithms to find the optimal solution. Furthermore fragmentation does not only affect the jobs you start on a system, but also the performance of the already running jobs. An optimized batch system should take this into account.

Applications which suffer under contention and communicate in a cartesian grid should use a version of `MPI_Cart_create` with reordering. The method we presented here provides such an optimized mapping. It was easy to implement and the overhead of the optimization process is neglectible compared to simulation times of real applications.

## 5 Acknowledgements

The authors would like to thank Monika Wierse from SGI/Cray for support in using the T3E and Thomas Beisel and Rolf Rabenseifner from HLRS for useful hints and informations.

## References

- [1] Message Passing Interface Forum. *MPI: A Message Passing Interface Standard*. <http://www.mpi-forum.org/>.
- [2] Michael Resch, Holger Berger, and Thomas Boenisch. A comparison of MPI performance on different MPPs. In Marian Bubak, Jack Dongarra, and Jerzy Wasniewski, editors, *Recent Advances in Parallel Virtual Machine and Message Passing Interface*, Lecture Notes in Computer Science, pages 75–83, Heidelberg, 1997. Springer.
- [3] Michael Resch, Thomas Boenisch, and Holger Berger. Performance of MPI on a cray T3E. Technical report, Third European CRAY-SGI MPP Workshop, Paris (France), Sept. 1997.
- [4] Roger Hockney and Micheal Berry. Report-1. Technical report, PARK-BENCH Committee, 1994. <http://www.netlib.org/parkbench/>.
- [5] Steven L. Scott. Synchronisation and comunication in the T3E. In *Int. Conf. on Architectural Support for Programming Languages and Operating Systems proceedings*, volume 31, pages 26–36, October 1996.
- [6] Steven L. Scott and Gregory M. Thorson. Adaptive routing in a high performance 3d torus. In *HOT Interconnects*, volume IV. Stanford University, August 1996.