University of Zurich[UZH]

Zurich Open Repository and Archive
University of Zurich
Main Library
Strickhofstrasse 39
CH-8057 Zurich
www.zora.uzh.ch

Year: 2013

Improving communication performance of sparse linear algebra for an atomistic simulation application

Pousa, C ; Hutter, J ; Vandevondele, J

DOI: https://doi.org/10.3233/978-1-61499-381-0-405

Posted at the Zurich Open Repository and Archive, University of Zurich
ZORA URL: https://doi.org/10.5167/uzh-100698
Conference or Workshop Item
Accepted Version

Originally published at:
Pousa, C; Hutter, J; Vandevondele, J (2013). Improving communication performance of sparse linear algebra for an atomistic simulation application. In: International Conference on Parallel Programming – ParCo 2013, Garching, Germany, 10 September 2013 - 13 September 2013, 405-414.
DOI: https://doi.org/10.3233/978-1-61499-381-0-405

Improving Communication Performance of Sparse Linear Algebra for an Atomistic Simulation Application

Christiane POUSA [a,1], Jürg HUTTER [a] and Joost VANDEVONDELE [b]
[a] *University of Zurich - Institute of Physical Chemistry, Zurich, Switzerland*
[b] *Swiss Federal Institute of Technology Zurich ETH Nanoscale Simulations - Department of Materials Zurich, Switzerland*

Abstract.
The design of modern parallel machines leads to powerful machines, but with complex architectures and hierarchical topologies. As a result, communication overheads associated with hardware asymmetry and interconnection network increase. In order to achieve scalable performances on these machines, it is essential to reduce communication costs on parallel applications such as CP2K. From computational chemistry domain, CP2K is a real-world parallel application that performs atomistic and molecular simulations. A linear-scaling DFT implementation based on an efficient sparse linear algebra kernel allows CP2K to simulate a million of atoms. Since this kernel is communication bound, the hardware asymmetry and interconnection network of current machines leads to a slowdown on CP2K performance for large number of processes. In this paper, we introduce a heuristic based process mapping to reduce the communication costs. It takes into account the machine topology and the sparse linear algebra kernel communication pattern to map processes over the machine. Results show that our process mapping provides up to 30% of performance improvements when compared with the default implementation of CP2K.

Keywords. atomistic and molecular simulation, process mapping, multicore architecture

1. Introduction

Modern large scale parallel machines are assembled with an increased number of multicore compute nodes interconnected by an efficient network. This design leads to powerful machines, but with complex architectures and hierarchical topologies. Owing to this, communication costs of parallel applications increase due to the variations on latency and bandwidth for large number of processes. In order to achieve scalable performances on these machines, parallel applications such as CP2K [1] must reduce all-to-all communication, use virtual and physical local groups for point-to-point communication and

[1]Corresponding Author: University of Zurich, Switzerland; E-mail: pousa.christiane@gmail.com.

guarantee locality for communicating groups. The latter can be achieved through the use of process mapping, which efficiently maps the application processes over the machine to reduce latency and increase available bandwidth [2,3].

CP2K is a real-world parallel application from computational chemistry domain that is routinely used to perform atomistic and molecular simulations [1,4]. CP2K supports atomistic and molecular simulation of solid state, liquid, molecular and biological systems. These simulations allow researchers to concept novel materials and to better understand existing ones. CP2K package is composed of state-of-the-art methods such as DFT (density functional theory), semi-empirical NDDO approximation and second order Møller-Plesset perturbation theory, that produce accurate simulations [5,6,7]. Additionally, CP2K implements an efficient algorithm of sparse linear algebra kernel (sparse matrix-matrix multiplication), that makes it capable to simulate systems with up to a million of atoms [8].

From the implementation side, CP2K exploits parallelism to accelerate its simulation on large scale parallel machines. A hybrid model that combines message passing (MPI) and shared memory model (OpenMP and CUDA) allows CP2K to fully exploit the current parallel machines. However, even such a hybrid model, can not always guarantee strong scalability for CP2K. Communication overheads due to the hardware asymmetry and interconnection network of modern machines still remain. An affinity module has been recently included in CP2K package to reduce the hardware asymmetry for memory accesses within a compute node [9]. However, on the inter-compute node level, any locality that takes into account the network topology for communicating processes is guaranteed by CP2K. Therefore, communication overheads associated to variations on latency and bandwidth of interconnection networks remain, reducing its overall performance.

In this paper, we focus on CP2K simulations that are governed by the linear-scaling DFT algorithm. More precisely, on the study and performance improvement of the inter node communication for the sparse matrix-matrix multiplication kernel. We first study the impact of different process mapping strategies on this kernel. Such study is performed using a synthetic benchmark that mimics the behavior of the sparse matrix-matrix multiplication kernel. The synthetic benchmark allows us to isolate the communication behavior of this kernel from the remaining kernels of CP2K, enabling a better understanding of the strategies.

In light of such results, we introduce a heuristic to reduce the communication overhead and consequently, improve the sparse matrix-matrix multiplication kernel performance. The heuristic relies on the Hilbert space filling curve, but it also takes into account the network topology and the communication pattern of the kernel. Using the mapping produced by Hilbert curve, a communication cost function is computed for the different directions of the CRAY XE6 interconnection network. The less expensive direction is used to map CP2K processes following the Hilbert curve. The results show that our heuristic reduces communication overheads, improving CP2K overall performance.

This paper is organized as follows. In Section 2, we discuss some related work. Section 3 presents the study of different process mapping strategies on the sparse matrix-matrix multiplication kernel. Section 4 introduces our heuristic to reduce the communication overhead for the kernel. The results with the proposed heuristic are reported in Section 5. Finally, we present our conclusions and future work in Section 6.

2. Related Work

A number of works have introduced algorithms to improve communication performance for parallel applications on high performance machines. Most part of the research has been done for applications with regular communication patterns [10,11,12,13,14,15,2]. Process mapping strategies and load balancing that deal with irregular communication patterns have also been proposed [3,16].

In [10,11], the authors present a process placement policy and a rank reordering strategy to map MPI applications on multi-core systems. Both proposals rely on graphs that describes the hardware topology and the application communication pattern. For the process placement policy, the authors uses Scotch software[2] to match the graphs and generate a mapping for the MPI processes. The rank reordering also uses the hardware topology information, but in this case a tree representation is used instead of the complete graph. The reordering is generated by an algorithm that matches the tree representation of the hardware with the communication graph of the application.

Eduardo et. al. analyze the performance impact of load balancing strategies in a weather forecasting application [12,13]. They rely on a runtime system named *Charm++* that is able to obtain the communication pattern of the application and information about the underling parallel machine. Using such information the runtime system applies load balancing algorithms that equalizes the load on the processors, while taking into account the communication costs. Based on the characteristics of the application, the authors propose a load balancing algorithm that uses the Hilbert space filling curve definition [17]. The proposed load balancing traverse the tasks of the application with this curve to recursively bisect it according to the load of each task. As a result, tasks that communicate more are placed close on the machine. In the works presented above a flat representation of the network is used, no information about the network topology is exploited. However, several works have shown that the network representation can have an important effect on the generated mapping [14,15,2,3,16].

In [14], an efficient process remapping for hierarchical clusters is proposed. The authors have designed MPI topology functions to take into account the network topology when mapping processes. The network topology is obtained through the computation of the distance between two nodes in the cluster. They rely on available tools (e.g. ibtracert) to discover the distances. Topology-aware mapping strategies for torus networks are proposed in [15,2]. In applications with point-to-point communications, tasks that communicate more are placed on nearby compute nodes. Therefore, the topology-aware mapping reduces the distance between the communicating tasks, reducing the number of hops for communication. When collectives with sub-communicators are used, the proposal of the authors is to increase the available bandwidth by providing more possible paths for the messages. In both cases the network topology is exploit to generate the process mapping. Topology mapping strategies are proposed for large-scale parallel architectures in [3]. The strategies aim to reduce congestion and dilation perceived by the application. To do so, the application communication graph, the network topology of the machine and the processes allocation are used as inputs for different algorithms. The proposed algorithms are (i) greedy ,(ii) recursive bisection, (iii) graph similarity and (iv) simulated annealing. The network and communication representation allows the support of heterogeneous

[2] http://www.labri.fr/perso/pelegrin/scotch/

networks and applications with irregular communication patterns. In [16], the authors present a load balancing algorithm that models latencies and bandwidths of the parallel machine. This model allows the algorithm to represent the distances and communication costs among hardware resources. The algorithm aims to improve the application performance by increasing core usage and communication performance. They rely in a complete representation of the machine, taking into account not only the compute node architecture, but also the interconnection network.

3. Applying Process Mapping on Sparse Linear Algebra

In this section, we present the study of the impact of different process mapping strategies on the sparse linear algebra kernel. We start presenting the experimental environment used in our study. After that, we describe the main characteristics of the considered kernel. We end the section with our study.

3.1. Experimental Environment

We have selected a representative high performance computing machine to conduct our experiments. In this section, we describe the hardware details of the machine and the software environment used in our experiments.

The selected machine is a CRAY XE6 composed of 1496 compute nodes based on 32-core AMD Interlagos processor. Each core has a private cache L1 of (16 KB) and L2 (2 MB), and each eight cores share a LLC. The compute nodes are interconnected by a high performance networking with Gemini 3D torus interconnect. The machine is named Monte Rosa and it is hosted at CSCS-Switzerland[3]. Concerning the software environment of the machine, it runs the Cray Linux Environment operating system version 4.0.3. For code compilation, we have used the GNU Compiler Collection version 4.6.3 with the Cray MPICH2 Message Passing Interface for the CRAY machines (http://www.epcc.ed.ac.uk/t3dmpi/Product).

3.2. Sparse Linear Algebra Kernel

The sparse linear algebra kernel of CP2K implements a sparse matrix-matrix multiplication and it is named DBCSR library[4]. The library is based on a distributed blocked compressed sparse row to represent the matrices needed in a CP2K simulation.

On DBCSR, the algorithm used for the matrix multiplication is the Cannon one [18]. In this algorithm, the operations are performed in two-dimensional matrices in a distributed way. To compute the matrices, process are organized on a virtual two dimensional grid, which is after used to distribute the matrices for each process. To compute its data, each process p communicates with four neighbors, left-right (rank$\pm$1) and up-down (rank$\pm\sqrt{P}$). This communication is necessary to transfer/get the data of the two input matrices to/from other processor. The main characteristic of cannon algorithm is that memory requirements are not dependent of the number of processors.

[3] www.cscs.ch

[4] http://www.hector.ac.uk/cse/distributedcse/reports/cp2k03/cp2k03/node6.html

The communication of cannon algorithm implemented in DBCSR uses asynchronous send and receive MPI operations. It is divided in two steps, the up-down communications and the left-right ones. On each iteration of the multiplication process, the MPI tasks compute what and to whom they have to send/receive data. After that, they wait for their data to perform the multiplication.

3.3. Impact of Process Mapping Strategies

As aforementioned in Section Related Work, a number of process mapping solutions have been proposed by researchers. Cover all of them would be a huge work. Therefore, we have selected some representative process mapping strategies to study. The selected strategies are: (i) packed, folded and round-robin, (ii) Hilbert space filling curve [17,19] and (iii) Greedy, Recursive bisection and Graph Similarity (RCM) strategies proposed in [3]. Packed, round-robin and folded strategies are provided by the runtime system of Cray environments [20].

```
// Cannon iterations
for(int i = 0; i < ITERATIONS; i++)
{
    MPI_Request req[4];
    MPI_Status stat[4];

    MPI_Irecv(recvLeft,n,MPI_DOUBLE,neighborLeft,0,CommCart,&req[0]);
    MPI_Irecv(recvDown,n,MPI_DOUBLE,neighborDown,1,CommCart,&req[1]);
    MPI_Isend(sendRight,n,MPI_DOUBLE,neighborRight,0,CommCart,&req[2]);
    MPI_Isend(sendUp,n,MPI_DOUBLE,neighborUp,1,CommCart,&req[3]);

    //matrix-matrix multiplication computation
    for( j = 0; j < WORK; j++)
      dgemm_(.....);

    MPI_Waitall(4,req,stat);
  }
```

Figure 1. Code Snippet of the Sparse Linear Algebra Benchmark.

Our study is performed using a synthetic benchmark that mimics the behavior of the sparse matrix-matrix multiplication kernel presented in the previous section. Figure 1 depicts the code snippet of this benchmark. The synthetic benchmark allows us to isolate the communication behavior of this kernel from the remaining kernels of CP2K. Therefore, a better understanding of the impact of each strategy is obtained.

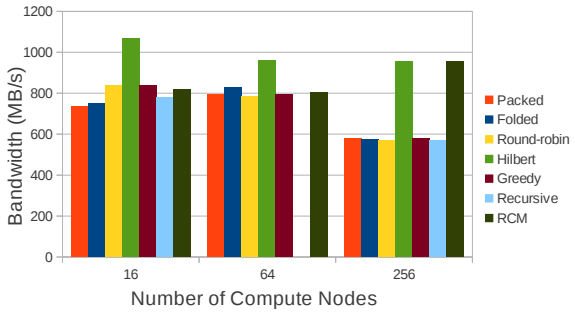

Figure 2. Bandwidth for Different Process Mapping Strategies.

Figure 2 reports the measured bandwidth for the benchmark when the selected process mapping strategies are applied. One can observe that for 64 compute nodes the results are not presented for the recursive strategy. For this number of nodes, we have experienced a technical problem with the library that provides this strategy. Therefore, the results are not presented for this mapping strategy.

Usually, the default mapping strategy on the target machine is packed one. Overall, the others mapping strategies improved the bandwidth for the benchmark when compared to the default one. The round-robin and folded strategies did not significantly improve the bandwidth because they spread adjacent tasks over compute nodes similarly to packed strategy. This results in: (i) an increased distance between the communicating tasks and (ii) overloaded links. The strategies proposed in [3] provided some improvements when compared to the default mapping. However, since they do not exploit the structure of cartesian grids, on average, the obtained mappings did not improve bandwidth. For all number of used compute nodes the Hilbert mapping provided the best bandwidths. Hilbert mapping can provide better bandwidths because (i) it reduces the contention on the links, by exploiting more links and (ii) its mapping matches the communication pattern of the benchmark, by keeping four directions neighbors of a task close to it. Therefore, communication costs are reduced by increasing the available bandwidth.

Since we were interested in Cray XE6, we also had to take into account in our study the bandwidth on each direction (X, Y and Z) of the 3D torus. In order to study the impact of the bandwidth in each direction of the torus, we conducted some experiments using the Hilbert mapping. For given a Hilbert mapping, we placed the MPI processes firstly in one specific direction and then when there is no more nodes on this direction, we start with a different one. From these experiments, we have observed that in the Cray XE6 the directions X and Z provides better bandwidths. Compared to the Y direction, X and Z provided up to 30% better bandwidths for Hilbert mapping.

4. Process Mapping Strategy

It is well known that computing an optimal mapping for a given set of tasks on a parallel machine in polynomial time is not possible. Therefore, to compute an efficient mapping in a reasonable time a heuristic is usually considered. In this section, we introduce a heuristic based process mapping strategy to improve network bandwidth for a sparse linear algebra kernel of CP2K application. It relies on a Hilbert space filling curve and takes into account the communication pattern of the kernel and the interconnection network characteristics.

The reason for using Hilbert space filling curve in our heuristic is twofold: (i) its proprieties matches with the communication pattern of the target kernel, since it keeps close the four neighbors of a task (up, down, right and left) and (ii) it distributes tasks over the computes nodes of the Cray XE6 in such way that network links are better exploited. Another important characteristic of Hilbert curve is that our previous studies have shown that it also fits well some of other important kernels of CP2K application [9].

Both, communication graph and interconnection network are the inputs for our heuristic. They are represented as two dimensional matrices. For the communication graph, we have a $p \times 4$ matrix, where for each process p_i the four tasks ids that it communicates are kept. The communication graph is obtained at CP2K initialization step and

```
Input: Communication Graph C,
       Network Graph N, MPI Grid G,
       Initial Mapping M'

Output: Mapping M

Begin
1   initial_hb ← compute_hop_bytes(M')
2   Hilbert_M ← compute_hilbert(M',G)
3   for each d in D(X,Y,Z)
3     for each i in C
4       node_i ← coordinates(i, Hilbert_M, N)
5       for each j in C(i)
6         node_j ← coordinates(j, Hilbert_M, N)
7       current_hb(d) += compute_hop_bytes(node_i, node_j, msg)
8   direction ← find_smallest_hb(current_hb)
9   if ( current_hb(direction) < initial_hb)
10      M ← ( Hilbert_M, N(direction) )
11  else
      M ← M '
End
```

Figure 3. Algorithm - Heuristic Based Mapping.

saved for all MPI processes. The network is represented as a n matrix, where for each node n_i we have its three coordinates in the network. The network matrix is obtained using Cray tool named *xtdb2proc*.

The heuristic first computes for a given cartesian grid its mapping on the target machine, following the Hilbert space filling curve proprieties. After that, the obtained mapping is used to compute the hop-bytes performance metric [21] for each direction of the 3D torus. The direction that ended with the smallest hop-bytes is the one selected as a start point for the new mapping. However, the mapping is only used to place the MPI processes for the sparse linear algebra kernel if the obtained hop-bytes is better than the initial mapping.

Algorithm 1 presents the pseudo code of our heuristic based process mapping. In line 1, the hop-bytes is computed for the initial mapping. The hop-bytes metric gives to our heuristic an estimation of the contention on the interconnection network. In the next line, the Hilbert mapping is computed and saved in $Hilbert\_M$ data structure. After that, it starts the loop to compute the hop-bytes for the obtained $Hilbert\_M$ in each direction of the Torus (lines 3-7). In the end of this loop, the direction with the smallest hop-bytes is computed and compared to the initial hop-bytes. If the initial mapping has worse hop-bytes, the generated $Hilbert\_M$ is used. However, the processes are mapped taking into account firstly the computed $direction$, than it goes to the second $direction$ with the smallest hop-bytes and finally to the direction with the worst hop-bytes.

Concerning implementation details, our process mapping is applied in the initialization step of CP2K. In this step, a routine is responsible for creating the MPI communicator and loading the simulated system information on all MPI tasks. Our strategy is implemented as a part of this routine. We take as input the communicator created by CP2K and as an output we provide to CP2K a new communicator that has a different ordering for the MPI processes. This communicator is latter passed as a parameter for the sparse linear algebra kernel. Therefore, the whole application uses our mapping. This solution avoids data migration, which can be very costly, between the sparse linear algebra kernel and the other kernels of CP2K.

5. Results

In this section, we analyze the performance of our heuristic based process mapping. The experimental environment presented in section 3 is also used for the results presented in this section.

For the results presented bellow, we used the SVN version 12412 of CP2K [1]. Although CP2K has a hybrid implementation with MPI processes and threads, in our experiments we built the MPI only version. This version allows us to better observe the impact of process mapping on the communication of the sparse linear algebra kernel (DBCSR). Examples of how to compile this version of CP2K can be found in the CP2K repository. For all compilations we used flags for code optimization (i.e. -O3). Concerning the simulations, we have used an input file that computes energy for a given configuration of water molecules (2048 molecules). In this simulations, DBCSR represents up to 50% of the total execution time of the CP2K run. For our experiments, we performed strong scaling tests and the best of at least three independent runs was used to evaluate the heuristic.

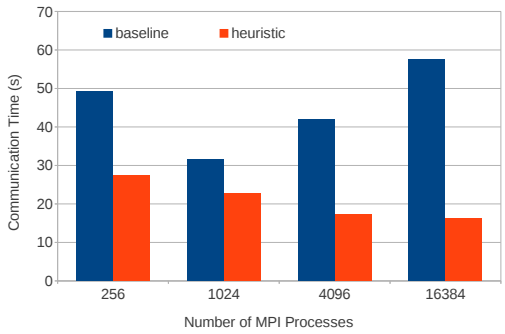

Figure 4. Communication Time in seconds for the Sparse Linear Algebra Kernel.

We use as performance metric to evaluate our strategy the communication time spent in the sparse linear algebra kernel. This metric was selected because the communication of the selected kernel can be up to 80% of the total time spent on this kernel. The time presented in Figure 4 comprehends all communication performed within the kernel. We present the communication time in seconds for the baseline version of CP2K and the version with the proposed mapping. The baseline version used the packed process mapping strategy to place the MPI processes over the allocated compute nodes.

According to Figure 4, our process mapping improves the communication time for all number of MPI processes. Moreover, as the number of processes increases, the communication time of the baseline version also increases. Contrary to this, the communication time with the proposed heuristic to map process decreases. Therefore, our heuristic allows better scalability of the sparse linear algebra kernel for a increased number of processes. This stems from the fact that our heuristic allows better usage of the network links and keeps communicating tasks close. Since our heuristic takes into account the performance of each direction of the 3D torus, it can improve even more the mapping obtained with the Hilbert curve.

6. Conclusions

In this work, we studied the impact of different process mapping strategies on a sparse linear algebra kernel of CP2K application. Based on the results of this study, we proposed a heuristic to perform process mapping for CP2K.

The heuristic relies on Hilbert space filling curve, but also takes into account the machine interconnection network topology and the communication patterns of the kernel. Together these parameters are used in a communication cost function that allows our heuristic increase bandwidth perceived by the MPI processes.

Our experimental results showed that the proposed heuristic reduces the communication overhead for the kernel and consequently, improves the CP2K performance. Overall this mapping provided 40% more bandwidth on a CRAY XE6, improving the kernel performance. The proposed heuristic balances intra and inter node communication, providing a better usage of the network links.

Our future works include the study of process mapping to improve the performance of other kernels of CP2K application, the proposal of new heuristics for mapping processes and the proposal of a process mapping algorithm that exploits the different communication patterns presented on CP2K kernels.

Acknowledgements

JV acknowledges financial support by the European Union FP7 in the form of an ERC Starting Grant under contract no. 277910. Calculations were enabled by 2011 INCITE awards on the CRAY XK6 using resources of the National Center for Computational Sciences at Oak Ridge National Laboratory (ORNL), which is supported by the Office of Science of the U.S. DOE under Contract No. DE-AC05-00OR22725, and by the Swiss National Supercomputer Centre (CSCS) under project ID s238. The research leading to these results has received funding from the Swiss University Conference through the High Performance and High Productivity Computing (HP2C) Programme. We would like to thank Neil Stringfellow (CSCS) and Roberto Ansaloni (Cray) for the support with Cray systems and environment. Also, we would like to thank Peter Messmer (Nvidia) for the synthetic benchmark of the sparse linear algebra kernel.

References

[1] CP2K, "Open Source Molecular Dynamics," http://cp2k.org/, 2012.

[2] A. Bhatele, T. Gamblin, S. H. Langer, P.-T. Bremer, E. W. Draeger, B. Hamann, K. E. Isaacs, A. G. Landge, J. A. Levine, V. Pascucci, M. Schulz, and C. H. Still, "Mapping applications with collectives over sub-communicators on torus networks," in *Proceedings of the ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '12. IEEE Computer Society, Nov. 2012, lLNL-CONF-556491. [Online]. Available: http://dl.acm.org/citation.cfm?id=2388996.2389128

[3] T. Hoefler and M. Snir, "Generic topology mapping strategies for large-scale parallel architectures," in *Proceedings of the international conference on Supercomputing*, ser. ICS '11. New York, NY, USA: ACM, 2011, pp. 75–84.

[4] J. Vandevondele, M. Krack, F. Mohamed, M. Parrinello, T. Chassaing, and J. Hutter, "Quickstep: Fast and accurate density functional calculations using a mixed gaussian and plane waves approach," *Computer Physics Communications*, vol. 167, no. 2, pp. 103–128, 2005.

[5] B. G. Lippert, M. Parrinello, and J. Hutter, "A hybrid gaussian and plane wave density functional scheme," *Molecular Physics*, vol. 92, no. 3, pp. 477–488, 1997.

[6] M. Guidon, J. Hutter, and J. VandeVondele, "Robust Periodic Hartree-Fock Exchange for Large-Scale Simulations Using Gaussian Basis Sets," *Journal of Chemical Theory and Computation*, vol. 5, no. 11, pp. 3010–3021, 2009.

[7] M. Del Ben, J. Hutter, and J. VandeVondele, "Second-Order Møller-Plesset Perturbation Theory in the Condensed Phase: An Efficient and Massively Parallel Gaussian and Plane Waves Approach," *Journal of Chemical Theory and Computation (ASAP)*, vol. 0, no. 0, 2012.

[8] J. VandeVondele, U. Borštnik, and J. Hutter, "Linear scaling self-consistent field calculations with millions of atoms in the condensed phase," *Journal of Chemical Theory and Computation*, vol. 8, no. 10, pp. 3565–3573, 2012.

[9] C. P. Ribeiro and J. Hutter, "Improving Atomistic and Molecular Simulations Performance on Parallel Machines with a Hierarchical Mapping Strategy," https://sites.google.com/site/pousachristiane/publications, 2012.

[10] G. Mercier and J. Clet-Ortega, "Towards an efficient process placement policy for MPI applications in multicore environments," in *Europvm/mpi 2009*, ser. Lecture Notes in Computer Science, M. Ropo and al., Eds., vol. 5759. Espoo, Finland: Springer-Verlag, 2009, pp. pp 104–115. [Online]. Available: http://hal.inria.fr/inria-00392581

[11] G. Mercier and E. Jeannot, "Improving MPI Applications Performance on Multicore Clusters with Rank Reordering," in *EuroMPI*, Springer, Ed., vol. 6960, Santorini, Italy, Sep. 2011, pp. 39–49. [Online]. Available: http://hal.inria.fr/hal-00643151

[12] E. R. Rodrigues, P. O. A. Navaux, J. Panetta, C. L. Mendes, and L. V. Kalé, "Optimizing an mpi weather forecasting model via processor virtualization," in *HiPC*, 2010, pp. 1–10.

[13] E. R. Rodrigues, P. O. A. Navaux, J. Panetta, A. Fazenda, C. L. Mendes, and L. V. Kale, "A Comparative Analysis of Load Balancing Algorithms Applied to a Weather Forecast Model," in *Proceedings of 22nd International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*, Itaipava, Brazil, 2010.

[14] M. J. Rashti, J. Green, P. Balaji, A. Afsahi, and W. Gropp, "Multi-core and network aware mpi topology functions," in *Proceedings of the 18th European MPI Users' Group conference on Recent advances in the message passing interface*, ser. EuroMPI'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 50–60.

[15] A. Bhatele and L. V. Kale, "Benefits of topology aware mapping for mesh interconnects," *Parallel Processing Letters*, vol. 18, no. 04, pp. 549–566, Dec. 2008. [Online]. Available: http://www.worldscientific.com/doi/abs/10.1142/S0129626408003569

[16] L. L. Pilla, C. P. Ribeiro, P. Coucheney, F. Broquedis, B. Gaujal, P. O. Navaux, and J.-F. Méhaut, "A topology-aware load balancing algorithm for clustered hierarchical multi-core machines," *Future Generation Computer Systems*, no. 0, pp. –, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X13001374

[17] D. Hilbert, "Ueber die stetige Abbildung einer Line auf ein Flächenstück," *Mathematische Annalen*, vol. 38, no. 3, pp. 459–460, 1891.

[18] L. E. Cannon, "A cellular computer to implement the kalman filter algorithm," Ph.D. dissertation, Montana State University, 1969.

[19] X. Liu and G. Shrack, "Encoding and decoding the hilbert order," *Software: Practice and Experience*, vol. 26, no. 12, pp. 1335–1346, 1996.

[20] NERSC, "Rank Reordering," http://www.nersc.gov/users/computational-systems/hopper/performance-and-optimization/reordering-mpi-ranks/, 2013.

[21] A. Bhatele, G. Gupta, L. V. Kale, and I.-H. Chung, "Automated Mapping of Regular Communication Graphs on Mesh Interconnects," in *Proceedings of International Conference on High Performance Computing (HiPC)*, 2010.