Topology-Aware Mapping Techniques for Heterogeneous HPC Systems: A Systematic Survey

Saad B. Alotaibi[1], Dr. Fathy alboraei[2]
Faculty of Computing and Information Technology
King Abdulaziz University
Riyadh, Saudi Arabia

***Abstract*—At the present time, the modern platforms of high-performance computing (HPC) consists of heterogeneous computing devices which are connected through complex hierarchical networks. Moreover, it is moving towards the Exascale era and which makes the number of nodes to increase as well as the number of cores within a node to increase. As a consequence, the communication costs and the data movement are increasing. Given that, the efficient topology-aware process mapping has become vital to efficiently optimize the data locality management in order to improve the system performance and energy consumption. It will also decrease the communication cost of the processes by matching the application virtual topology (exploited by the system for assigning the processes to the physical processor) to the target underlying hardware architecture called physical topology. Additionally, improving the locality problem which is one of the most challenging issues faced by the current parallel applications. In this survey paper, we have studied various topology-aware mapping techniques and algorithms.**

Keywords—Virtual topology; physical topology; topology-aware mapping; parallel applications; communication pattern

I. Introduction

Good topology-aware process mapping has an acute role in improving the performance of the parallel applications in high-performance computing (HPC) as well as the energy consumption, considering the increasing hierarchical, heterogeneous and complex nature of the current and future high-performance computing (HPC) platforms. The "Heterogeneous" term refers to non-symmetry in a few or several system aspects. The heterogeneity appears in several parts such as; networks and can emerge from hardware heterogeneity (CPUs, GPUs, FPGAs), software heterogeneity (Compilers, operating system, libraries, etc.) and the network topology complexity [1]. For that matter, the applications of high-performance computing need to adapt the heterogeneity platforms to optimum execution.

As an illustration, the topology-aware process mapping is a way of carrying out a particular task to enhance parallel application execution by decreasing the communication cost of processes by matching the application of virtual topology (exploited by the system for assigning the processes to the physical processor) to the target underlying hardware architecture called physical topology. One of the advantages of topology-aware mapping is the decreased cost of communication, by matching the application data to the processors that are physically close one to the other.

In order to do a topology-aware process mapping, it is necessary to choose the parallel programming models that help in this matter. To put it another way, the parallel programming model has a valuable help in application execution, because some of the parallel programming models have a mechanism that helps the application to exploit the underlying hardware to improve communication and the locality. Moreover, it will be helpful for virtual topology management to reorganize the processes according to the target underlying hardware architecture. Therefore, the most important parallel programming model is the Message Passing Interface (MPI) which is the standard model of the parallel programming models.

As discussed above, we propose the main three steps to make an efficient topology-aware process mapping, as follows:

1) Develop a virtual topology by gathering the application communication pattern.

2) Develop a physical topology by modeling the underlying hardware architecture.

3) Develop a clever algorithm or technique by matching the numbers of computing elements and the process ranks of the application.

The following architecture explains the previous steps "Fig. 1".

The mapping of topologies is of two types: static and dynamic. In the static approach, the mapping can be done prior to the execution. As for the second approach which is dynamic mapping, it happens at runtime (remap the processes to another processor or core during the runtime) [2].

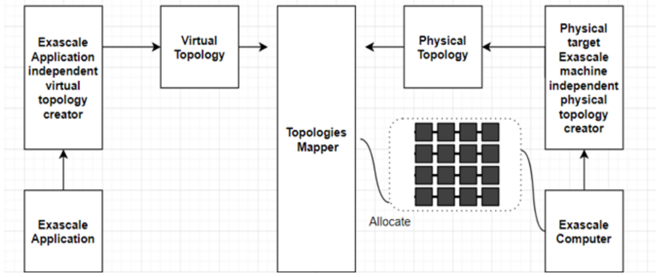

Exascale Application independent virtual topology creator
Virtual Topology
Topologies Mapper
Physical Topology
Physical target Exascale machine independent physical topology creator
Allocate
Exascale Application
Exascale Computer

Fig. 1. High-Level Architecture of Topology-Aware Process Mapping.

This paper is organized as follows: section 2 comprises the definitions of the topologies with examples, section 3 includes the previous related work, whilst section 4 discusses the definition of the problem and the section 5 concludes this paper.

II. Topologies Definitions

A. Virtual Topology

The term virtual topology means the dependence among the software processing entitles. These dependencies may be defined as the data that is exchanged between the processes or an access to the memory by the application threads. In other words, the virtual topology refers to the application communication patterns [2]. Furthermore, the virtual topology has several types such as graph topologies and Cartesian topologies. The example of the virtual topology is shown in "Fig. 2"

0 (0,0)	1 (0,1)	2 (0,2)	3 (0,3)
4 (1,0)	5 (1,1)	6 (1,2)	7 (1,3)
8 (2,0)	9 (2,1)	10 (2,2)	11 (2,3)

Fig. 2. Virtual Topology Example, (0.0) is a Coordinate and 0 is a Rank Id.

B. Pysical Topology

Nowadays, the modern machines are increasingly complex, include multiple processors, multi-core processors (socket = package), simultaneous multithreading, NUMA nodes, shared caches, and multiple GPUs, NICs, etc. Similarly, the underlying hardware known as physical topology includes the NUMA memory nodes, cores, simultaneous multithreading, sockets and shared caches [3]. Correspondingly, the application needs to understand the target underlying hardware for optimum execution. The example of the underlying hardware architecture is shown in "Fig. 3".

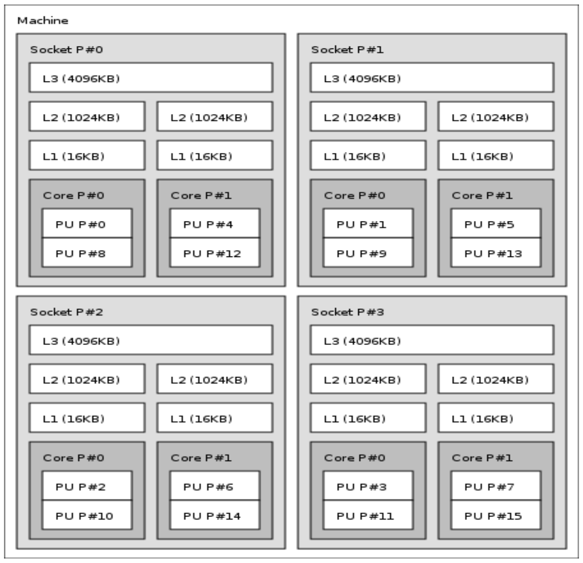
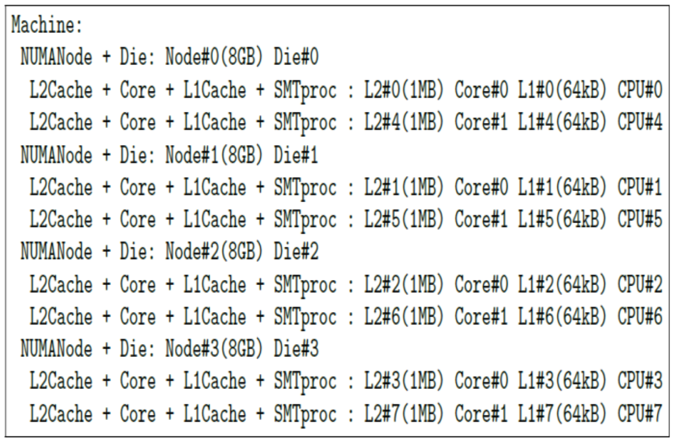


Fig. 3. High-Level Architecture of the Target Machine.



```
Machine:
 NUMANode + Die: Node#0(8GB) Die#0
  L2Cache + Core + L1Cache + SMTproc : L2#0(1MB) Core#0 L1#0(64kB) CPU#0
  L2Cache + Core + L1Cache + SMTproc : L2#4(1MB) Core#1 L1#4(64kB) CPU#4
 NUMANode + Die: Node#1(8GB) Die#1
  L2Cache + Core + L1Cache + SMTproc : L2#1(1MB) Core#0 L1#1(64kB) CPU#1
  L2Cache + Core + L1Cache + SMTproc : L2#5(1MB) Core#1 L1#5(64kB) CPU#5
 NUMANode + Die: Node#2(8GB) Die#2
  L2Cache + Core + L1Cache + SMTproc : L2#2(1MB) Core#0 L1#2(64kB) CPU#2
  L2Cache + Core + L1Cache + SMTproc : L2#6(1MB) Core#1 L1#6(64kB) CPU#6
 NUMANode + Die: Node#3(8GB) Die#3
  L2Cache + Core + L1Cache + SMTproc : L2#3(1MB) Core#0 L1#3(64kB) CPU#3
  L2Cache + Core + L1Cache + SMTproc : L2#7(1MB) Core#1 L1#7(64kB) CPU#7
```

Fig. 4. Hardware Topology Information.

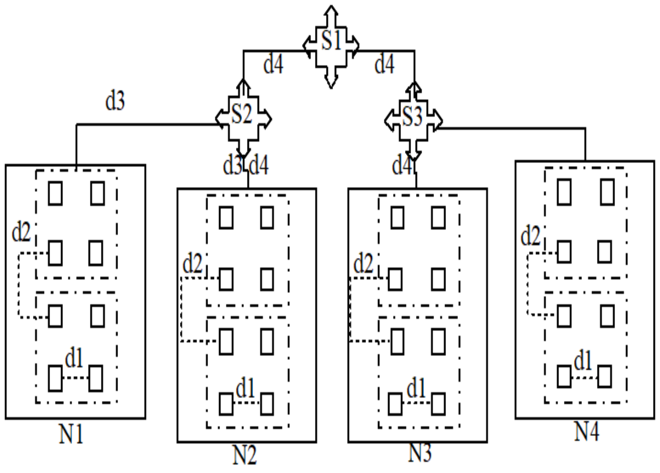


Fig. 5. Physical Topology Distance, d = distince, N = node and s = switch.

Likewise, we can gather the information on the target machine using the topology discovery mechanism as shown in "Fig. 4"

Given that, the physical topology is the hardware affinity known as physical topology distance [4], shown in "Fig. 5".

C. Parallel Programming Model

The main parallel programming models for high-performance computing are OpenMP (which are used for shared memory architecture) and MPI (which are used for distributed memory systems). At the present time, we have several parallel programming models such as OpenCL (Open Computing Language –used for the heterogeneous parallel computing), OpenCV (which has the power to concentrate on the real-time applications) and OpenACC (which is a programming standard and was intended to simplify parallel programming of heterogeneous CPU/GPU systems) [5] [6].

Additionally, in the high-performance computing we can make hybrid parallel programming models to do a specific task that takes the advantages of the shared and distributed memory. "table-1" shows the parallel programming models as well as the systems that implement them [6].

TABLE I. PARALLEL PROGRAMMING MODELS AND THEIR IMPLEMENTED SYSTEMS

Programming Model	**Example Programming Systems**
Shared memory	
Dynamic scheduling, nested bulk synchronous	OpenMP, TBB, Cilk++
Dynamic scheduling, the general synchronization	pthreads, OpenMP, TBB, Cilk++
Distributed memory	
Bulk-synchronous	BSP, MPI with collectives/barriers, X10 with clocks
Static scheduling, two-sided communication	MPI point-to-point
Static scheduling, one-sided communication	MPI RDMA, SHMEM, UPC, Fortran
Hybrid scheduling (static across nodes, dynamic within nodes)	MPI+OpenMP, DPLASMA
The local view of data and control	MPI, Fortran
The local view of control, global view of data	UPC, Global Arrays
Global view of data and control	OpenMP, Chapel
CoProcessor/Accelerator separate memory	OpenCL, OpenACC, CUDA
Domain-specific languages and libraries	PETSc, Liszt, TCE

D. *Parallel Computing Systems*

The modern engineering and science applications require a massive amount of computing because it deals with very complex problems. In order to address these complex problems, we need powerful computing systems such as parallel computing. As an illustration, parallel computing is one of the most powerful computations that can make numerous calculations and execute the processes, simultaneously. To put it differently, large problems can often be divided into smaller ones, and then solved at the same time [7].

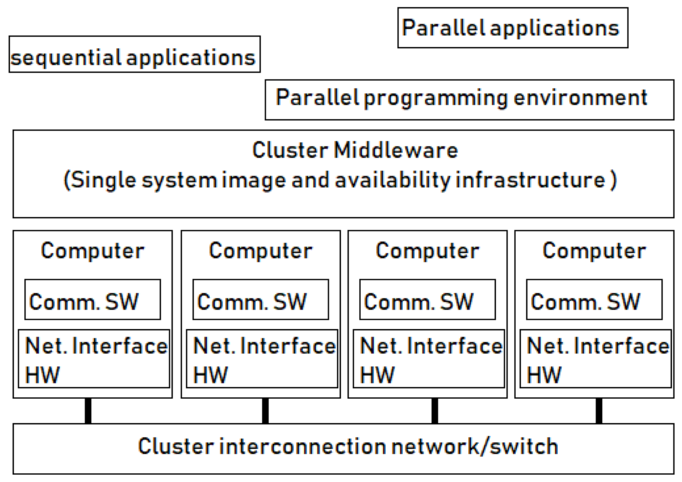


Fig. 6. High-Level Architecture of Parallel Computing.

On the negative side and in our case, the programmers face many challenges with the parallel systems such as the complex hierarchy of the hardware, methods to minimize the memory usage by the applications, less communication, and data locality.

The high-level architecture of parallel computing is shown in “Fig. 6”.

III. BACKGROUND AND RELATED WORK

The modern platforms of high-performance computing (HPC) consists of heterogeneous computing devices which are connected through complex hierarchical networks. In order to efficiently execute the data-parallel Exascale applications on that platforms, we need to balance a load of the processors, as well as minimize the communications cost. To achieve that we need to separate the data among processors whilst considering their speed. The second can be optimized by decreasing the communications volume by mapping the application data to the processors that are physically close to one another. Moreover, the topology information will be used as the guide to improve the communications in the hierarchical-heterogeneous platforms.

Nowadays, as we are moving towards the Exascale, the topology-aware process mapping is becoming an important approach to improve the performance and reduce the power consumption of Exascale applications. Accordingly, most researchers in this area have proposed many techniques and approaches for finding the best and efficient topology-aware process mapping. As can be seen, every researcher focusses on different aspects of how to build the efficient mapping of the process-to-processor. It is also noticed that most researchers come up with their own mapping approach and try to make efficient topology-aware process mapping.

Briefly, we have summarized all the previously done studies on the topology-aware process mapping problem. To begin with, Emmanuel et al. [7] have proposed techniques to deal with NUMA node clusters for reducing the communications costs. The proposed techniques can gather the information of the application communication pattern and the details of the target machine hardware, and then compute the relevant ranks of reordering application process. Eventually, the new ranks are used for reducing the application communication costs. As a matter of fact, those techniques are based on the TreeMatch algorithm. This algorithm deals with resource binding technique such as computing unit numbers and the rank reordering technique as the new MPI ranks. However, the algorithm design is as follows:

```
  Input: $T$// The topology tree
  Input: $m$ // The communication matrix
  Input: $D$ // The depth of the tree
1 groups[1..$D-1$]=$\emptyset$ // How nodes are grouped on each level
2 foreach depth← $D-1..1$ do // We start from the leaves
3     $p$ ← order of $m$
      // Extend the communication matrix if necessary
4     if $p$ mod arity($T, depth-1$) $\neq 0$ then
5         $m$ ←ExtendComMatrix($T$,$m$,depth)
6     groups[depth]←GroupProcesses($T$,$m$,depth)// Group
      processes by communication affinity
7     $m$ ←AggregateComMatrix($m$,groups[depth])      // Aggregate
      communication of the group of processes
8 MapGroups($T$,groups) // Process the groups to built the
  mapping
```

The work by Guillaume et al. [8] has modified the function of the MPICH2 implementation of the MPI_Dist_graph_create for reordering the process ranks of MPI. The objective is to create a map between the hardware topology and the application communication pattern. Nonetheless, this modification is achieved through two main but different methods, the core binding, and the rank reordering.

Balaji et al. [9] say that the varying mapping of the application on the large-scale systems is an important factor that affects the overall performance. Furthermore, the authors have highlighted the mapping impact on the application performance of "the IBM Blue Gene/Q systems" with the network topology of the 5D torus.

Francois et al. [3] have observed that the number of cores, memory nodes, and shared caches are increasing, thus, making the hardware topology very complex. Moreover, the high-performance computing applications need to be careful while adapting their placement to the target underlying hardware. For that matter, they proposed the hardware locality (HWLOC) tool that gathers the information of the physical topology including caches, processors, and memory nodes which makes it visible to the application as well as the runtime systems. This tool is used by the most important parallel programming models such as OpenMP & MPI.

Joshua et al. [10] have proposed a Locality-Aware Mapping algorithm to distribute the parallel application processes across processing resources in the high-performance computing system. This algorithm is capable of dealing with both, heterogeneous and homogeneous hardware systems. In the final analysis, they implemented it on the OpenMPI.

Bhatele et al. [11] have proposed various heuristics that are based on the hop-bytes metrics for mapping the graphs of irregular communication to the mesh topologies. Their heuristics try to place the communicating processes close to one another.

Mercier et al. [12] built the topology-aware mapping, based on the Scotch library. Generally speaking, they used the virtual topology (The application communication pattern) and the physical topology as a complete weighted graph.

Rashti et al. [13] have extracted the network topologies and intra-node using the InfiniBand tools and HWLOC library respectively. To develop the undirected graph with edges that represent the performance of the communication between cores depending on their distances. Then, this mapping technique is executed by the Scotch library.

Ito et al. [14] have proposed a similar mapping technique but using the existing bandwidth between the nodes measured at the time of execution for assigning the edge weights in the graph of the physical topology. Again, the method of this mapping technique was implemented by the Scotch library.

Chung et al. [15] proposed an efficient technique based on the hierarchical mapping which partitions the physical topology graphs and the process into numerous super nodes. Also, the very first mapping assigns process topology graph supernodes to the equivalent peers in the graph of the physical topology.

Cyril Bordage et al. [16] proposed a Netloc tool for collecting the physical topology that is integrated with a Scotch practitioner for computing the topology-aware MPI process placement. However, their experiments were based on the fat-tree machine.

K. B. Manwade et al. [17] proposed a novel technique known as a “ClustMap” for mapping the application and system topologies.

Abhinav Bhatele et al. [18] constructed an automatic mapping framework that can help the developer to automate the application communication pattern and physical topology of the parallel application. In addition, their framework can analyze the process topology to find regular patterns and then identify the communication graphs dimensions for the application.

Jingjin Wu et al. [19] proposed a strategy for the mapping of the hierarchical task that implements inter and intra node mapping. They considered supercomputers with torus network and fat-tree topologies, additionally providing two mapping algorithms. The first can deal with both inter-node and intra-node mapping. The second can partition the nodes of the computation regarding its affinity.

Torsten Hoefler et al. [2] demonstrate a new heuristic based on the graph similarity and shows its utility with the virtual topology on real physical topologies. In other words, their mapping strategies support the heterogeneous networks and try to reduce the congestion on fat-tree, torus, and the PERCS network topologies for irregular communication patterns.

Subramoni et al. [20] proposed efficient topology mapping on the InfiniBand networks for detecting the InfiniBand network topology and that can be done using the neighbor joining algorithm.

Deveci et al. [21] considered machines with the allocation of the sparse node and then applied a geometric partitioning algorithm to processors and tasks to find the appropriate mapping.

Agarwal et al. [22] proposed a greedy heuristic through the estimation functions that are used to evaluate the mapping decisions effects.

Mohammad et al. [23] used the network/node architecture and graph embedding modules for mapping the application communication topology onto the multi-core clusters physical topology with multi-level networks. As the result, they have got the great improvement in the application communication performance as well as the execution time. In the final analysis, this result is obtained by Micro-benchmark.

IV. Discussions

Aggregated power for computing is recognized as the most recent phenomenon for data-intensive tasks in the 21st century. High-performance computing is able to handle simulation modeling as well as support standard workstations. Through carrying out several computing operations within a reasonable amount of time, high-performance computing is able to counter performance challenges related to limited data sources. This is achieved using high-end specialized hardware that incorporates several units which gather computing power. Additionally, the units use the concept of parallelization to distribute data and operations across the various subsequent levels. This is due to a large amount of data movement and lack of application placement patterns onto the elements of the hardware processing. In short, when we study the process placement, we must focus on the system hierarchy of the high-performance computing (HPC) because the system hierarchy increases more and more, and the nodes become multi-levels of memory (non-volatile memory, faster but smaller MCDRAM for KNL, standard DRAM, etc.) and composed of multicore processors. Moreover, the network that connects these nodes has very complex topology [24]. Thus, it is concluded that the process placement is not an easy task in case of very effective process placement. Additionally, the topology mapping or process placement has a critical role on the parallel application performance and we need to map these processes onto processors carefully. Therefore, the goal of every successful mapping algorithm relies on how to reduce the communication costs by carefully mapping the processes that are closest to each other and require most communication. Algorithmically, the mapping process has two kinds; the first one is how the machine computes the messages communication costs and the second one is how the application can describe the computing elements affinity. Because the affinity of the computing entities is very important in case of mapping the processes on the processors which are close to each other.

Lastly, it was witnessed that the topology-aware process mapping is an active research filed. The both, application communication pattern (virtual topology) and the underlying hardware details (physical topology) are not difficult to extract, the main contribution is the topology process mapping algorithm. In fact, we advise the interested researchers to use HWLOC tool to extract the physical topology (underlying hardware details) [3] and use any graph partitioning or MPI ranks reordering for virtual topology [7].

V. Conclusion and Future Work

At the present time, we observe that the number of nodes are increasing, as well as the number of cores within a node are increasing. As a result, the high-performance computing systems are becoming very complex, which leads to the increase in the heterogeneity levels at the communication channels, such as inter-node and intra-node communications. The diversity in the performance through different communication channels in the high-performance computing systems make it significant to think carefully about information of topology at higher levels. The knowledge of topology facilitates to fulfill the effective exploitation of underlying communication channels which leads to an increase in communication performance at the application level. Therefore, the topology-aware process mapping is a necessary approach for improving the performance of communication in high-performance computing systems. In addition, the topology-aware process mapping helps in reducing the lot of congestion that happens in the system hierarchy on several levels. As we know the congestion has its effect on communication performance. Ultimately, based on the previous description we are aiming and focusing on how to improve the HPC systems performance without adding any extra overhead and/or the power consumption. We will focus on the mapping between the nodes (internode) and the mapping within a node (intra-node) for achieving the efficient performance as much as we can. Given that, we have proposed an efficient new technique based on hybrid parallel programming model as a tri-model for mapping virtual topology onto physical topology to optimize the data locality management for increasing the performance and reducing the power consumption in the HPC systems. This approach can optimize the mapping of inter-node by taking into account the communication pattern of the inter-node and the network topology. Moreover, it will optimize the intra-node mapping whereby the node physical topology and the corresponding communication pattern of intra-node. According to the mapping process, we will consider the load balancing within nodes as the nodes will be heterogeneous.

References

[1] Tania Malik, (2016) Topology-aware Optimization of Communication Cost of Parallel Applications in Heterogeneous HPC Systems, PhD. University College Dublin

[2] Torsten Hoefler, Emmanuel Jeannot, Guillaume Mercier. An Overview of Process Mapping Techniques and Algorithms in High-Performance Computing. Emmanuel Jeannot and Julius Zilinskas. High Performance Computing on Complex Environments, Wiley, pp.75-94, 2014.

[3] F. Broquedis et al., "hwloc: A Generic Framework for Managing Hardware Affinities in HPC Applications," 2010 18th Euromicro Conference on Parallel, Distributed and Network-based Processing, Pisa, 2010, pp. 180-186.

[4] M. J. Rashti, J. Green, P. Balaji, A. Afsahi, and W. Gropp. Multi-core and network aware MPI topology functions. In Proc. European MPI Users’ Group Meeting (EuroMPI), pages 50–60, 2011.

[5] M.D. Chougule, P. H. Gutte, “Parallel Programming Models: A Systematic Survey” in an International journal IJCSIT of July issue 2014.

[6] W. Gropp, M. Snir, “Programming for Exascale Computers”, Computing in Science & Engineering, 2013, 15(6), P.27-35

[7] Jeannot, E., Mercier, G. and Tessier, F., 2014. Process placement in multicore clusters: Algorithmic issues and practical techniques. Parallel and Distributed Systems, IEEE Transactions on, 25(4), pp.993-1002.

[8] G. Mercier and E. Jeannot, Improving MPI Applications Performance on Multicore Clusters with Rank Reordering, EuroMPI, p.3949, 2011.

[9] P. Balaji, R. Gupta, A. Vishnu, and P. Beckman. “Mapping communication layouts to network hardware characteristics on massive-scale blue gene systems”. Computer Science-Research and Development, 26(3-4):247–256, 2011.

[10] Joshua Hursey , Jeffrey M. Squyres , Terry Dontje, Locality-Aware Parallel Process Mapping for Multi-core HPC Systems, Proceedings of the 2011 IEEE International Conference on Cluster Computing, p.527-531, September 26-30, 2011

[11] A. Bhatel´e and L. V. Kal´e. Heuristic-based techniques for mapping irregular communication graphs to mesh topologies. In International Conference on High Performance Computing and Communications (HPCC),pages.765–771,2011.

[12] G. Mercier and J. Clet-Ortega. Towards an efficient process placement policy for MPI applications in multicore environments. In Recent Advances in Parallel Virtual Machine and Message Passing Interface (EuroPVM/MPI), pages 104–115. 2009.

[13] M. J. Rashti, J. Green, P. Balaji, A. Afsahi, and W. Gropp. Multi-core and network aware MPI topology functions. In Proc. European MPI Users’ Group Meeting (EuroMPI), pages 50–60, 2011.

[14] S. Ito, K. Goto, and K. Ono. Automatically optimized core mapping to subdomains of domain decomposition method on multicore parallel environments. Computers & Fluids, 80(0):88–93, 2013.

[15] I.-H. Chung, C.-R. Lee, J. Zhou, and Y.-C. Chung, “Hierarchical mapping for HPC applications,” in Proc Workshop Large-Scale Parallel Processing, 2011, pp. 1810–1818.

[16] Cyril Bordage, Clément Foyer, Brice Goglin. Netloc: a Tool for Topology-Aware Process Mapping. Euro-Par 2017: Parallel Processing Workshops, Aug 2017, Santiago de Compostela, Spain.

[17] K. B. Manwade and D. B. Kulkarni, “ClustMap: A Topology-Aware MPI Process Placement Algorithm for Multi-core Clusters”, in Intelligent Computing and Information and Communication, Jan 2018, pp. 67-76

[18] A. Bhatelé, G. R. Gupta, L. V. Kalé and I. H. Chung, "Automated mapping of regular communication graphs on mesh interconnects," 2010 International Conference on High Performance Computing, Dona Paula, 2010, pp. 1-10.

[19] Wu, Jingjin and Xiong, Xuanxing and Lan, Zhiling “Hierarchical Task Mapping for Parallel Applications on Supercomputers”, in J. Supercomput, 2015, pp. 1776-1802

[20] Subramoni H, Potluri S, Kandalla K, Barth B, Vienne J, Keasler J, Tomko K, Schulz K, Moody A, Panda D (2012) Design of a scalable infiniband topology service to enable network-topology-aware placement of processes. In: Proceedings of international conference on high performance computing, networking, storage and analysis, pp 1–12.

[21] M. Deveci, K. Kaya, B. Uc¸ar, and U. V. C¸ ataly ¨ urek, “Fast and high ¨ quality topology-aware task mapping,” in 2015 IEEE Intl. Parallel Distrib. Proc. Symp. (IPDPS), 2015, pp. 197–206.

[22] Agarwal T, Sharma A, Laxmikant A, Kale LV (2006) Topology-aware task mapping for reducing communication contention on large parallel machines. In: Proceedings of IEEE international symposium on parallel and distributed processing (IPDPS)

[23] Mehmet D. et al., “Exploiting Geometric Partitioning in Task Mapping for Parallel Computers”, Proceedings of the 2014 IEEE 28th International Parallel and Distributed Processing Symposium, p.27-36, May 19-23, 2014