Static Process Mapping Heuristics Evaluation for MPI Processes in Multi-core Clusters

Manuela K. Ferreira, Vicente S. Cruz, Philippe O. A. Navaux
Instituto de Informática Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brazil
{mkferreira, vscruz, navaux}@inf.ufrgs.br

Abstract

An important factor that must be considered to achieve high performance on parallel applications is the mapping of processes on cores. However, since this is an NP-hard problem, it requires different mapping heuristics that depend on the application and the hardware on which it will be mapped. This work compares two static process mapping heuristics, Maximum Weighted Perfect Matching (MWPM) and Dual Recursive Bipartitioning (DRB), with the best mapping found by Exhaustive Search (ES) in homogeneous multi-core clusters using MPI. The objective is to compare the performance improvement of MWPM with the already established DRB. The analysis of the NAS benchmarks running with 16 processes presents very similar performance improvements for both heuristics, with an average improvement of 13.79% for MWPM and 14.07% for DRB.

1. Introduction

Multi-core processors can be found in personal computers and high performance computers (HPC). Four to eight cores in a single chip are common today, and the trend is that we will have more cores per chip since Moore's law is still applying [2]. Consequently, old methods for developing and executing programs do not make a reasonable use of the resources of new architectures, *e. g.* leaving some cores running in an idle state. Hence, new methods are required to make an effective use of these resources.

When we place the processes of a parallel application on current multi-core environments, we have an instance of the process mapping problem [3], that is deciding in which processing unit each parallel process will execute with the goal to achieve the best performance. It is an NP-hard problem and consequently does not have a generic solution in a polynomial time, so we can consider the software details and hardware platform characteristics on which it will execute to find a specific solution.

The communication between processes has a significant impact on the performance of parallel applications. Hence it must be take into consideration in the development of process mapping heuristics to decrease the communication time and increase performance [6] [12]. Processes which have the highest amount of communication between each other should be placed on processors so that the communication cost is minimal.

The MPICH communication method, since version 2-1.3, is *nemesis* [5] that combines the already used socket communication method with the shared memory communication method. Moreover, it takes into account if the processes are sharing memory to decide which communication method to use. So, when running MPICH processes in a multi-core cluster, the faster shared memory communication method is used when processes exchange data within a chip – *intrachip* communication – or within a node – *intranode* communication. The socket communication method is only used when a process sends a message to another process located on a different node in the cluster (*internode* communication).

This paper evaluates two static process mapping heuristics for multi-core clusters, using the MPI version of the NAS Parallel Benchmark as the workload [1]. The first heuristic is the *Maximum Weight Perfect Matching* (MWPM) algorithm [9], which uses the communication pattern of the application to create a mapping using a variant of the Edmonds-Karp algorithm. The second heuristic uses the *Dual Recursive Bipartitioning* (DRB) algorithm, creating the mapping by applying a divide and conquer method [10]. Both heuristics are compared to the Exhaustive Search (ES).

The organization of this paper is the following. Section 2 presents the two heuristics that are compared on this work, MWPV and DRB, and the ES. The heuristics performance results are presented and analyzed in Section 3. On Section 4, we show some related works that uses static process mapping. Finally, Section 5 is dedicated for conclusion and future works.

2. Exhaustive Search and Static Process Mapping Heuristics

To describe the heuristics, we need do introduce some definitions. Let $A(P, B)$ be the communication graph of the application, where P is the set of processes that represents the vertices of A. Let B be the set of edges so that $B(p_x, p_y)$, where p_x, $p_y \in P$, represents the amount of communication exchanged between the processes p_x and p_y. In general, the three algorithms presented offer a method of placing a pair of processes (p_x, p_y) on cores that have a shared resource. Thus it is expected that the communication latency is minimized. in this work, the shared resource is a level of memory hierarchy.

To get the amount of messages transfered between processes and to create the communication pattern it is necessary to execute the application that will mapped. The communication pattern is stored on graph A, represented as an adjacency matrix, and it is used as an input parameter of both mapping heuristics.

The Exhaustive Search (ES) is used as performance baseline. It provides the best process mapping, but it takes a factorial time and is not suited for a considerable number of processes and cores. Therefore, a good solution to overcome this issue is to use a method which finds a reasonably good mapping in polynomial time. So, the reduction of the application performance is amortized by reducing the time needed to find a mapping. We compare the MWPM and DRB heuristics with ES to examine the performance difference between these algorithms.

All these methods are focused on process mapping on cores which share the L2 cache. As output, they generate an affinity file containing the ranks of the processes and core numbers on which they should be mapped. The following subsections describe the ES method and the two heuristics, MWPM and DRB.

2.1. Exhaustive Search

This method obtains the best process placement in terms of communication amount by searching exhaustively all combinations of process pairs that can be mapped on pairs of cores. After finding the best combination, it maps each pair of processes to a respective pair of cores which share the L2 cache.

The algorithm receives a list of all possible process pairs where each list element is a possible process mapping. The best mapping is defined by the element of the list where the sum of communication amount of all process pairs is the highest. Finally, these pairs are mapped to pairs of cores which share the L2 cache.

Although the ES method provides the best process mapping in terms of communication amount, it is not practical because the search comprises all tasks placement possibilities, which takes a factorial time.

2.2. Maximum Weighted Perfect Matching

The MWPM algorithm creates a process placement in polynomial time by modeling the mapping problem as a maximum graph pairing with minimum cost. This approach is a feasible solution and works in three steps. In the first step, the algorithm groups the processes that have a significant amount of exchanged messages in pairs, and places each of them in pairs of cores that shares a L2 cache and taking advantage of *intrachip* communication. The second step allocates each pair of processes-pairs on the processors using the *intranode* communication. The third step distributes the pairs of processes-pairs on the nodes considering the *internode* communication [7].

Basically, the algorithm works by choosing from the application graph $A(P, B)$ the processes that should stay closer within the memory hierarchy based on the amount of messages. The problem is to find a subset M of B so that for every process $p \in P$ there is only one edge $b \in B$ to which p is attached, and the sum of all edges $b \in M$ is maximal. This problem is solved by the Edmonds-Karp algorithm with a time complexity of $O(N^3)$ [8]. We applied this algorithm three times for each type of communication.

2.3. Dual Recursive Bipartitioning

The DRB algorithm is based on the divide and conquer method to generate the mapping graphs $A(P, B)$ and $T(C, L)$ [10]. $T(C, L)$ is the processor architecture graph, where C represents the cores of the processors and L the set of edges. $L(c_x, c_y)$ is defined as the latency of the communication channel between the cores c_x and c_y.

To create the mapping, the algorithm starts with the set of processes P of the application graph A, and a domain structure comprising the whole set of cores C. Then, it applies the functions domain bipartitioning and process bipartitioning on the domain structure and set of processes, respectively. This way, the domain structure is divided into two subdomains of cores, and the set of processes, into two disjoint subset of processes. The next step of the algorithm consists of attaching each of the new subsets of processes to a subdomain, and consequently minimizing the communication volume between these two subsets. These stages are repeated recursively until the set of processes has only one process and the subdomain has only one core, so the process singleton subset is mapped to a core singleton subdomain [11].

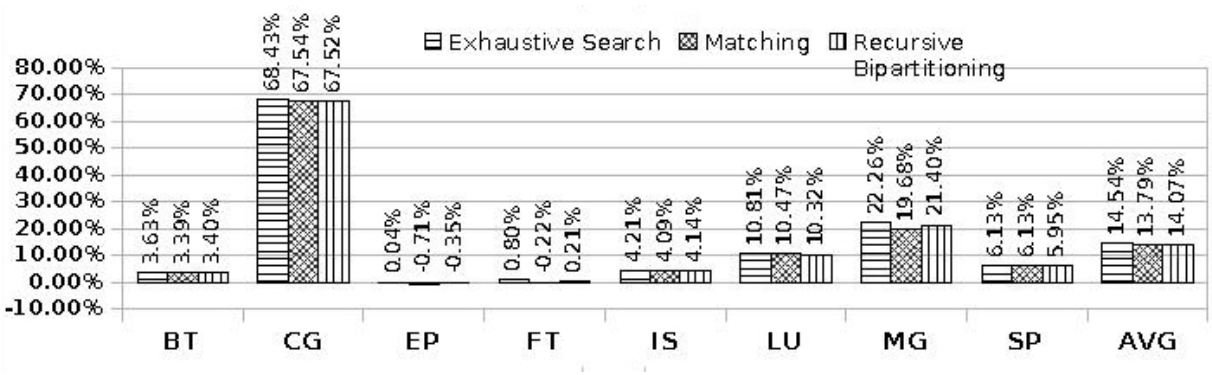

Figure 1. Performance Improvement Compared to the MPICH2 Default Mapping

3. Experimental Results

To evaluate the three algorithms, we used the BT, CG, EP, FT, IS, LU, SP and MG benchmarks of the NAS-NPB-MPI benchmark suite. All of them were compiled for the C class size and executed with 16 processes. To capture the communication pattern of each application and create their respective graphs $A(P, B)$ based on the amount of data exchanged between the processes, we executed all applications using the MPE library of the MPICH2-1.2 MPI distribution. Then, we applied the mapping generated by the three algorithms to a cluster containing two nodes with two Intel Xeon E5405 quad-core processors each. Each processor contains two 6MB L2 caches, each of which is shared between two cores. Finally, we measured the *speedup* of the ES, MWPM and DRB algorithms compared to the MPICH2 default mapping (round-robin). Table 1 shows the execution time of these four mappings and the variability for a confidence interval of 95%. Figure 1 shows the performance improvements of the three algorithms compared to the MPICH2 default mapping.

The FT and EP benchmarks present negative or very small improvement because all processes of FT exchange the same amount of messages, while the processes of EP do not exchange any messages. The CG benchmark has the best performance improvement for both heuristics since it has a higher difference of communication volume between specific pairs of processes, which is a good use case for the mapping algorithms.

We observed that the MWPM and the DRB algorithms improve the performance by the same amount. Moreover, their improvement is very close to the improvement achieved when mapping using the ES algorithm. Using ES, performance was improved by 14.54% in average, compared to 13.79% when using MWPM and 14.07% when using the DRB. We concluded that, if we execute the applications using 16 processes as we used in our experiments, it is better to use the ES algorithm to generate the best static process mapping. However, even though MWPM and DRB makes similar results, MWPM was developed for homogeneous architectures that share the L2 cache between pairs of cores, DRB can be applied on generic hardware architectures. Hence the latter algorithm is recommended in most static process mapping situations, but more research is needed to validate these results.

4. Related Works

A strategy to reduce the communication latency in parallel applications that have a static communication pattern is presented in [11]. It uses the DRB algorithm applied to a meteorology application, reaching a *speedup* of 9% using a technique that considers the cache memory sharing within a chip. However, as in our work, a previous execution of the application is required, and it is advantageous only in applications which are executed repeatedly, where the time spent in the previous execution is compensated by the *speedup* achieved on next computations. In our work, the DRB is not restricted only to one application, but it is executed in a generic way to compare the heuristics.

The work in [12] evaluates the process affinity on static process mapping with MPI for SMP-CMP clusters, because the authors argue that its impact on parallel applications performance is not very clear. The execution of NAS benchmarks is performed to analyze some characteristics, like scalability and communication latency, that indicates a reasonable affinity usage. In our work, we also analyze the impact of process affinity, but through three process mapping methods that explore the communication latency characteristic.

The Hardware Locality (hwloc) is a software presented in [4] that collects hardware information related to cores, caches and memory, and provides them to the mapping application. It was evaluated in pure MPI and hybrid MPI/OpenMP applications which use the information

	Default	Confid. Interv. 95%		ES	Confid. Interv. 95%		MWPM	Confid. Interv. 95%		DRB	Confid. Interv. 95%	
BT	373.37	373.04	373.69	360.29	359.68	360.9	361.14	360.87	361.4	361.09	360.79	361.4
CG	184.42	184.32	184.52	109.5	109.12	109.88	110.08	109.84	110.31	110.09	109.8	110.38
EP	34.54	34.49	34.59	34.53	34.26	34.79	34.79	34.38	35.2	34.66	34.31	35.01
FT	203.02	202.04	204.01	201.42	200.49	202.34	203.48	202.43	204.52	202.6	201.96	203.24
IS	18.03	17.89	18.16	17.3	17.2	17.4	17.32	17.2	17.44	17.31	17.2	17.43
LU	239.46	239.02	239.89	216.09	215.7	216.48	216.77	216.46	217.08	217.06	216.84	217.28
MG	40.69	40.58	40.8	33.28	33.26	33.3	34	33.63	34.37	33.52	33.21	33.82
SP	521.71	520.13	523.3	491.6	490.92	492.27	491.6	490.92	492.27	492.43	491.85	493.01

Table 1. Execution Times

provided by this software to execute a static and dynamic process mapping. The experiments presented in our work could be extended by using this tool to recognize the computer architecture when it is not known beforehand.

5. Conclusions and Future Works

In this paper we have done a performance evaluation with two static process mapping heuristics based on processes communication volume using the Exhaustive Search algorithm as a baseline for comparison. The results show that performance improvement of both MWPM and DRB are close to ES. For the execution on 16 cores, the ES had an average improvement of 14.54%, compared to 13.79% from MWPM and 14.07% from DRB, which means that both heuristics achieved equivalent performance improvements.

However, MWPM was developed for homogeneous architectures that share L2 cache between each pair of cores, while DRB does not have this restriction, since it can be applied to generic hardware architectures. It made us assume that this last heuristic could be used in most situations, but more research is needed to validate these results.

For the future, we intend to evaluate these heuristics in a non-dedicated environmnent to take into account the contention on the communication caused by the execution of other parallel applications which do not belong to the evaluated benchmark. In parallel we will also generate more execution time results considering different computer architectures rather than considering only processors that share L2 caches between pairs of cores.

References

[1] National Aeronautics and Space Administration (NASA). NAS Parallel Benchmarks (NPB3.3), Available in http://www.nas.nasa.gov/Resources/Software/npb.html, accessed in may 2010.

[2] K. Asanovic, et al. A view of the parallel computing landscape. *Commun. ACM*, 52:56–67, October 2009.

[3] S. Bokhari. On the mapping problem. *Computers, IEEE Transactions on*, C-30(3):207–214, march 1981.

[4] F. Broquedis, et al. hwloc: A generic framework for managing hardware affinities in hpc applications. *Parallel, Distributed, and Network-Based Processing, Euromicro Conference on*, 0:180–186, 2010.

[5] D. Buntinas, G. Mercier, and W. Gropp. Implementation and evaluation of shared-memory communication and synchronization operations in mpich2 using the nemesis communication subsystem. *Parallel Computing*, 33(9):634 – 644, 2007. Selected Papers from EuroPVM/MPI 2006.

[6] L. Chai, Q. Gao, and D. Panda. Understanding the impact of multi-core architecture in cluster computing: A case study with intel dual-core system. In *Cluster Computing and the Grid, 2007. CCGRID 2007. Seventh IEEE International Symposium on*, pages 471 –478, may 2007.

[7] E. Cruz, M. Alves, and P. Navaux. Process mapping based on memory access traces. In *Computing Systems (WSCAD-SCC), 2010 11th Symposium on*, pages 72 –79, oct. 2010.

[8] V. Kolmogorov, V. Blossom A new implementation of a minimum cost perfect matching algorithm. In *Mathematical Programming Computation*, pages 43-67, 2009.

[9] C. Osiakwan and S. Akl. The maximum weight perfect matching problem for complete weighted graphs is in pc. In *Parallel and Distributed Processing, 1990. Proceedings of the Second IEEE Symposium on*, pages 880 –887, dec 1990.

[10] F. Pellegrini and J. Roman. Experimental analysis of the dual recursive bipartitioning algorithm for static mapping. Technical report, TR 1038-96, LaBRI, URA CNRS 1304, Univ. Bordeaux I, 1996.

[11] E. Rodrigues, F. Madruga, P. Navaux, and J. Panetta. Multi-core aware process mapping and its impact on communication overhead of parallel applications. In *Computers and Communications, 2009. ISCC 2009. IEEE Symposium on*, pages 811 –817, july 2009.

[12] C. Zhang, X. Yuan, and A. Srinivasan. Processor affinity and mpi performance on smp-cmp clusters. In *Parallel Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium on*, pages 1 –8, april 2010.