A METHODOLOGY FOR ADAPTIVE RESOLUTION OF NUMERICAL PROBLEMS ON HETEROGENEOUS HIERARCHICAL CLUSTERS

Wahid Nasri, Sonia Mahjoub and Slim Bouguerra
ESSTT, 5 Avenue Taha Hussein - B.P. 56, Bab Menara - 1008 Tunis, Tunisia

Keywords: Cluster computing, Adaptive techniques, Scheduling, Parallel algorithms, Matrix multiplication problem.

Abstract: Solving a target problem by using a single algorithm or writing portable programs that perform well is not always efficient on any parallel environment due to the increasing diversity of existing computational supports where new characteristics are influencing the execution of parallel applications. The inherent heterogeneity and the diversity of networks of such environments represent a great challenge to efficiently implement parallel applications for high performance computing. Our objective within this work is to propose a generic framework based on adaptive techniques for solving a class of numerical problems on cluster-based heterogeneous hierarchical platforms. Toward this goal, we refer to adaptive approaches to better adapt a given application to a target parallel system. We apply this methodology on a basic numerical problem, namely solving the matrix multiplication problem, while determining an adaptive execution scheme minimizing the overall execution time depending on the problem and architecture parameters.

1 INTRODUCTION

Few years ago, there was a huge development of new parallel and distributed systems. Large collections of interconnected PCs (called clusters) have replaced traditional super-computers in many universities and companies. Due to the increasing performance of on-the-shelf components, such low-cost systems are a reasonable alternative for solving a large range of applications.

However, the introduction of such parallel systems has a major impact on the design of efficient parallel algorithms. Indeed, new characteristics have to be taken into account including scalability and portability. Moreover, such parallel systems are often upgraded with new generation of processors and network technologies. Today, as the systems are composed of collections of heterogeneous machines, it is very difficult for a user to choose an adequate algorithm because the execution supports are continuously evolving. One version will be well-suited for a parallel configuration and not for another. This portability issue becomes crucial because of the frequent changes of the components of the systems. These different elements require to revise the classical parallel algorithms which consider only regular architectures with static configurations and to propose new approaches.

The adaptive approaches are a promising answer to this problem. The idea is to adapt algorithms together with their execution to the target architecture. These algorithms may be automatically adapted to the execution context (data and support).

Our objective within this work is to propose a generic framework including the design of an automatic selection mechanism, based on adaptive techniques for dealing with scalability and portability issues on cluster-based heterogeneous hierarchical platforms for a class of regular numerical algorithms. The proposed methodology may be extended to a class of parallel applications which can be partitioned in a set of independent tasks (which may be non identical).

The remainder of the paper is organized as follows. We begin in section 2 by presenting the architectural model of the target parallel and distributed system and discussing some adaptive approaches. In section 3, we describe our adaptive framework and detail its components. Section 4 is devoted to a case study where we apply our methodology on the matrix multiplication problem. Section 5 concludes the paper and discusses some perspectives to extend this work.

2 BACKGROUND

2.1 Description of the Architectural Model

We assume in this work a generic model of a platform composed of heterogeneous hierarchical clusters as described in (Capello, 2005). The studied platform enjoys heterogeneity along three orthogonal axis : (a) The processors that populate the clusters may differ in computational powers, even within the same cluster, (b) The clusters composing the platform are organized hierarchically and are interconnected via a hierarchy of networks of possibly differing latencies, bandwidths and speeds. At the level of physical clusters, the interconnection networks are assumed to be heterogeneous, and (c) The clusters at each level of the hierarchy may differ in sizes.

We will extend this architecture to a one where the capacities of the links connecting clusters may change dynamically during the execution of the target parallel application.

2.2 Adaptive Approaches

It is well-known that no single algorithm can always achieve the best performance of a sequential or parallel application for different problem sizes and number of processors on a target parallel system. We can obtain good performances by mixing multiple algorithms for solving the same problem, where each algorithm can dominate the others in specific contexts. Thus, we should determine the more appropriate algorithm (which provides the best performance) in terms of a set of parameters (size of the problem, number of available processors, performances of the interconnection network, etc.), or to combine multiple ones for improving performances to fit well the characteristics of the target computational system. The software mechanism responsible for determining the best available choices at run-time is known as a switching function. The optimal choice of algorithm can be determined at run-time, typically by using data obtained by monitoring tools, such as the NWS (Network Weather Service) (Wolsky, 1997) which permits to measure many useful information, such as the hardware characteristics, the communication bandwidth, the system load, or any input-data that may influence the performance of the application. The result of this mechanism is called adaptive algorithm. This algorithm may use different techniques to adaptively determine the best algorithm. For instance, the algorithms presented in (Frigo, 1998; Thomas, 2005) use respectively machine learning and cascading techniques.

3 DESCRIPTION OF THE ADAPTIVE METHODOLOGY

In this section, we describe our methodology for adaptively executing parallel applications in an execution environment characterised by its heterogeneity and its hierarchical organization. An overview of the methodology is sketched in figure 1. The processing is separated in two successive phases. During the first one, we aim to partition the target platform to form subnets of similar characteristics by automatically discovering the network topology. Then, when executing the second phase, we have to determine for each subnet (i.e. cluster) the more appropriate algorithm among multiple algorithmic options leading to the minimum possible execution time of the given problem. We will finally determine an adaptive execution scheme identifying the details of the implementation. It is worthy to note that we may have at a given time different algorithms occurring on different clusters. Moreover, it is possible to use a combination of many algorithms to execute a task on the same cluster. In the sequel, we more detail the major components of the methodology.

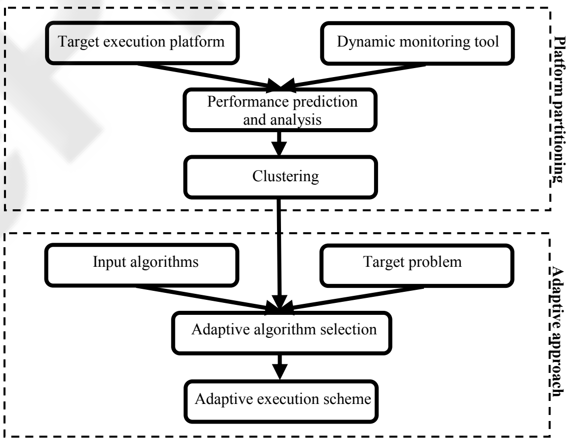
Target execution platform
Dynamic monitoring tool
Performance prediction and analysis
Clustering
Platform partitioning
Input algorithms
Target problem
Adaptive algorithm selection
Adaptive execution scheme
Adaptive approach

Figure 1: Adaptive methodology.

3.1 Partitioning the Platform

Since the target parallel system may be heterogeneous at many levels (computing powers, interconnection network performances, etc.), it is very difficult to manage such platform towards a high performance computing. One way to answer this problem and to minimize the inherent heterogeneity, and thus facilitating the execution, is to subdivide the network in homogeneous subnets (or logical clusters), as described below. At the end of this phase, we will obtain a set of logical clusters of homogeneous interconnection networks, which will be used to adaptively implementing algorithms inside each cluster during the second phase of the methodology.

3.1.1 Network Performance Measurments

The methodology starts by collecting available information from the target execution environment to be used in the step of clustering (see next section). There exist many tools for network monitoring, such as NWS (Network Weather Service). These tools permit to determine many useful parameters of the target parallel system like the current network status, the communication latency, the speeds of the processors, the CPU load, the available memory, etc. For instance, the communication latency and throughput permit to identify groups of machines with similar communication parameters.

3.1.2 Clustering

One reason to construct logical clusters is that machines may behave differently, and the easiest way to optimise communications is to group machines with similar performances (Barchet-Estefanel, 2004). In order to classify nodes in logical clusters, we can use a clustering algorithm similar to the one presented in (Lowekamp, 1996). This algorithm analyses each interconnection on the distance matrix containing the latencies between links in order to group nodes for which their incident edges respect a latency bound (by default 20%) inside that subnet. Note that the distance matrix was obtained when applying NWS on the clusters to determine the network information.

3.2 Adaptive Approach

Once the platform is partitioned in separated homogeneous hierarchical clusters, we have at this stage to determine, using an adaptive approach, the more performant algorithm from a set of algorithms reserved to solve the problem for each cluster. This mechanism may lead to a collection of various methods to be used at the same time on the available clusters. Any necessary characteristics are measured during the first phase corresponding to the network partitioning. We recall that the adaptive decision is made in terms of many information which might be of interest, such as those related to the target problem (size and type of data) and other ones related to the architecture structure (interconnection network, number of nodes, etc.). This phase ends by fixing an execution scheme detailing the implementation. Let us precise that the adaptive algorithm selection is based on analytical models able to predict performances of the parallel application on the target platform.

3.2.1 Strategy of Task Allocation

Assume that we have to execute a set of tasks. Our strategy requires to reserve a node (called *coordinator*) for controlling the overall execution of the tasks, and one node per cluster to be charged for communicating with the other clusters.

The coordinator starts the execution by assigning a task to each available cluster. Let us recall that clusters may have different performances and tasks may be non identical. Once a cluster finishes the execution of its task, it sends a request to the coordinator to get another task. Then, the coordinator proceeds first by identifying the necessary data to execute the task, which are assumed to be distributed over the platform, their locality (which clusters have the data), and then determining the path minimizing the transfer cost.

3.2.2 The Makespan Improvement Phase

Let us remark that at the end of the execution process, when the number of remaining tasks to execute is less than the number of available clusters, we can observe idle times which may be high especially when the idle clusters are the fastest ones. Figure 2 shows an example where we consider four clusters (C1, C2, C3 and C4). At the instant t1, C4 finishes the execution of its allotted task. Then, it asks the coordinator for another task. We assume at this level that it remains only two tasks to execute. C4 will be allotted a task and the last one will be executed by C3.

At this stage, we propose a strategy inspired from the technique of work-stealing (Blumofe, 1998) in order to reduce the idle time and improve performances. In this case, the first cluster becoming idle at the instant t2 (here is C1) asks the coordinator for identifying which cluster to be concerned with the stealing. The coordinator, having a global status of the execution process, determines the slowest cluster in the sense that finishes last, and decides if it will be performant to share the remaining portion of the task on two clusters (the requester and the slowest, i.e. C1 and C4 respectively). Similarly, the task under execution on C3 will be shared with C2. Let us mention here that sharing the execution is achieved only if this processing is able to reduce the execution time. Figure 2 shows a possible improvement leading to a reduced global execution time.

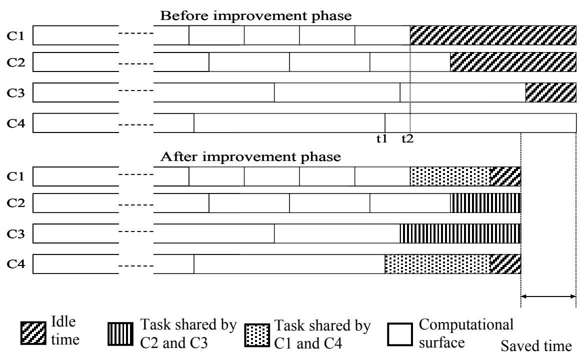

Figure 2: Scheduling tasks and improvement phase.

4 CASE STUDY: MATRIX MULTIPLICATION PROBLEM

We apply our adaptive methodology on a basic numerical problem, namely computing the product of two (large) dense square matrices. We begin by discussing some related works, and then describing our adaptive algorithm.

4.1 Related Works

The parallelization of the matrix multiplication problem was widely studied in the literature. Various optimized versions of this problem have been implemented in libraries on all existing (homogeneous or heterogeneous) parallel systems. We may particularly refer to works presented in (Beaumont, 2001; Desprez, 2004; Hunold, 2004; Lastovetsky, 2004; Ohtaki, 2004) where various methods have been applied, such as standard, fast, mixed, etc. However, only few parallel adaptive implementations have been developed.

However, to the best of our knowledge, no original work has been devoted to implement adaptive algorithms for matrix multiplication on heterogeneous hierarchical clusters where both computing resources and interconnection links are heterogeneous. Moreover, the network capacity may change dynamically during execution. The contribution of this paper is to intend to fill this gap.

4.2 Based Algorithms

We will use three based algorithms in our approach. The first one was designed by Beaumont et al. (Beaumont, 2004) where classical matrix multiplication algorithms have been implemented on heterogeneous clusters. The algorithms presented are very efficient but the distribution used is highly irregular. The second algorithm is proposed by Lastovetsky and Reddy (Lastovetsky, 2004) which have extended a two-dimensional homogeneous block-cyclic distribution to heterogeneous case that provides perfect load balancing on a grid of processors. The last algorithm is developed by Ohtaki et al. (Ohtaki, 2004) where they propose a recursive data decomposition, which enables both efficient load balancing and incrementing of the recursion level in Strassen's algorithm for heterogeneous clusters. Let us precise that we may use more based algorithms, but we will generate an additional cost for the poly-algorithmic decision.

4.3 Description of the Proposed Adaptive Algorithm

4.3.1 Data Distribution

Let A, B and C=A*B be three square matrices of size n. We assume that due to a previous work, the input matrices A and B are distributed on two different clusters. Our objective here is to distribute the matrices over the available clusters. To compute C, we propose to partition the matrix into equal square blocks of size r each. The size is chosen so that we create coarse-grained tasks that are assigned later to disjoint clusters. Computing a block of C requires a row block of A and a column block of B. So, the initial data distribution is a row block-wise distribution for A and a column block-wise distribution for B. We have now to allocate tasks on (say x) available clusters. The initial allocation that we adopt is presented in figure 3, where we allot cyclically a column of blocks to each cluster. Let us mention that this preliminary allocation will be dynamically adapted during execution.

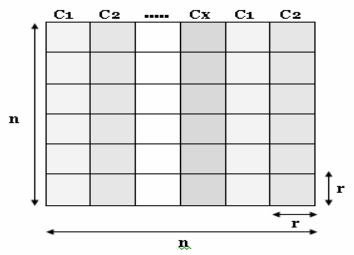

Figure 3: Task allocation on x clusters.

To be able to start computing blocks, we have to send to each cluster a row block of matrix A and a column block of matrix B. For a new block of the same column block of matrix C, a cluster requires only a row block of A. For a new column block of C allotted to the same cluster, the later requires only a column block of B for the first square block to compute; for the remaining blocks, the cluster has all necessary data. Since the fastest clusters will finish the execution of their allotted tasks first, we will apply the strategy described in section 3.2.2 to reduce the idle time of these clusters when they have no work to perform.

Remark that at the beginning, all clusters require the same row block of matrix A to start execution. Since the corresponding data is located in a one cluster, we will have to broadcast it on clusters. Each cluster can start its computation as soon as the data is available. We describe in the following how to schedule tasks during the execution.

4.3.2 Scheduling Tasks

Since our problem is reduced here to a set of independent tasks, we will apply the approach proposed in section 3.2. Let us describe the methodology with an example. Consider the platform presented in figure 4, composed initially of three different clusters C1, C2 and C3, with a 1-Port model and two level hierarchy networks with different bandwidths and latencies. After applying the first phase of the methodology corresponding to partitioning the platform in logical homogeneous clusters, C3 will be separated into two sub-clusters C31 and C32. We assume here that the coordinator is a node of C2, matrices A and B are initially located in C31 and C2 respectively, and that the matrix decomposition in blocks leads to 25 tasks to be scheduled on the four clusters.

4.3.3 Adaptive Algorithm Selection

The selection of the best algorithm to execute a given task is based on a performance matrix containing the duration of each algorithm, among the set of input algorithms, on each cluster. This matrix is obtained using analytical models. Formally, assuming a cost model, we denote by $P(Alg_i,C_j)$ the performance of algorithm Algi on cluster Cj. Algi is qualified to be the best on cluster Cj when $P(Alg_i,C_j)=\min\{P(Alg_k,C_j), 1\leq k\leq q\}$, where

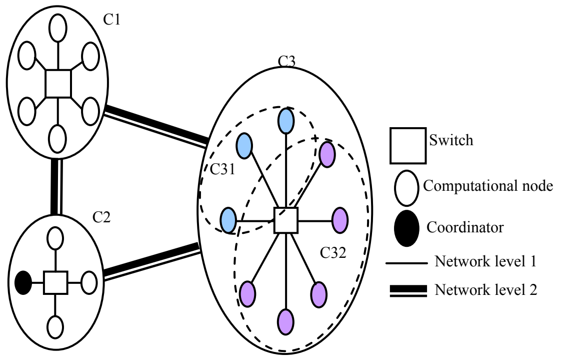

Figure 4: The platform after clustering.

q represents the number of available algorithms. It is worthy to note that due to the diversity of clusters composing the platform, we may have at a given time various algorithms executing different tasks, each on a cluster.

Figure 5 shows the scheduling of the different tasks of the example presented previously when the more performant algorithm is used on each cluster. We assumed, since clusters are different in computing powers, that the execution time of a task, which remains unchanged using a given algorithm, is different on two different clusters. We also considered that overhead due to communicating the same amount of data may change from an execution to another due to a possible variation of the network capacity.

5 CONCLUDING REMARKS AND FUTURE WORKS

We have presented in this paper a new (two phase) methodology based on adaptive approaches, including the design of an automatic selection mechanism, for dealing with parallel

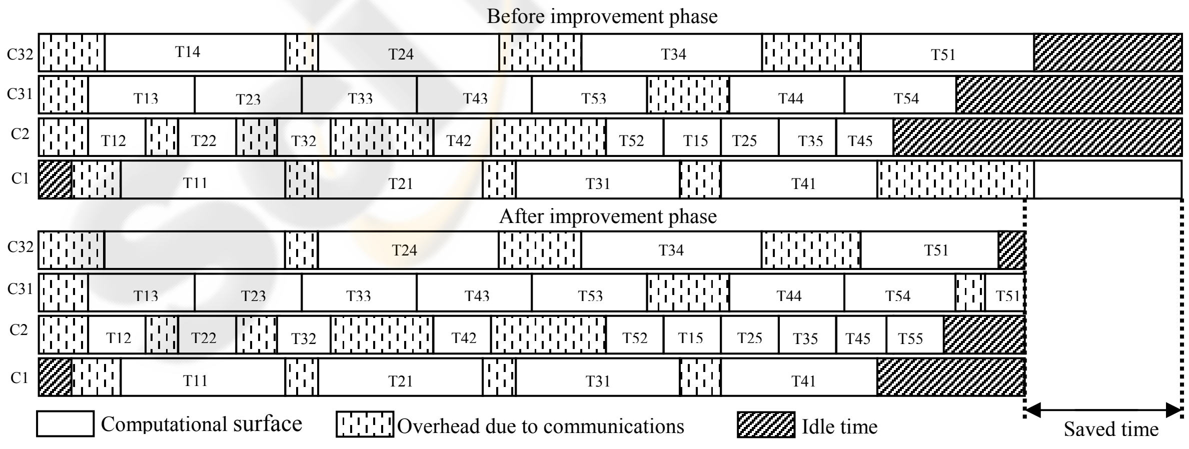

Figure 5: Scheduling tasks before and after improvement phase.

implementations of a class of regular numerical algorithms and parallel applications which may be partitioned in a set of independent tasks on cluster-based heterogeneous hierarchical platforms. We applied the approach on a basic numerical problem, namely solving the matrix multiplication problem, while achieving the minimum possible execution time depending on the problem and architecture parameters.

As future prospects, we first intend to validate this approach by achieving experiments on real platforms, and apply the methodology on other types of parallel applications. We also plan to integrate other existing adaptive approaches to our framework to benefit from the powerful of these techniques.

REFERENCES

Barchet-Estefanel, L. A. and Mounié, G., 2004, Identifying Logical Homogeneous Clusters for Efficient Wide-area Communications, *In Proceedings of the Euro PVM/MPI*, 2004.

Beaumont, O., Boudet, V., Rastello, F. and Robert, Y., 2001, Matrix Multiplication on Heterogeneous Platforms, *IEEE Transactions on Parallel and Distributed Systems*, 12 (10), 2001.

Blumofe, R. D. and Leiserson, C. E., 1998, Space-Efficient Scheduling of Multithreaded Computations, *SIAM Journal on Computing*, 27(1), 1998.

Bosilca, G., Chen, Z., Dongarra, J., Eijkhout, V., Fagg, G. E., Fuentes, E., Langou, J., Luszczek, P., Pjesivac-Grbovic, J., Seymour, K., You, H. and Vadhiyar, S. S., 2005, Self Adapting Numerical Software (SANS) Effort, *IBM Journal of Research and Development*, 2005.

Capello, F., Fraigniaud, P., Mans, B. and Rosenberg, A. L., 2005, An Algorithmic Model for Heterogeneous Hyper-Clusters: Rationale and Experience, *International Journal of Foundations of Computer Science* 16(2), 195-215, 2005.

Chen, Z., Dongarra, J., Luszczek, P. and Roche, K., 2003, Self Adapting Software for Numerical Linear Algebra and Lapack for Clusters, *Parallel Computing*, 2003.

Daoudi, E. M., Gautier, T., Kerfali, A., Revire, R. and Roch, J.-L., 2005, Algorithmes parallèles à grain adaptatif et applications, *Techniques et Sciences Informatiques*, Hermès, 2005.

Desprez, F. and Suter, F., 2004, Impact of mixed-parallelism on parallel implementations of the Strassen and Winograd matrix multiplication algorithms, *Concurrency and computation : practice and experience*, 16, 2004.

Dutot, P. -F., Mounié, G. and Trystram, D., 2004, Scheduling Parallel Tasks — Approximation Algorithms, chapter 26 of the handbook of scheduling, edited by Joseph Y-T. Leung, 2004.

Eskenazi, E. M., Fioukov, A. V., Hammer, D. K., Obbink, H. and Pronk, B., 2004, Analysis and Prediction of Performance for Evolving Architectures, *In Proceedings of the 30th EUROMICRO Conference (EUROMICRO'04)*, 2004.

Frigo, M. and Johnson, S., 1998, FFTW : an adaptive software architecture for the Fast Fourier Transform, *In Proceedings of ICASSP*, 1998.

Hartmann, O., Kuhnemann, M., Rauber, T. and Runger, G., 2006, Adaptive Selection of Communication Methods to Optimize Collective MPI Operations, *In Proceedings of the 12th Workshop on Compilers for Parallel Computers (CPC'06)*, 2006.

Hong, B. and Prasanna, V. K., 2002, Adaptive Matrix Multiplication in Heterogeneous Environments, *In Proceedings of the 9th International Conference on Parallel and Distributed Systems (ICPADS'02),* 2002.

Hunold, S., Rauber, T. and Runger, G., 2004, Multilevel Hierarchical Matrix Multiplication on Clusters, *In Proceedings of the 18th International Conference on Supercomputing*, 2004.

Lastovetsky, A. and Reddy, R., 2004, On performance analysis of heterogeneous parallel algorithms, *Parallel Computing*, 30, 2004.

Li, J., 1996, A poly-algorithm for parallel dense matrix multiplication on two dimensional process grid topologies, *PhD Thesis*, University of Mississippi, 1996.

Lowekamp, B. B. and Beguelin, A., 1996, ECO: Efficient Collective Operations for Communication on Heterogeneous Networks, *In Proceedings of the 10th International Parallel Processing Symposium*, 1996.

McCracken, M. O., Snavely, A. and Malony, A. D., 2003, Performance Modeling for Dynamic Algorithm Selection, *International Conference on Computational Science*, 2003.

Nasri, W., Trystram, D. and Achour, S., 2006, Adaptive Algorithms for the Parallelization of the Dense Matrix Multiplication on Clusters, *International Journal of Computational Science and Engineering*, to appear, 2006.

Ngoko, Y., 2005, Poly-algorithmes pour une programmation efficace des problèmes numériques. Exemple du produit de matrices, *Master Thesis, University of Yaoundé I*, 2005.

Ohtaki, Y., Takahashi, D., Boku, T. and Sato, M., 2004, Parallel Implementation of Strassen's Matrix Multiplication Algorithm for Heterogeneous Clusters, *IPDPS'04*, 2004.

Roche, K. J. and Dongarra, J. J., 2002, Deploying parallel numerical library routines to cluster computing in a self adapting fashion, *Parallel Computing, Advances and Current Issues*, 2002.

Thomas, N., Tanase, G., Tkachyshyn, O., Perdue, J., Amato, N. M. and Rauchwerger L., 2005, A Framework for Adaptive Algorithm Selection in STAPL, *In Proceedings of PPoPP'05*, 2005.

Whaley, R. C., Petitet, A. and Dongarra, J. J., 2001, Automated empirical optimizations of software and the ATLAS project, *Parallel Computing*, 27, 2001.

Wolski, R., Spring, N. and Peterson, C., 1997, Implementing a Performance Forecasting System for Metacomputing : The Network Weather Service, *In Supercomputing*, 1997.