Future Generation Computer Systems 66 (2017) 11–26

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

An approach for an efficient execution of SPMD applications on Multi-core environments

Ronal Muresano[a,*], Hugo Meyer[a,b], Dolores Rexachs[a], Emilio Luque[a]

[a] Computer Architecture and Operating System Department (CAOS), University Autonoma of Barcelona (UAB), Barcelona, Spain

[b] Computer Sciences Group, Barcelona Supercomputing Center (BSC-CNS), Barcelona, Spain

HIGHLIGHTS

- A method for efficient execution on Multicore cluster is presented.
- The method combines efficiency and speedup in order to improve the performance execution on multi-core clusters.
- A mapping and a scheduling techniques are proposed in order to improve the efficiency and speedup.
- The method finds the maximum strong and weak scalability point with error rates lower than 5%.
- Considerable improvements are achieved using the method on large scale systems.

ARTICLE INFO

Article history:
Received 4 June 2015
Received in revised form
16 November 2015
Accepted 19 June 2016
Available online 1 July 2016

Keywords:
Performance improvements
Multi-core
Mapping
Scheduling
Scalability analysis
SPMD

ABSTRACT

Executing traditional Message Passing Interface (MPI) applications on multi-core cluster balancing speed and computational efficiency is a difficult task that parallel programmers have to deal with. For this reason, communications on multi-core clusters ought to be handled carefully in order to improve performance metrics such as efficiency, speedup, execution time and scalability. In this paper we focus our attention on SPMD (Single Program Multiple Data) applications with high communication volume and synchronicity and also following characteristics such as: static, local and regular. This work proposes a method for SPMD applications, which is focused on managing the communication heterogeneity (different cache level, RAM memory, network, etc.) on homogeneous multi-core computing platform in order to improve the application efficiency. In this sense, the main objective of this work is to find analytically the ideal number of cores necessary that allows us to obtain the maximum speedup, while the computational efficiency is maintained over a defined threshold (strong scalability). This method also allows us to determine how the problem size must be increased in order to maintain an execution time constant while the number of cores are expanded (weak scalability) considering the tradeoff between speed and efficiency. This methodology has been tested with different benchmarks and applications and we achieved an average improvement around 30.35% of efficiency in applications tested using different problems sizes and multi-core clusters. In addition, results show that maximum speedup with a defined efficiency is located close to the values calculated with our analytical model with an error rate lower than 5% for the applications tested.

1. Introduction

The increasing use of multi-core processors in High Performance Computing (HPC) is evident in the top500,[1] in which most of today's clusters are set up with multi-core architecture. However, the increase in complexity and the hierarchical communication architecture present on these multi-core clusters create significant programming challenges which have to be managed carefully if programmers wish to harness the inclusion of more parallelisms inside the nodes [1,2]. The parallel programmers have to deal with some architectural characteristics, such as: number of cores per chip, shared cache between cores, bus interconnection, memory bandwidth, communication congestion, etc. [3]. All these elements are becoming more important for programmer to consider,

* Corresponding author.
E-mail addresses: rmuresano@caos.uab.es (R. Muresano), hugo.meyer@caos.uab.es, hugo.meyer@bsc.es (H. Meyer), dolores.rexachs@uab.es (D. Rexachs), emilio.luque@uab.es (E. Luque).

[1] TOP500: a list which provides a rank of powerful parallel machines for HPC on the world, www.top500.org.

http://dx.doi.org/10.1016/j.future.2016.06.016
0167-739X/© 2016 Elsevier B.V. All rights reserved.

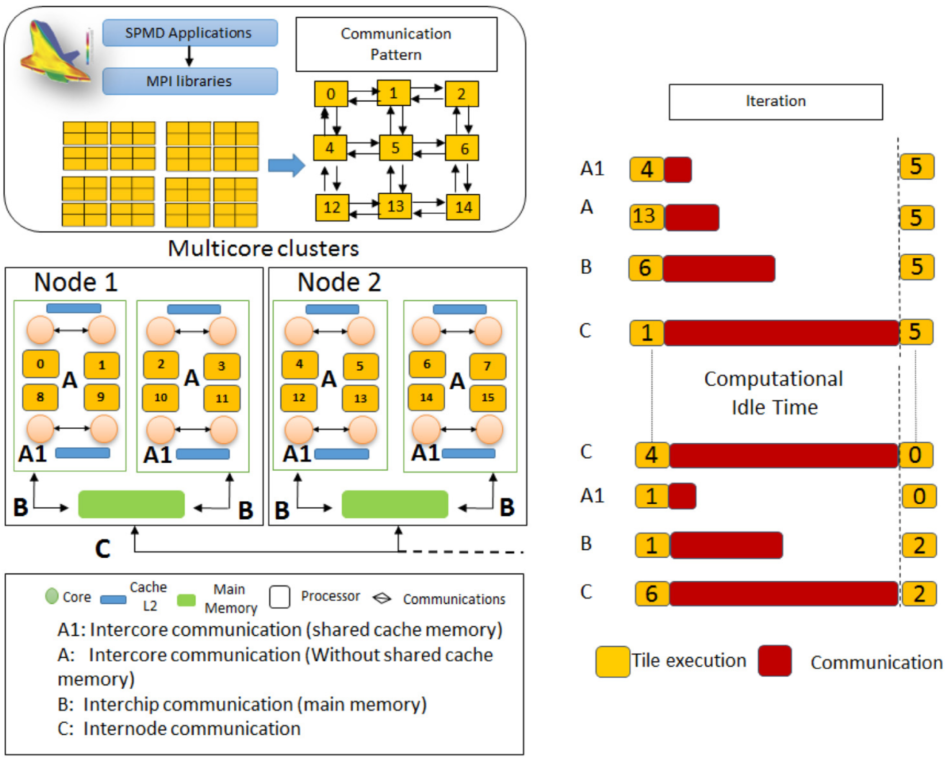

Fig. 1. Mapping and execution of SPMD applications on a Multi-core cluster.

in case that application's scalability and efficiency want to be improved [4].

The multi-core nodes integrate a homogeneous computation architecture, which in some cases are composed of 2, 4, 6, 8, etc., cores by chip processors. However, a node can include several chip processors creating a small high speed parallel machine inside the node. Nevertheless, these nodes have to be analyzed as heterogeneous when we are working with applications that have a very high communication frequency between parallel processes. The communications between cores in these architectures use a hierarchical communication architecture that uses different paths and speeds to perform the communication processes inside the node [5–7]. For example, the parallel processes in a multi-core cluster can communicate using the cache memory or main memory for communications inside the node (Intercore and Interchip communications), or using the local area network to perform the communication with another process located in another node of the cluster (Internode communication). This communication architecture can create unbalanced issues that seriously affect the application performance, especially those applications which have a very coupled behavior (Fig. 1).

Performance metrics that are commonly used to measure, such as: execution time, speedup, computational efficiency and strong and weak application scalability are all seriously affected. All these metrics are influenced in different ways due to the degradations and load balancing problems generated by the communications links [8]. Another important aspect to consider is that many MPI applications have been designed without considering the computational architecture characteristics. An example is the monocore nodes, where the communication processes were homogeneous and most of them have to be updated in order to take advantages of multi-core architecture.

A parallel paradigm which is seriously affected when executed on a hierarchical communication architecture is the SPMD (Single Program Multiple Data). This paradigm is focused on executing the same program in all processing elements but using different sets of tiles [9,10]. However, many SPMD applications share data between parallel processes and their communications can be a very big problem, especially when we have applications very well coupled, such as: application of finite differences, fluid dynamics, weather models, econometrics models, etc., all of which have to communicate tiles between MPI processes in each iteration. Hence, a SPMD tile is computed in a similar time due to the homogeneity of the core. However, the communication processes among neighbors are performed using different communications links depending on the location of the SPMD processes on the multi-core clusters. These behaviors may cause serious delays in tile synchronization when these applications are executed on multi-core clusters.

An example of this problem is illustrated in Fig. 1. The example shows us an SPMD application where each tile communicates with four tiles. As we can observe, this application needs to repeat a set of iterations but the iteration $i + 1$ depends on the results obtained in the iteration i. In this sense, the tiles are divided and assigned to each core to start the computation. Then, the computation processes have to wait until the slowest communications link finishes receiving its information to start the new iteration. These delays are due to the tile dependencies on the code. In some cases, the communication speed between MPI processes can vary in an order of magnitude for the same data packages depending on the link.

To solve these inefficiencies, we have developed a methodology which includes an analytical method that allows us to manage the communication latencies using some characteristics of each SPMD application over the parallel machine (e.g. computation and communication tile ratio). This method permits us to determine a relationship between scalability and efficiency. The objectives of this method are addressed in two analytically directions. The first one is to find the ideal number of cores needed to obtain the maximum speedup with a certain level of efficiency defined by the programmer (maximum strong scalability point). The second one is to determine how the application problem has to be increased in order to maintain an execution time constant while the number