Future Generation Computer Systems 37 (2014) 1–13

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

Two new fast heuristics for mapping parallel applications on cloud computing

I. De Falco, U. Scafuri, E. Tarantino *

Institute of High Performance Computing and Networking, National Research Council of Italy (ICAR-CNR), Via P. Castellino 111, 80131 Naples, Italy

HIGHLIGHTS

- The paper deal with the mapping problem.
- Specific reference is made to task interaction graph applications.
- Two new fast heuristics are proposed.
- These heuristics are improvements of the classical Min–min and Max–min algorithms.
- The results demonstrate the effectiveness of the proposed algorithms.

ARTICLE INFO

Article history:
Received 19 December 2012
Received in revised form
16 December 2013
Accepted 24 February 2014
Available online 6 March 2014

Keywords:
Cloud computing
Mapping
Communicating tasks
Heuristics

ABSTRACT

In this paper two new heuristics, named Min–min-C and Max–min-C, are proposed able to provide near-optimal solutions to the mapping of parallel applications, modeled as Task Interaction Graphs, on computational clouds. The aim of these heuristics is to determine mapping solutions which allow exploiting at best the available cloud resources to execute such applications concurrently with the other cloud services.

Differently from their originating Min–min and Max–min models, the two introduced heuristics take also communications into account. Their effectiveness is assessed on a set of artificial mapping problems differing in applications and in node working conditions. The analysis, carried out also by means of statistical tests, reveals the robustness of the two algorithms proposed in coping with the mapping of small- and medium-sized high performance computing applications on non-dedicated cloud nodes.

© 2014 Elsevier B.V. All rights reserved.

1. Introduction

The cloud paradigm [1–3], commercially supported by important firms as for instance Google [4], Amazon [5], and Microsoft [6], refers to the use of computing resources (hardware and software) delivered as a service over a network.

The current cloud systems offer on-demand a broad range of virtualized services which can be classified into three major models: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). IaaS refers to the practice of delivering on demand IT infrastructure as a commodity to customers. PaaS provides a development platform in which customers can create and execute their own applications. SaaS endows the user with an integrated service comprising hardware, development platforms, and applications. All these models allow users to have at their disposal the resources needed without having any knowledge about their numbers, characteristics, and location.

Although unsuitable to efficiently solve compute-intensive parallel applications, many efforts are being dedicated by manufacturers and scientists to provision the cloud systems with new functionalities. These capabilities allow executing in reasonable times small- and medium-sized high performance computing (HPC) applications [7–9].

For the contemporary cloud systems the customers who have to execute in parallel a multitask application can already negotiate, by an IaaS contract, the leasing of virtual machines for a prefixed time interval. Unfortunately such a leasing generally provides functionalities relatively to the networking, the data storage, the physical servers, and the virtualization software. The management of the operating system, of the middleware and of the runtime phase, the choice of the number of virtual machines with specific HW/SW characteristics, the mapping of the tasks on these virtual machines, and the scheduling are left to the customers.

An alternative is that the customers want to commit to a cloud middleware the management and the execution of their multitask

* Corresponding author. Tel.: +39 081 6139525; fax: +39 081 6139531.
E-mail address: ernesto.tarantino@na.icar.cnr.it (E. Tarantino).

http://dx.doi.org/10.1016/j.future.2014.02.019
0167-739X/© 2014 Elsevier B.V. All rights reserved.

applications concurrently with the current workload of the cloud system. This service can be negotiated by a new form of PaaS contract. To support this new contract the cloud middleware, in addition to mapping on the physical resources the virtual machines and all the other services requested by other customers, must also establish the optimal task/node deployment for the submitted multitask applications.

Unfortunately, most of the mapping tools described in scientific literature make reference to the allocation of independent tasks. Only a few of them can deal with applications with communicating tasks [10–13], yet they all can work only as long as applications are modeled as Direct Acyclic Graphs (DAGs) [14–16]. This kind of applications is much less general and flexible than multitask applications modeled as Task Interaction Graphs (TIGs) [17,18]. In the TIG model all the tasks are considered simultaneously executable and the communications can take place at any time, in general according to iterative and non-deterministic patterns. This means that there is no precedence relationship among tasks: each task cooperates with its neighbors [19].

In this paper we wish to move a first step towards filling this gap by introducing two mapping heuristics to efficiently allocate TIG applications on non-dedicated cloud resources. As shown in [20,21], given the NP-complete nature of the mapping, metaheuristic algorithms are the most appropriate to attain approximate solutions that meet the application requirements in a reasonable time [22–24].

In [25] a wide comparison among eleven heuristics is reported for allocation of independent tasks. The conclusions state that for the different situations, implementations, and parameter values used there, Genetic Algorithm (GA) consistently gave the best results. The average performance of the relatively simple Min–min [26] heuristic was always within 12% of the GA heuristic. Also Max–min [27] turns out to be quite effective and, as Min–min, has the advantages of not necessitating parameter tuning and of being much faster in terms of convergence times. Moreover, more recently, Luo et al. [28] have taken 20 different fast greedy heuristics into account when aiming to allocate independent tasks. Their results confirm that the classical Min–min performs very well, since it is within the three best algorithms.

Since literature assesses the robustness and the effectiveness of Min–min and Max–min for independent tasks, we have decided to improve them so that they can account for communication times too. It should be emphasized that these seminal algorithms take care of computations only, so communications among tasks are neglected whenever a node has to be chosen as the most suitable allocation for a given task. This is due to the sequential nature of these algorithms, which iteratively place one task at a time, so communication times cannot be considered if all the tasks have not yet been mapped.

To overcome this drawback a brand-new way of taking communications into account in algorithms such as Min–min and Max–min is presented. Namely, two new heuristics, Min–min-C and Max–min-C, are introduced. They are based on the classical Min–min and Max–min algorithms yet communications too are considered, and seem therefore very promising for an effective mapping of communicating tasks making up parallel applications modeled as TIGs.

The effectiveness of our two algorithms is evaluated by performing the mapping of artificial applications on a cloud infrastructure at different workload operating conditions, and is assessed against that of the two original algorithms.

Paper structure is as follows: Section 2 reports on the related research; Section 3 presents the working environment, while Section 4 explains our algorithms. In Section 5 the test problems experienced are reported, the results attained are discussed, and a statistical analysis is presented. Finally in Section 6 conclusions are given, and a discussion on the approach proposed and on the open problems to deal with is outlined.

2. Related research

The selection of the cloud resources that, on the basis of physical characteristics (computational power, frequency, memory, bandwidth, …) and load (known or estimated), better support the services as they are negotiated by the customers is nearly always a problem of considerable difficulty.

In [29,30] the mapping is performed manually through a laborious procedure whose results are particularly error-prone when the number of virtual machines reaches hundreds or thousands of nodes.

Fang et al. in [23] discuss a mapping mechanism related to independent tasks, based on the two levels of load balance, which considers the flexibility and virtualization in cloud computing. The first-level scheduling is from the users' application to the virtual machine, and creates the description of a virtual machine according to the resources and other configuration information demanded by the application tasks. The second level is from the virtual machine to host resources, and finds appropriate resources for the virtual machine in the host resources under certain rules, based on the description of the virtual machine for each task. The load of the virtual machine is evaluated by the predicted execution time of the tasks running on it.

In [31,32] approaches for a rule-based mapping which are able to automatically adapt the mapping between virtual machines and physical hosts' resources are advanced. The authors extended the open source solution Eucalyptus and the papers differ for the performance evaluation metrics of the chosen mapping policies: maximizing computation performance and virtual machine locality to achieve a high performance, and minimizing energy consumption in the first case, while the second approach includes the waiting time, the turnaround time, and the response time of the proposed algorithm.

Unfortunately the criteria and the mapping algorithms presented up to now cannot be exploited to efficiently execute TIG applications on cloud nodes. In fact, the mapping of this kind of applications introduces further degrees of complexity if the cloud resources, besides being heterogeneous and geographically dispersed, have features that vary even substantially over time as the local loads and the network bandwidth dynamically change [26,33]. The same holds for the classical mapping algorithms which, known as NP-complete already on traditional parallel and distributed systems, cannot work adequately in heterogeneous environments [34], such as clouds are.

Considered the NP-nature of the mapping problem, a metaheuristic algorithm has been investigated by Mehdi et al. [24]. To speed up the mapping process and ensure the fulfillment of all task deadlines and QoS requirements, the authors introduce a fast algorithm that can find a mapping using genetic algorithms with 'exist if satisfy' condition. Mapping time and makespan are the performance metrics that are used to evaluate the proposed system.

A very recent and interesting paper is [13], in which two greedy algorithms are used to generate the static allocation for the tasks composing a DAG application in a cloud. One is called Cloud List Scheduling (CLS), and the other Cloud Min–Min Scheduler (CMMS). Since the original Min–min algorithm does not consider the dependences among tasks, in CMMS the mappable task set must be updated in every scheduling step to maintain the task dependences. Namely, tasks in the mappable task set are the tasks whose predecessor tasks are all assigned. It is worth noting that this is a first step to extend the use of Min–min to interacting tasks, yet this modification affects DAG structures only, and cannot work for the here considered TIGs, which are a much more general structure.

The problem of allocating TIGs to computing systems is addressed in several papers but limited either to local area networks [35] or to clusters of PCs [36], or to grid environments relatively to dedicated nodes with advance reservation [37], and restricted to the task grouping [38].

Download English Version:

https://daneshyari.com/en/article/425860

Download Persian Version:

https://daneshyari.com/article/425860

Daneshyari.com