Contents lists available at ScienceDirect

Future Generation Computer Systems

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

A hybrid evolutionary algorithm for task scheduling and data assignment of data-intensive scientific workflows on clouds

Luan Teylo[a], Ubiratam de Paula[b], Yuri Frota[a], Daniel de Oliveira[a,*], Lúcia M.A. Drummond[a]

[a] *Institute of Computing, Fluminense Federal University, Niterói, Brazil*
[b] *UFRRJ – Federal Rural University of Rio de Janeiro, Rio de Janeiro, Brazil*

HIGHLIGHTS

- A new workflow model that considers tasks and data.
- The mathematical formulation of Task Scheduling and Data Assignment Problem.
- The design of a Hybrid Evolutionary Algorithm (HEA) for scheduling tasks and data.
- An extensive experimental evaluation, based on synthetic and real executions.

ARTICLE INFO

Article history:
Received 29 April 2016
Received in revised form 25 March 2017
Accepted 10 May 2017
Available online 18 May 2017

Keywords:
Clouds
Combinatorial optimization
Task scheduling
Data assignment
Hybrid evolutionary algorithm

ABSTRACT

A growing number of data- and compute-intensive experiments have been modeled as scientific workflows in the last decade. Meanwhile, clouds have emerged as a prominent environment to execute this type of workflows. In this scenario, the investigation of workflow scheduling strategies, aiming at reducing its execution times, became a top priority and a very popular research field. However, few work consider the problem of data file assignment when solving the task scheduling problem. Usually, a workflow is represented by a graph where nodes represent tasks and the scheduling problem consists in allocating tasks to machines to be executed at a predefined time aiming at reducing the makespan of the whole workflow. In this article, we show that the scheduling of scientific workflows can be improved when both task scheduling and the data file assignment problems are treated together. Thus, we propose a new workflow representation, where nodes of the workflow graph represent either tasks or data files, and define the Task Scheduling and Data Assignment Problem (TaSDAP), considering this new model. We formulated this problem as an integer programming problem. Moreover, a hybrid evolutionary algorithm for solving it, named HEA-TaSDAP, is also introduced. To evaluate our approach we conducted two types of experiments: theoretical and practical ones. At first, we compared HEA-TaSDAP with the solutions produced by the mathematical formulation and by other works from related literature. Then, we considered real executions in Amazon EC2 cloud using a real scientific workflow use case (SciPhy for phylogenetic analyses). In all experiments, HEA-TaSDAP outperformed the other classical approaches from the related literature, such as Min–Min and HEFT.

1. Introduction

The recent advances in computer science have allowed for several different fields of science to benefit from computational simulations in their experiments. These called *in silico* experiments [1,2] are consuming and producing an unprecedented volume of data to be further processed and analyzed, thus being considered as data-intensive experiments. This huge volume of data is found in experiments in many areas, for instance, phylogenetic analysis [3], computational fluid dynamics [4], astronomy [5], *etc.*. Thus, scientists perform their analyses using complex computational simulations and increasing volumes of data in their daily duties.

Most of these experiments are represented as a chaining of scientific programs, where the output of a specific program is the input of another program. In order to manage the execution of

* Corresponding author.
E-mail addresses: luanteylo@ic.uff.br (L. Teylo), upaula@ufrrj.br (U. de Paula), yuri@ic.uff.br (Y. Frota), danielcmo@ic.uff.br (D. de Oliveira), lucia@ic.uff.br (L.M.A. Drummond).

http://dx.doi.org/10.1016/j.future.2017.05.017
0167-739X/© 2017 Elsevier B.V. All rights reserved.

these complex experiments, scientific workflows can be a prominent solution. A scientific workflow is an abstraction that structures the steps of a scientific experiment as a graph of activities (*i.e.* scientific program invocations), in which nodes correspond to data processing activities and edges represent the dataflow among them [1]. Moreover, scientific workflows are commonly managed by complex software named Scientific Workflow Management Systems (SWfMS) that are used to define, execute, and monitor the data-intensive experiments. Well-known SWfMS are Swift/T [6], Pegasus [7], VisTrails [8], Apache Taverna [9] and Kepler [10].

In the same experiment, a scientific workflow is usually re-executed as many times as needed, varying the input dataset or the input parameter values to interpret the quality of the result produced by each execution of the workflow. This situation is well-known in parallel computing as parameter sweep [11] and it occurs when the same workflow (and its activities) is executed using different input data files (or a partition of the input data) and/or different configurations (different parameter values) until the exploration finishes (*i.e.* the analysis of the results is complete). Thus, since workflow activities commonly process big data, to explore data parallelism and parameter sweep we consider that each workflow activity may correspond to several executable tasks. Each task is considered the smallest unit of processing and may execute in parallel by consuming a different portion of the input data. Thus, in the context of this article, a task is the representation of an atomic execution of an activity, which processes a different set of parameter values, a data partition or chunk [12]. In addition, many of the data-intensive workflows are also compute-intensive, since a single task may execute for several hours or even days.

As the complexity of the scientific workflows grows in terms of exploration of thousands of huge datasets or several parameters, the performance requirements for such workflows have been pushing the envelope on the capacity of sequential systems (*e.g.* personal computers with a few of processors) for a while already. If executed sequentially, these data- and compute-intensive workflows could execute for several months, which is error-prone and not desirable due to the competition in science nowadays [13]. Thus, the demand for High Performance Computing (HPC) environments allied to parallelism techniques is extreme for these workflows to produce results in a feasible time for scientists.

Traditional HPC environments such as clusters, supercomputers and computational grids were used over the last decade to execute scientific workflows in parallel. However, in the last decade, Clouds [14] have emerged as a prominent environment to run data- and compute-intensive workflows [15]. Cloud computing is a type of Internet-based computing where virtually unlimited infrastructure, platform, and software are provided on demand and as services (*i.e.*, IaaS, PaaS and SaaS, respectively). Clouds follow a pay-*per*-use model [16,17] where users only pay for resources they actually used and for the time they used those resources. Virtual Machines (VMs) and storage areas are types of resources provided by clouds. Using clouds scientists are not required to acquire expensive infrastructure (such as a cluster) to execute their experiments neither spend much effort to configure a new infrastructure (as in a grid). To enable a data- and compute-intensive workflow execution in a cloud, the execution of each task has to be scheduled to a corresponding VM. Then, the scheduling problem is to decide where to execute all tasks. Scheduling tasks to distributed computing resources is an NP-complete problem [18], even when we consider simple scenarios. However, there are some characteristics of clouds that makes this scheduling process a little bit more complicated.

First of all, in clouds there are several options of VM types to be deployed. Each one with different processing and storage capacity, different bandwidth and financial cost. In addition, some of these VMs may not suitable for HPC (*e.g.* the micro and nano VMs in Amazon EC2 cloud). Thus, when we are executing workflows in parallel in the cloud, the deployed virtual cluster is commonly composed by heterogeneous VMs and this heterogeneity has to be considered in the scheduling approach. The second problem is data locality and transfer. Many of existing workflows consume and produce many GB or even TB of data and this data (or at least a partition of this data) has to be eventually transferred from one VM to another during the workflow execution. These data transfers can produce a huge (and negative) impact in the workflow execution. Let us consider the Montage workflow [5] as example. A simple execution of Montage may produce data files with several GBs. If, during a workflow execution, this data file is transferred several times from one VM to another, a considerable portion of the total workflow execution time will be spent only on data transfer instead of data processing (which is the focus of the experiment). Thus, when we are scheduling tasks of a workflow in the cloud we have to try to avoid unnecessary data transfers, or when the data transfers are necessary, we have to diminish the impact of data transfer in the total execution time of the workflow.

To exemplify this issue, let us consider the case where we have two tasks: $task_1$, which is a short term task (*i.e.* it is not compute-intensive) and $task_2$, which is a long term task (*i.e.* it is a compute-intensive task and requires high processing capacity to finish in a feasible time) in a workflow. In this example, $task_2$ consumes the data produced by $task_1$ (*i.e.* there is a data dependency). Let us also consider that $task_1$ was executed in a *medium* Vm1 of Amazon EC2 cloud and produces a data file with several GBs. To avoid data transfers, we face 2 possible scenarios: (i) to schedule $task_2$ to Vm2 (or Vm3) that have more processing capacity (such as a *2XLarge* VM in Amazon EC2 cloud), which will result in a costly data transfer from Vm1 to Vm2 (or Vm3) and (ii) to execute $task_2$ in Vm1 where $task_1$ was executed. The first scenario will imply into a costly data transfer, but then we can assume that $task_2$ will properly execute in a feasible time. The second scenario does not imply data transfers, but Vm1 may be not suitable to execute $task_2$, so the time needed to process $task_2$ in Vm1 can be huge. The scheduling approach has then to analyze the trade-off between transferring data and executing $task_2$ in Vm2 (or Vm3); or avoiding data transfer and executing $task_2$ in Vm1. In addition, when a data transfer is needed and defined by the scheduling approach, it can be performed by the SWfMS as a independent task of the workflow and executed before the task that will consume the data. This way, when the scheduled time to execute the task comes, all data will be already placed in the correct VM. Thus, it is clear that data distribution and task distribution are not independent problems and have to be analyzed together by the scheduling approach.

Finally, besides the impact of transfer data, we have also to consider a set of data constraints. These constraints usually define that some data cannot (or it is not recommended to) be moved (we call this as static data files), because it is either too big or for proprietary reasons (*e.g.* the Brazilian Internet law defines that all data produced by Brazilian federal universities and research centers should be stored in data-centers located in Brazil. All data cannot be moved or copied beyond the Brazilian frontier — actually this restriction motivated Amazon to create a data-center in the city of São Paulo). For example, if scientists are running phylogenetic analysis workflows [3] they commonly access the RefSeq database (www.ncbi.nlm.nih.gov/refseq/). This database is a static dataset, since it is unfeasible to transfer the entire dataset to the cloud (due to the huge volume of data). Thus, this requires that the scheduling approach "fixes" some tasks to specific VMs that execute as "near" as possible of static data files. Although these constraints "fix" some tasks to specific resources, they do not reduce the complexity of the workflow scheduling.

The aforementioned scenario leads to the development of several solutions for the workflow task scheduling problem in clouds [12,19–24]. However, these solutions do not consider data