Future Generation Computer Systems 64 (2016) 61–74

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

Workflow-and-Platform Aware task clustering for scientific workflow execution in Cloud environment

Jyoti Sahni, Deo Prakash Vidyarthi *

School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi-110067, India

HIGHLIGHTS

- Studies the effect of ineffective parallelism and system overheads on workflow execution in distributed environments.
- Establishes requirement of a task clustering technique which realizes maximum possible parallelism while minimizing system overheads and resource wastage.
- Proposes an autonomic Workflow-and-Platform Aware (WPA) task clustering technique.
- Evaluates the proposed method with state-of-art algorithms on four scientific workflows.

ARTICLE INFO

Article history:
Received 6 January 2016
Received in revised form
22 March 2016
Accepted 4 May 2016
Available online 25 May 2016

Keywords:
Scientific workflows
Coarse and fine grain tasks
Task clustering
Load consolidation
Performance analysis

ABSTRACT

A scientific workflow, usually consists of a good mix of fine and coarse computational granularity tasks displaying varied runtime requirements. It has been observed that fine grained tasks incur more scheduling overhead than their execution time, when executed on widely distributed platforms. Task clustering is extensively used, in such situations, as a runtime optimization method which involves combining multiple short duration tasks into a cluster, to be scheduled on a single resource. This helps in minimizing the scheduling overheads of the fine grained tasks. However, tasks grouping curtails the degree of parallelism and hence needs to be done optimally. Though a number of task clustering techniques have been developed to reduce the impact of system overheads, they fail to identify the appropriate number of clusters at each level of workflow in order to achieve maximum possible parallelism. This work proposes a level based autonomic Workflow-and-Platform Aware (WPA) task clustering technique which takes into consideration both; the workflow structure and the underlying resource set size for task clustering. It aims to achieve maximum possible parallelism among the tasks at a level of a workflow while minimizing the system overheads and resource wastage. A comparative study with current state of the art task clustering approaches on four well-known scientific workflows show that the proposed method significantly reduces the overall workflow execution time and at the same time is able to consolidate the load onto minimum possible resources.

1. Introduction

Research in diverse scientific domains (such as physics, bio-informatics, earth science and astronomy) often involves execution of complex large size applications for simulation and validation of the behaviour of different real-world activities. Many of such large scale scientific applications are usually constructed as workflows which consists of loosely coupled set of computational tasks linked through control and data dependencies [1]. The tasks comprising a workflow may vary in size with a few tasks requiring short run time (in tens of seconds), while some others may exhibit reasonably large run time requirements (in tens of minutes) [2,3]. Since comprehensively these tasks represent substantial amount of computation requirements, such complex workflows demand high-performance distributed computing environments where the tasks may be distributed amongst multiple computing nodes in order to complete their execution in a reasonable amount of time. Traditionally, developers of scientific applications have been using local workstations, supercomputers, clusters and grid platforms for running such workflows. A number of workflow management systems (WMSs) such as Pegasus [4], ASKALON [5], TAVERNA [6], Apache Tez [7] and Microsoft Dryad [8] have been designed

* Corresponding author. Tel.: +91 11 26704738.
E-mail addresses: jyoti92_scs@jnu.ac.in (J. Sahni), dpv@mail.jnu.ac.in (D.P. Vidyarthi).

http://dx.doi.org/10.1016/j.future.2016.05.008
0167-739X/

to facilitate execution of scientific workflows on distributed environments. Of late, Cloud computing has emerged as a utility-oriented distributed computing model which may be exploited for execution of huge and complex scientific applications. However, running these applications in widely distributed environments such as Cloud may involve significant system overheads [9]. This may adversely affect the overall execution time of the workflow. Further, execution of these applications in pay-as-you-go Cloud model without considering the effect of these delays may also result in additional financial overheads on the user. A cost and time effective workflow execution in distributed environment thus demands minimization of the overheads incurred.

The overheads imposed by WMSs and the execution environment for the execution of large scale scientific workflows can be classified into four major delays [9]: queue delay, workflow engine delay, job postscript delay, and data transfer delay. A number of optimization techniques such as task clustering [10], job throttling [11], data pre-staging [12] and over-provisioning [13] have been developed and employed to target different delay classes and to reduce their effect on the overall execution time of the workflow. Of these different optimization techniques, clustering techniques employed to increase the computational granularity of tasks by merging smaller tasks into a cluster based on certain clustering parameters are of particular interest. This helps in reducing the queue waiting time due to decreased number of tasks. However, tasks clustering limits the degree of parallelism and hence should be done optimally and judicially. Too few clusters limit parallelism opportunity while too many clusters result in increased system overheads often exceeding the ideal run time [14]. Although, a number of task clustering techniques have been developed for reducing the impact of system overheads on workflow execution in a distributed environment, they fail to identify a clustering which allows maximum possible parallelism among tasks at a level of a workflow while limiting the system overheads and resource wastage. This is because most of the existing clustering techniques depend upon the user to specify the number (and sometimes constitution) of clusters. This is not only cumbersome for the user but may also result in an inept clustering of tasks in case the users do not have a good knowledge of the workflow structure and/or the underlying distributed environment.

This work proposes an autonomic Workflow-and-Platform Aware (WPA) task clustering technique which takes into consideration both the workflow structure and the underlying resource set size to obtain a clustering that allows maximum possible parallelism among tasks at a level of a workflow while minimizing the system overheads and resource wastage. Two important points, considered in the proposed work, are as follows.

(a) *Limited resource wastage due to ineffective parallelism*: the algorithm ensures that the clustering of the parallel tasks at a level will result in limited resource wastage. For this, it is ascertained that the clustering allows execution of only those tasks in parallel that improves the overall run time of the workflow.
(b) *Minimum queuing overhead*: the algorithm ensures that the parallelism is sufficiently coarse so that the cost of managing parallelism is not that significant as the cost of doing parallel work.

Embraced with these features, the proposed approach not only helps in improving the run time performance of a workflow but also results in consolidating the load to minimum possible resources at each level in the workflow execution. Consequently, this helps in increasing the ability of the underlying resource set to accommodate more number of applications.

The rest of this paper is organized as follows. Section 2 discusses the related work. Section 3 provides an overview of the scientific workflow application model and the workflow execution architecture assumed in this work. Section 4 details the proposed clustering algorithm. Section 5 presents the simulation experiment done to evaluate the performance of the proposed work followed by the discussion on the results. Finally, Section 6 concludes the work.

2. Related work

Several research studies have been conducted in the past that analyses different system overheads involved in the execution of scientific workflow applications in a widely distributed environment such as grid and cloud [15,16,9]. Stratan et al. [15] proposed a methodology for testing grid workflow engines based on realistic, repeatable testing capable of characterizing five aspects: overhead, raw performance, stability, scalability and reliability. They noted that many times the main bottleneck in a busy system is the head node and therefore the head node resource consumption should not be ignored. Prodan et al. [16] presented a systematic and complete classification of overheads that occur while executing scientific workflows in a dynamic grid environment. Chen et al. [9] further extended the work of [16] by measuring the distribution and overlap of major overheads imposed by WMSs and execution environments. The existence of system overheads, as identified by these works, establishes the need for employing techniques such as task clustering for performance improvement.

Task clustering for performance improvement of scientific applications running in distributed platforms has been addressed in a number of research contributions. The objective is to group a number of *fine-grained* tasks into *coarse-grained* tasks and to run it on a single resource. This helps in saving the queuing overhead when resources are limited and in reducing the data transfer time when grouped tasks share some input data. However, tasks grouping may limit the degree of inherent parallelism and hence needs to be done optimally and judicially [14].

Several works have addressed task granularity control of bag-of-tasks (BOT) applications. Muthuvelu et al. [17] proposed a clustering algorithm that groups bag of tasks based on the processing time of tasks on the resources. The algorithm sorts the resources in decreasing order of their processing capacity (expressed in MIPS) and groups the tasks to a resource up to its resource capacity. This process is repeated until all the tasks are grouped and assigned to some resources. Ang et al. [18] extended the work of [17] and proposed a framework for bandwidth-aware job grouping based scheduling for improving dissemination of jobs in grid computing. Liu and Liao [19] proposed a grouping strategy based adaptive fine grained job scheduling algorithm which clusters jobs on the basis of resource characteristics. Later Muthuvelu et al. [20] extended their work in [17] and presented a parameterized job grouping strategy in which jobs are grouped based on their processing requirements, resource policies, network conditions and user's QoS requirements. More lately, in [21], Muthuvelu et al. proposed policies and approaches to decide granularity of a group of tasks based on the task processing requirements and network utilization constraints while satisfying user's QoS requirements (budget and deadline). These techniques though significantly reduce scheduling and queuing overheads, however, they do not consider dependencies between the tasks and thus are suitable only for applications with independent tasks.

Task clustering in scientific workflows has also been addressed in a number of research contributions. [22–24] studied the tasks clustering on an unbounded number of fully connected processors. Zomaya and Chan [25] proposed a genetic algorithm based task clustering algorithm for fixed number of processors. Since these works are targeted towards multiprocessor systems, they focus only on reducing the data communication delay and do not