Future Generation Computer Systems 46 (2015) 69–84

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

Using imbalance metrics to optimize task clustering in scientific workflow executions

Weiwei Chen [a,*], Rafael Ferreira da Silva [a], Ewa Deelman [a], Rizos Sakellariou [b]

[a] University of Southern California, Information Sciences Institute, Marina del Rey, CA, USA

[b] University of Manchester, School of Computer Science, Manchester, UK

HIGHLIGHTS

- Generalize the runtime imbalance and dependency imbalance problem in task clustering.
- Propose quantitative imbalance metrics to improve task clustering.
- Evaluate the imbalance metrics and balanced task clustering methods with five workflows.

ARTICLE INFO

Article history:
Received 15 February 2014
Received in revised form
13 September 2014
Accepted 19 September 2014
Available online 13 October 2014

Keywords:
Scientific workflows
Performance analysis
Scheduling
Workflow simulation
Task clustering
Load balancing

ABSTRACT

Scientific workflows can be composed of many fine computational granularity tasks. The runtime of these tasks may be shorter than the duration of system overheads, for example, when using multiple resources of a cloud infrastructure. Task clustering is a runtime optimization technique that merges multiple short running tasks into a single job such that the scheduling overhead is reduced and the overall runtime performance is improved. However, existing task clustering strategies only provide a coarse-grained approach that relies on an over-simplified workflow model. In this work, we examine the reasons that cause Runtime Imbalance and Dependency Imbalance in task clustering. Then, we propose quantitative metrics to evaluate the severity of the two imbalance problems. Furthermore, we propose a series of task balancing methods (horizontal and vertical) to address the load balance problem when performing task clustering for five widely used scientific workflows. Finally, we analyze the relationship between these metric values and the performance of proposed task balancing methods. A trace-based simulation shows that our methods can significantly decrease the runtime of workflow applications when compared to a baseline execution. We also compare the performance of our methods with two algorithms described in the literature.

1. Introduction

Many computational scientists develop and use large-scale, loosely-coupled applications that are often structured as scientific workflows. Although the majority of the tasks within these applications are often relatively short running (from a few seconds to a few minutes), in aggregate they represent a significant amount of computation and data [1,2]. When executing these applications in a multi-machine, distributed environment, such as the Grid or the Cloud, significant system overheads may exist and may slow-down the application execution [3]. To reduce the impact of such overheads, task clustering techniques [4–12] have been developed to group *fine-grained* tasks into *coarse-grained* tasks so that the number of computational activities is reduced and so that their computational granularity is increased. This reduced the (mostly scheduling related) system overheads. However, there are several challenges that have not yet been addressed.

A scientific workflow is typically represented as a directed acyclic graph (DAG). The nodes represent computations and the edges describe data and control dependencies between them. Tasks within a level (or depth within a workflow DAG) may have different runtimes. Proposed task clustering techniques that merge tasks within a level without considering the runtime variance may cause load imbalance, i.e., some clustered jobs may be composed of short running tasks while others of long running tasks. This

* Correspondence to: USC Information Sciences Institute, 4676 Admiralty Way Ste 1001, Marina del Rey, CA, 90292, USA. Tel.: +1 310 448 8408.

E-mail addresses: weiweich@usc.edu, wchenpublic@gmail.com, weiweich@acm.org (W. Chen), rafsilva@isi.edu (R. Ferreira da Silva), deelman@isi.edu (E. Deelman), rizos@cs.man.ac.uk (R. Sakellariou).

http://dx.doi.org/10.1016/j.future.2014.09.014
0167-739X/

imbalance delays the release of tasks from the next level of the workflow, penalizing the workflow execution with an overhead produced by the use of inappropriate task clustering strategies [13]. A common technique to handle load imbalance is overdecomposition [14]. This method decomposes computational work into *medium-grained* balanced tasks. Each task is coarse-grained enough to enable efficient execution and reduce scheduling overheads, while being fine-grained enough to expose significantly higher application-level parallelism than what is offered by the hardware.

Data dependencies between workflow tasks play an important role when clustering tasks within a level. A data dependency means that there is a data transfer between two tasks (output data for one and input data for the other). Grouping tasks without considering these dependencies may lead to data locality problems, where output data produced by parent tasks are poorly distributed. As a result, data transfer times and failure probabilities increase. Therefore, we claim that data dependencies of subsequent tasks should be considered.

We generalize these two challenges (Runtime Imbalance and Dependency Imbalance) to the general task clustering load balance problem. We introduce a series of balancing methods to address these challenges. However, there is a tradeoff between runtime and data dependency balancing. For instance, balancing runtime may aggravate the Dependency Imbalance problem, and vice versa. Therefore, we propose a series of quantitative metrics that reflect the internal structure (in terms of task runtimes and dependencies) of the workflow and use them as a criterion to select and balance the solutions.

In particular, we provide a novel approach to capture the imbalance metrics. Traditionally, there are two approaches to improve the performance of task clustering. The first one is a top-down approach [15] that represents the clustering problem as a global optimization problem and aims to minimize the overall workflow execution time. However, the complexity of solving such an optimization problem does not scale well since most solutions are based on genetic algorithms. The second one is a bottom-up approach [4,10] that only examines free tasks to be merged and optimizes the clustering results locally. In contrast, our work extends these approaches to consider the neighboring tasks including siblings, parents, and children, because such a family of tasks has strong connections between them.

The quantitative metrics and balancing methods were introduced and evaluated in [16] on five workflows. In this paper, we extend this previous work by studying:

- the performance gain of using our balancing methods over a baseline execution on a larger set of workflows;
- the performance gain over two additional task clustering methods described in the literature [9,10];
- the performance impact of the variation of the average data size and number of resources;
- the performance impact of combining our balancing methods with vertical clustering.

The rest of the paper is organized as follows. Section 2 gives an overview of the related work. Section 3 presents our workflow and execution environment models. Section 4 details our heuristics and algorithms for balancing. Section 5 reports experiments and results, and the paper closes with a discussion and conclusions.

2. Related work

System overhead analysis [17,18] is a topic of great interest in the distributed computing community. Stratan et al. [19] evaluate in a real-world environment Grid workflow engines including DAGMan/Condor and Karajan/Globus. Their methodology focuses on five system characteristics: overhead, raw performance, stability, scalability, and reliability. They point out that resource consumption in head nodes should not be ignored and that the main bottleneck in a busy system is often the head node. Prodan et al. [18] offered a complete Grid workflow overhead classification and a systematic measurement of overheads. In Chen et al. [3], we extended [18] by providing a measurement of major overheads imposed by workflow management systems and execution environments and analyzed how existing optimization techniques improve the workflow runtime by reducing or overlapping overheads. The prevalent existence of system overheads is an important reason task clustering provides significant performance improvement for workflow-based applications. In this paper, we aim to further improve the performance of task clustering under imbalanced load.

The low performance of *fine-grained* tasks is a common problem in widely distributed platforms where the scheduling overhead and queuing times at resources are high, such as Grid and Cloud systems. Several works have addressed the control of task granularity of bags of tasks. For instance, Muthuvelu et al. [4] proposed a clustering algorithm that groups bags of tasks based on their runtime—tasks are grouped up to the resource capacity. Later, they extended their work [5] to determine task granularity based on task file size, CPU time, and resource constraints. Recently, they proposed an online scheduling algorithm [6,7] that groups tasks based on resource network utilization, user's budget, and application deadline. Ng et al. [8] and Ang et al. [9] introduced bandwidth in the scheduling framework to enhance the performance of task scheduling. Longer tasks are assigned to resources with better bandwidth. Liu and Liao [10] proposed an adaptive fine-grained job scheduling algorithm to group fine-grained tasks according to processing capacity and bandwidth of the current available resources. Although these techniques significantly reduce the impact of scheduling and queuing time overhead, they do not consider data dependencies.

Task granularity control has also been addressed in scientific workflows. For instance, Singh et al. [11] proposed level- and label-based clustering. In level-based clustering, tasks at the same level of the workflow can be clustered together. The number of clusters or tasks per cluster is specified by the user. In the label-based clustering method, the user labels tasks that should be clustered together. Although their work considers data dependencies between workflow levels, it is done manually by the users, which is prone to errors and it is not scalable. Recently, Ferreira da Silva et al. [12,20] proposed task grouping and ungrouping algorithms to control workflow task granularity in a non-clairvoyant and online context, where none or few characteristics about the application or resources are known in advance. Their work significantly reduced scheduling and queuing time overheads, but did not consider data dependencies.

A plethora of balanced scheduling algorithms has been developed in the networking and operating system domains. Many of these schedulers have been extended to the hierarchical setting. Lifflander et al. [14] proposed to use work stealing and a hierarchical persistence-based rebalancing algorithm to address the imbalance problem in scheduling. Zheng et al. [21] presented an automatic hierarchical load balancing method that overcomes the scalability challenges of centralized schemes and poor solutions of traditional distributed schemes. There are other scheduling algorithms [22] that indirectly achieve load balancing of workflows through makespan minimization. However, the benefit that can be achieved through traditional scheduling optimization is limited by its complexity. The performance gain of task clustering is primarily determined by the ratio between system overheads and task runtime, which is more substantial in modern distributed systems such as Clouds and Grids.