Engineering Science and Technology, an International Journal 18 (2015) 256–269

HOSTED BY

Contents lists available at ScienceDirect

Engineering Science and Technology, an International Journal

journal homepage: http://www.elsevier.com/locate/jestch

Full length article

Adaptive workflow scheduling in grid computing based on dynamic resource availability

Ritu Garg*, Awadhesh Kumar Singh

Computer Engineering Department, National Institute Of Technology, Kurukshetra, Haryana, India

ARTICLE INFO

Article history:
Received 13 October 2014
Received in revised form
19 December 2014
Accepted 5 January 2015
Available online 4 February 2015

Keywords:
Grid computing
DAG grid workflow
Adaptive workflow scheduling
Re-scheduling
Resource monitoring

ABSTRACT

Grid computing enables large-scale resource sharing and collaboration for solving advanced science and engineering applications. Central to the grid computing is the scheduling of application tasks to the resources. Various strategies have been proposed, including static and dynamic strategies. The former schedules the tasks to resources before the actual execution time and later schedules them at the time of execution. Static scheduling performs better but it is not suitable for dynamic grid environment. The lack of dedicated resources and variations in their availability at run time has made this scheduling a great challenge. In this study, we proposed the adaptive approach to schedule workflow tasks (dependent tasks) to the dynamic grid resources based on rescheduling method. It deals with the heterogeneous dynamic grid environment, where the availability of computing nodes and links bandwidth fluctuations are inevitable due to existence of local load or load by other users. The proposed adaptive workflow scheduling (AWS) approach involves initial static scheduling, resource monitoring and rescheduling with the aim to achieve the minimum execution time for workflow application. The approach differs from other techniques in literature as it considers the changes in resources (hosts and links) availability and considers the impact of existing load over the grid resources. The simulation results using randomly generated task graphs and task graphs corresponding to real world problems (GE and FFT) demonstrates that the proposed algorithm is able to deal with fluctuations of resource availability and provides overall optimal performance.

1. Introduction

Recently, the rapid development of networking technology and web has led to the possibilities of using large number of geographically distributed heterogeneous resources owned by different organizations. These developments have led to the foundation of new paradigm known as *Grid Computing* [9,15]. Grid Computing is a type of parallel and distributed system that involves the integrated and collaborative use of resources depending on their availability and capability to satisfy the demands of researchers requiring large amount of communication and computation power to execute advanced science and engineering applications. Precedence constrained parallel applications (workflows) are one of the typical application models used in scientific and engineering fields requiring large amount of bandwidth and powerful computational resources. To achieve the promising potential of distributed resources, effective and efficient scheduling algorithm is important. The grid scheduler acts as an interface between user and distributed grid resources. The workflow scheduling in grid is one of the key challenges, which deals with assigning workflow tasks to the available grid resources while maintaining the task precedence (dependency) constraints and to meet the quality of service (QoS) demands of the user like minimizing the overall execution time.

In general, scheduling tasks on distributed grid resources belongs to a class of NP-hard problems [16]. So heuristics or approximations are the preferred options to obtain near optimal solutions. Many heuristics have been devoted to this problem as discussed in literature [2,7,21,30] considering that accurate prediction is available for computation cost and communication cost of resources. However, in real environment, it is difficult to accurately predict the values due to heterogeneous and dynamic

* Corresponding author.
E-mail addresses: ritu.59@gmail.com (R. Garg), aksinreck@rediffmail.com (A.K. Singh).
Peer review under responsibility of Karabuk University.

http://dx.doi.org/10.1016/j.jestch.2015.01.001

characteristics of the grid environment. The dynamicity of the grid resources is due to both the network connectivity and computational nodes.

As the grid resources are not dedicated, and can be used by the other users simultaneously, which leads to load variations on resources. The local tasks have more priority and handled first in comparison to grid tasks. The fluctuations in the resource availability (computing speed of the host and links bandwidth) due to resource's local loads and competition from other users cause the original schedule to become sub optimal. If for instance, load of the processor increases, the execution time of the tasks assigned to it will increase. Further, sudden increase in link load, increases the data transfer time between computers where dependent tasks resides. In case of grid applications especially for long running jobs (days or weeks), the performance degradation caused by load over resources is unacceptable. It leads to the necessity of relocating the tasks to other resources. Hence, it is a key challenge to maintain an application performance during its execution, if their resources suddenly receive high workload.

In order to ensure high performance in dynamic and unreliable grid environment, we considered the adaptive scheduling [19] where scheduling policy change dynamically as per the previous and current behavior of the system to cope with the variations in the resource availability. Here, initial scheduling of all the tasks is performed statically and then rescheduling of unexecuted tasks is performed when required. The ability to discover and monitor the status of resources at run time is fundamental for the adaptive operation of the grid here.

In this paper, we proposed a novel Adaptive Workflow Scheduling (AWS) algorithm for grid applications consisting of workflow tasks (dependent tasks) to meet the performance requirements based on QoS information like availability along with the accessibility of the resources as indicated by service level agreement (SLA). It considers the processors (computing speed) and network links (bandwidth) availability by monitoring the load over non-dedicated grid processing nodes and network links. The procedure involves static task scheduling, periodic resource monitoring and rescheduling the remaining unexecuted tasks in order to deal with changes/fluctuations occurring at run time and to achieve minimum execution time (makespan) of the workflow grid application. The procedure of proposed AWS differs from other approaches in literature by considering the dynamic availability of resources, both computing nodes and communication links due to existence of local load or load by other users. It considers (i) Degradation of resource performances especially computing speed of nodes and network links bandwidth as per SLA, as a source for triggering rescheduling. (ii) Evaluate the benefits of rescheduling considering cost of reevaluating the schedule and overhead due to transfer of data. (iii) Availability of newly added resources.

The AWS algorithm is efficient one as it achieves minimum execution time of the application with the help of rescheduling the computation away from: overloaded computing nodes, nodes with overloaded communication links that can slow down the computation and it also considers the addition of new nodes to increase the performance of the application. Further, algorithm also provides load balancing by supporting rescheduling of tasks form overloaded resources.

The rest of the paper is organized as follows. We briefly mention the related work in Section 2. Section 3, describes the mathematical model, including the resource model, task model along with the problem definition. Thereafter, in Section 4, we explain the proposed Adaptive workflow Scheduling algorithm. Section 5 includes the pseudo code of the algorithm and detailed example. Section 6 discusses the simulation and result analysis. Finally section 7, gives the conclusion.

2. Related work

The problem of scheduling in grid, for workflow (DAG-based) tasks has already been addressed in the literature. Most of the related work attempt to achieve the minimum execution time (makespan) on heterogeneous grid environment. To schedule scientific workflow applications, Heterogeneous Earliest Finish Time (HEFT) [29], is the most popular list based heuristic. It orders the workflow tasks based on priorities and then assign them to suitable resources to achieve high performance. Similarly another list based heuristics Min–min, Max–Min [22], Critical-Path-on-a-processor (CPOP) [29] are studied to achieve high performance. The PCH algorithm [5] uses a hybrid clustering-list-scheduling strategy, where tasks with heavy communication cost are grouped together and assigned to the same resource in a cluster. It aims to reduce the schedule length by reducing the communication cost. Further, the paper [14] describes the design, development and evolution of the Pegasus Workflow Management System, which maps abstract workflow descriptions onto distributed computing infrastructures in order to achieve reliable and scalable workflow execution.

The critical issue in list heuristics for DAG scheduling is the accurate prediction for both the computation and the communication costs. However, in a real grid environment, system is less reliable and more dynamic: individual resource capability varies over time due to internal or external factors, thus, it is difficult to estimate these values accurately. To deal with resource dynamicity, two types of approaches are proposed in literature: dynamic scheduling and adaptive scheduling. In dynamic scheduling, all tasks are scheduled at run time while in adaptive scheduling; an advance static schedule is generated using available estimates and schedules responds to changes at run time with the help of rescheduling. GrADS [4] is the typical rescheduling mechanism which schedules workflow grid tasks based on three popular heuristics of Min–min, Max-min and Suffrage. It focuses on iterative workflows, allowing system to perform rescheduling at each iteration. Rescheduling is activated by contract violation between user and resource provider. If the performance is expected by migration, then unexecuted jobs are migrated to new mapped resources.

Other rescheduling methods proposed are AHEFT [31] and SLACK [26]. In AHEFT, author proposed an adaptive rescheduling algorithm based on static strategy. Here the workflow planner adapts to grid dynamics with the help of run time executor. Rescheduling is performed on the basis of FEA, which is the earliest time when output file is available for dependent tasks, if performance increase is there. While in SLACK [26] it is using the concept of spare time, which does not affect the schedule length of the workflow. If execution time of task goes beyond the spare time then only selective rescheduling event is triggered. The major drawback of these studies is that rescheduling is performed on periodic basis on performance degradation. Moreover, during rescheduling initial information of grid resources is used, without reflecting dynamically updated information.

Another approach to deal with dynamic performance changes of grid resources is proposed in [6] which uses the path clustering heuristic (PCH) to generate the initial schedule and then round based approach is used to reschedule. On each round some of the tasks based on a criterion are sent to execute and then performance of the resource is measured in that round. If performance is below the threshold value, then the algorithm reschedules the non executed tasks. In [11] the author proposed the adaptive list scheduling service algorithm (ALSS) for workflow tasks in order to deal with dynamic nature of service oriented grid environment. Low overhead rescheduling is considered only for services on the