Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 108C (2017) 1394–1403

Procedia Computer Science

International Conference on Computational Science, ICCS 2017, 12-14 June 2017, Zurich, Switzerland

Job-flow Anticipation Scheduling in Grid

Victor Toporkov, Dmitry Yemelyanov, and Alexander Bobchenkov

National Research University "MPEI"
ul. Krasnokazarmennaya, 14, Moscow, 111250, Russia
{ToporkovVV, YemelyanovDM, BobchenkovAV}@mpei.ru

Abstract

In this paper, a heuristic user job-flow scheduling approach for Grid virtual organizations with non-dedicated resources is discussed. Users' and resource providers' preferences, virtual organization's internal policies, resources geographical distribution along with local private utilization impose specific requirements for efficient scheduling according to different, usually contradictive, criteria. With increasing resources utilization level the available resources set and corresponding decision space are reduced. This further complicates the task of efficient scheduling. In order to improve overall scheduling efficiency we propose a heuristic anticipation scheduling approach. Initially it generates a near optimal but infeasible scheduling solution which is then used as a reference for efficient resources allocation.

Peer-review under responsibility of the scientific committee of the International Conference on Computational Science

Keywords: scheduling, Grid, resources, utilization, heuristic, job batch, virtual organization, anticipation

1 Introduction and Related Works

In distributed environments with non-dedicated resources such as utility Grids the computational nodes are usually partly utilized by local high-priority jobs coming from resource owners. Thus, the resources available for use are represented with a set of slots - time intervals during which the individual computational nodes are capable to execute parts of independent users' parallel jobs. These slots generally have different start and finish times and a performance difference. The presence of a set of slots impedes the problem of coordinated selection of the resources necessary to execute the job flow from computational environment users. Resource fragmentation also results in a decrease of the total computing environment utilization level [4, 15].

Two established trends may be outlined among diverse approaches to distributed computing. The first one is based on the available resources utilization and application level scheduling [2]. As a rule, this approach does not imply any global resource sharing or allocation policy. Another trend is related to the formation of user's virtual organizations (VO) and job flow scheduling [8, 10]. In this case a metascheduler is an intermediate chain between the users and local resource management and job batch processing systems.

1877-0509
Peer-review under responsibility of the scientific committee of the International Conference on Computational Science
10.1016/j.procs.2017.05.011

Uniform rules of resource sharing and consumption, in particular based on economic models, make it possible to improve the job-flow level scheduling and resource distribution efficiency. VO policy may offer optimized scheduling to satisfy both users' and VO common preferences. The VO scheduling problems may be formulated as follows: to optimize users' criteria or utility function for selected jobs [5, 11], to keep resource overall load balance [18], to have job run in strict order or maintain job priorities [9], to optimize overall scheduling performance by some custom criteria [1, 13], etc.

VO formation and performance largely depends on mutually beneficial collaboration between all the related stakeholders. However, users' preferences and VO common preferences (owners' and administrators' combined) may conflict with each other. Users are likely to be interested in the fastest possible running time for their jobs with least possible costs whereas VO preferences are usually directed to available resources load balancing or node owners' profit boosting. Thus, VO policies in general should respect all members and the most important aspect of rules suggested by VO is their fairness.

A number of works understand fairness as it is defined in the theory of cooperative games, such as fair quotas [3, 7], fair user jobs prioritization [9], non-monetary distribution [12]. The cyclic scheduling scheme (CSS) [17] implements a fair scheduling optimization mechanism which ensures stakeholders interests to some predefined extent.

The downside of a majority centralized metascheduling approaches is that they loose their efficiency and optimization features in distributed environments with a limited resources supply. For example, in [15] a traditional backfilling algorithm provides better scheduling outcome when compared to different optimization approaches in resource domain with a minimal performance configuration. The general root cause is that in fact the same scarce set of resources (being efficient or not) have to be used for a job flow execution or otherwise some jobs might hang in the queue. And under such conditions user jobs priority and ordering greatly influence the scheduling results. At the same time application-level brokers are still able to ensure user preferences and optimize the job's performance under free-market mechanisms.

A main contribution of this paper is a heuristic CSS-based job-flow scheduling approach which retains optimization features and efficiency even in distributed computing environments with limited resources. The rest of the paper is organized as follows. Section 2 presents a general CSS fair scheduling concept. The proposed heuristic-based scheduling technique is presented in Section 3. Section 4 contains simulation experiment setup and results for the proposed scheduling approach. Finally, Section 5 summarizes the paper.

2 Cyclic Alternative-Based Fair Scheduling Model and Limited Resources

Scheduling of a job flow using CSS is performed in time cycles known as scheduling intervals, by job batches [17]. The actual scheduling procedure consists of two main steps. The first step involves a search for alternative scenarios of each job execution or simply alternatives [14]. During the second step the dynamic programming methods [17] are used to choose an optimal alternatives' combination (one alternative is selected for each job) with respect to the given VO and user criteria. This combination represents the final schedule based on current data on resources load and possible alternative executions.

An example for a user scheduling criterion stated problem may be a minimization of overall job running time, a minimization of overall running cost, etc. This criterion describes user's preferences for that specific job execution and expresses a type of an additional optimization to