Future Generation Computer Systems 26 (2010) 608–621

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

Computational models and heuristic methods for Grid scheduling problems

Fatos Xhafa[a,*,1], Ajith Abraham[b]

[a] Department of Computer Science and Information Systems, Birkbeck, University of London, UK

[b] Machine Intelligence Research Labs (MIRLabs), Scientific Network for Innovation and Research Excellence, USA

ARTICLE INFO

Article history:
Received 24 August 2009
Received in revised form
20 November 2009
Accepted 29 November 2009
Available online 4 December 2009

Keywords:
Grid computing
Scheduling
Independent scheduling
Multi-objective optimization
Heuristics
Meta-heuristics

ABSTRACT

In this paper we survey computational models for Grid scheduling problems and their resolution using heuristic and meta-heuristic approaches. Scheduling problems are at the heart of any Grid-like computational system. Different types of scheduling based on different criteria, such as static versus dynamic environment, multi-objectivity, adaptivity, etc., are identified. Then, heuristic and meta-heuristic methods for scheduling in Grids are presented. The paper reveals the complexity of the scheduling problem in Computational Grids when compared to scheduling in classical parallel and distributed systems and shows the usefulness of heuristic and meta-heuristic approaches for the design of efficient Grid schedulers. We also discuss on requirements for a modular Grid scheduling and its integration with Grid architecture.

1. Introduction

Grid computing and Grid technologies primarily emerged for satisfying the increasing demand of the scientific computing community for more computing power. Geographically distributed computers, linked through the Internet in a Grid-like manner, are used to create virtual supercomputers of vast amount of computing capacity able to solve complex problems from e-Science in less time than known before. Thus, within the last years we have witnessed how Grid computing has helped to achieve breakthroughs in meteorology, physics, medicine and other computing-intensive fields. Examples of such large-scale applications are known from optimization [1–3], Collaborative/e-Science Computing [4,5] and Data-Intensive Computing [6], to name a few.

Grid computing is still in the development stage, and many challenges are to be addressed. Among these, improving its efficiency is a key issue. The question is: *How do we make use of a large number of computers worldwide, ranging from simple laptops, to clusters of computers and supercomputers connected through heterogenous networks in an efficient, secure and reliable manner?*

For the majority of Grid systems, scheduling is a very important mechanism. In the simplest of cases, scheduling of jobs can be done in a blind way by simply assigning the incoming tasks to the available compatible resources. Nevertheless, it is a lot more profitable to use more advanced and sophisticated schedulers. Moreover, the schedulers would generally be expected to react to the dynamics of the Grid system, typically by evaluating the present load of the resources, and notifying when new resources join or drop from the system. Additionally, schedulers can be organized in a hierarchical form or can be distributed in order to deal with the large scale of the Grid.

An important issue here is how to formally define the Grid scheduling problem. In this paper we present the most important and useful computational models for this purpose. Then, we focus on the design of efficient Grid schedulers using heuristic and meta-heuristic methods. Heuristic and meta-heuristic methods have proven to be efficient in solving many computationally hard problems. They are showing their usefulness also in the Grid computing domain, especially for scheduling and resource allocation. We analyze why heuristic and meta-heuristic methods are good alternatives to more traditional scheduling techniques and what make them appropriate for Grid scheduling.

The rest of the paper is organized as follows. We present in Section 2 a few important concepts from Grid computing, and introduce a few types of Grids in view of the needs for different types of scheduling and resource allocation. Then, in Section 3 we identify different types of scheduling problems arising in Grid systems. Computational models for Grid scheduling are given in Section 4, while in Section 5 we focus on the current state of

* Corresponding address: Department of Computer Science and Information Systems, Birkbeck, University of London, Malet Street, London WC1E 7HX, UK. Tel.: +44 0 20 7763 2105; fax: +44 0 20 7631 6727.

E-mail addresses: fatos@dcs.bbk.ac.uk (F. Xhafa), ajith.abraham@ieee.org (A. Abraham).

1 On leave from Technical University of Catalonia, Spain.

0167-739X/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2009.11.005

using heuristic and meta-heuristic methods for solving scheduling problems in Grid systems. The integration of Grid schedulers into Grid architecture is tackled in Section 6. A few other issues such as security and Grid services scheduling are discussed in Section 7. We end the paper in Section 8 with some conclusions.

2. The *many* Grids

The roots of Grid computing can be traced back to the late 1980s and the first concept that laid the basis of today's Grid systems were developed by researchers from distributed supercomputing for numerical and optimization systems. By the late 1990s, the terms Computational Grids and Grid Computing were popularized by Foster et al. [7], who developed the Globus Toolkit as a general middleware for Grid Systems. Since then, Grid systems have advanced very quickly. In the following subsections we briefly review the most important types of Grids that have appeared during recent years.

Computational Grids. Computational Grids are among the first type of Grid systems. They were developed due to the need to solve problems that require processing a large quantity of operations or data. In spite of the fact that the capacity of the computers continues to improve, the computational resources do not respond to the continuous demand for more computational power. Moreover, many statistical studies have shown that computers from companies, administration, etc. are usually underutilized. One of the main objectives of the Computational Grid is, therefore, to benefit from the existence of many computational resources through the sharing.

Scavenging Grids. In such Grids, the politics of "*scavenging*" is applied, according to which, each time a machine remains idle, it reports its state to the Grid node responsible for the management and planning of the resources. Then, this node usually assigns to the idle machine the next pending task that can be executed in that machine. Scavenging normally hinders the owner of the application, since in the event that the idle machine changes its state to be busy with tasks not coming from the Grid system, the application is suspended or delayed. This situation would create completion times not predictable for Grid-based applications. Sethi@home project is an example of scavenging Grids.

e-Science Grids. Under the name of e-Science Grids are known types of Grids that are primarily devoted to the solution of problems from science and engineering. Such Grids give support to the computational infrastructure (access to computational and data resources) needed to solve many complex problems arising in areas of science and engineering. Representative examples are EGEE Grid Computing, UK e-Science Grid, German D-Grid, BIG GRID (the Dutch e-Science Grid) and French Grid'5000, among others.

Data Grids. Data Grids primarily deal with data repositories, sharing, access and management of large amounts of distributed data. Many scientific and engineering applications require access to large amounts of distributed data; however, different data could have their own format. In such Grid systems many types of algorithm, such as replication, are important to increase the performance of Grid-enabled applications that use large amounts of data. Also, data copy and transfer is important here in order to achieve high throughput.

Enterprise Grids. Nowadays Grid computing is becoming a significant component of business as well. Indeed, today's e-business must be able to respond to increasing consumer demands and adjust dynamically and efficiently to marketplace shifts. Enterprise Grids enable running several projects within one large enterprise to share resources in a transparent way. Enterprise Grids are thus showing great and innovative changes on how computing is used. The Grid offers a large potential to solving business problems by facilitating global access to enterprise computing services and data. Examples of Enterprise Grids are "Sun Grid Engine", "IBM Grid", "Oracle Grid" and "HP Grid".

Desktop Grids. A new form of Enterprise Grids is also emerging in institutions, the so-called Desktop Grids, which use the idle cycles of desktop PCs. Small enterprises and institutions usually are equipped with hundreds or thousands of desktops, mainly used for office tasks. This amount of PCs is thus a good source for setting up a Grid system for the institution. In this case, the particularity of the Grid system is its unique administrative domain, which makes it easier to manage due to the low heterogeneity and volatility of resources. Of course, the desktop Grid can cross many administrative domains and in this case the heterogeneity and volatility of the resources is an issue, as in a general Grid system setting.

3. Scheduling problems in Grid systems

Rather than a problem, scheduling in Grid systems is a family of problems. This is due to the many parameters that intervene scheduling as well as due to the different needs of Grid-enabled applications. In the following, we give some basic concepts of scheduling in Grid systems and identify the most common scheduling types. Needless to say, job scheduling in its different forms is computationally hard; it has been shown that the problem of finding optimum scheduling in heterogeneous systems is in general NP-hard [8].

3.1. Basic concepts and terminology

Although many types of resources can be shared and used in a Grid system, usually they are accessed through an *application* running in the Grid. Normally, an application is used to define the piece of work of higher level in the Grid. A typical Grid scenario is as follows: an application can generate several jobs, which in turn can be composed of subtasks; the Grid system is responsible for sending each subtask to a resource to be solved. In a simpler Grid scenario, it is the user who selects the most adequate machine to execute its application or subtasks. However, in general, Grid systems must dispose of *schedulers* that automatically and efficiently find the most appropriate machines to execute an assembly of tasks.

3.1.1. New characteristics of scheduling in Grids

The scheduling problem is one of the most studied problems in the optimization research community. However, in the Grid setting there are several characteristics that make the problem different and more challenging than its version of conventional distributed systems. Some of these characteristics are the following.

- *The dynamic structure* of the Computational Grid. Unlike traditional distributed systems, resources in a Grid system can join or leave the Grid in an unpredictable way. This could be simply due to losing connection to the system or because their owners switch off the machine or change the operating system, etc. Given that the resources cross different administrative domains, there is no control over the resources.
- The *high heterogeneity of resources*. In Grid systems, computational resources could be very disparate in their computing capacity, ranging from laptops, desktops, clusters, supercomputers and even small computational devices. Current Grid infrastructures are not yet much versatile but heterogeneity is among most important features in any Grid system.
- The *high heterogeneity of jobs*. Jobs arriving at any Grid system are diverse and heterogenous in terms of their computational