Future Generation Computer Systems 28 (2012) 1058–1069

Contents lists available at SciVerse ScienceDirect

Future Generation Computer Systems

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

A GridWay-based autonomic network-aware metascheduler

Luis Tomás [a,*], Agustín C. Caminero [b], Omer Rana [c], Carmen Carrión [a], Blanca Caminero [a]

[a] Department of Computing Systems, The University of Castilla-La Mancha, Albacete, Spain

[b] Department of Communication and Control Systems, The National University of Distance Education, Madrid, Spain

[c] Cardiff School of Computer Science, 5 The Parade, Cardiff, CF24 3AA, United Kingdom

ARTICLE INFO

Article history:
Received 5 November 2010
Received in revised form
6 June 2011
Accepted 21 August 2011
Available online 31 October 2011

Keywords:
Grid computing
Autonomic systems
GridWay
Quality of service

ABSTRACT

One of the key motivations of computational and data grids is the ability to make coordinated use of heterogeneous computing resources which are geographically dispersed. Consequently, the performance of the network linking all the resources present in a grid has a significant impact on the performance of an application. It is therefore essential to consider network characteristics when carrying out tasks such as scheduling, migration or monitoring of jobs. This work focuses on an implementation of an autonomic network-aware meta-scheduling architecture that is capable of adapting its behavior to the current status of the environment, so that jobs can be efficiently mapped to computing resources. The implementation extends the widely used GridWay meta-scheduler and relies on exponential smoothing to predict the execution and transfer times of jobs. An autonomic control loop (which takes account of CPU use and network capability) is used to alter job admission and resource selection criteria to improve overall job completion times and throughput. The implementation has been tested using a real testbed involving heterogeneous computing resources distributed across different national organizations.

1. Introduction

Computational and data grids allow the coordinated use of heterogeneous computing resources within large-scale parallel applications in science, engineering and commerce [1]. Since organizations sharing their resources in such a context still keep their independence and autonomy [2], grids are highly variable systems in which resources may join/leave the system at any time. This variability makes Quality of Service (*QoS*) highly desirable, though often very difficult to achieve in practice. One reason for this limitation is the lack of a central entity that orchestrates the entire system. This is especially true in the case of the network that connects the various components of a grid system.

Achieving an *end-to-end* QoS is often difficult, as without resource reservation any guarantees on QoS are often hard to achieve. Furthermore, in a real grid system, reservations may not be always feasible, since not all the Local Resource Management Systems (LRMS) permit them. There are also other types of resource properties, such as bandwidth, which lack a global management entity, thereby making their reservation impossible.

However, for applications that need a timely response (i.e., distributed engine diagnostics [3] or collaborative visualization [4]), the grid must provide users with some assurance about the use of resources—a non-trivial subject when viewed in the context of network QoS. In a grid, entities communicate with each other using an interconnection network—resulting in the network playing an essential role in grid systems [5].

In a previous contribution [6], authors proposed an autonomic network-aware grid meta-scheduling architecture as a possible solution. This architecture takes into account the status of the system in order to make meta-scheduling decisions—paying special attention to network capability. This is a modular architecture in which each module works independently of others, thereby providing an architecture that can be adapted to new requirements easily. In this paper, the aforementioned architecture has been implemented as an extension to the GridWay meta-scheduler [7], with case studies and performance results provided to demonstrate how it can used. A scheduling technique that makes use of Exponential Smoothing (ES) [8] to calculate predictions on the completion times of jobs is also provided. Thus, the main contributions of this paper are: (1) an implementation of an architecture to perform autonomic network-aware meta-scheduling based on the widely used GridWay system; (2) a scheduling technique that relies on ES to predict the completion times of jobs; (3) a performance evaluation carried out using a testbed involving workloads and heterogeneous resources from several organizations.

* Correspondence to: Instituto de Investigación en Informática de Albacete (I3A), Universidad de Castilla-La Mancha, Campus Universitario s/n 02071, Spain. Tel.: +34 967 599 200x2696; fax: +34 967 599 343.

E-mail addresses: luistb@dsi.uclm.es (L. Tomás), accaminero@scc.uned.es (A.C. Caminero), o.f.rana@cs.cardiff.ac.uk (O. Rana), carmen@dsi.uclm.es (C. Carrión), mariablanca.caminero@uclm.es (B. Caminero).

0167-739X/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2011.08.019

The structure of the paper is as follows: Section 2 reviews existing approaches for supporting QoS in grids, along with grid meta-schedulers. Section 3 discusses a scenario in which an autonomic scheduler can be used and harnessed. Section 4 contains details about the implementation based on an extension to the GridWay meta-scheduler. Section 5 presents a performance evaluation of our approach, with conclusions and suggestions for further work identified in Section 6.

2. Related work

The provision of QoS in a grid system has been explored by a number of research projects, such as GARA [5], G-QoSM [9], GNRB [10–12], among others. The proposals which provide scheduling of users' jobs to computing resources are GARA and G-QoSM, and the schedulers used are DSRT [13] and PBS [14] in GARA, whilst G-QoSM uses DSRT. These schedulers (DSRT and PBS) only pay attention to the load of the computing resource, thus a powerful unloaded computing resource with an overloaded network could be chosen to run jobs, which decreases the performance received by users, especially when the job requires a high network I/O.

A network is a key component within a grid system due to the coordinated use of distributed resources, thus attention should be paid when carrying out tasks such as scheduling, migration, or monitoring [15]. Surprisingly, many of the above efforts do not take network capability into account when scheduling tasks, thus other meta-schedulers available today, such as GRMS [16], CSF [17], VIOLA [18], Grid Service Broker [19], Grid Network Broker (GNB) [6] or GridWay [7] need to be analyzed.

The main drawback of GRMS is that it is obsolete since the last Globus version it deals with is 2.4 [20]. CSF provides reservations based on a number of features, including the network, but it is a centralized engine and is not intended for bulk data transfer, rather it primarily tackles scheduling heterogeneities [21]. VIOLA provides co-allocation support for both computational and network resources, but the focus in VIOLA is on co-allocation and reservation—which is not always possible if the network is under ownership of a different administrator. Grid Service Broker already includes network information provided by Network Weather Service (NWS) [22] to perform the meta-scheduling, but it requires information on the effective bandwidth between all the data hosts and all the compute hosts in the system to perform network-aware scheduling, which makes the proposal difficult to scale. GNB [6] is an autonomic network-aware metascheduling framework, but it is only tested by means of simulations. GridWay is straight-forward to install and use. Besides, one of the main goals of GridWay is its modular architecture that allows extensions to be implemented easily. However, the network is not among the parameters it uses to perform meta-scheduling.

GridWay has been chosen as the basis for the implementation described in this paper due to the reasons provided above. However, it also has been necessary to make GridWay network-aware in order to fully support the devised techniques.

On the other hand, conceptually, an autonomic system requires: (a) sensor channels to sense the changes in the internal state of a system and the external environment in which the system is situated, and (b) motor channels to react to and counter the effects of the changes in the environment by changing the system and maintaining equilibrium. The changes sensed by the sensor channels have to be analyzed to determine if any of the essential variables have gone out of their viability limits. If so, it has to trigger some kind of planning to determine what changes to inject into the current behavior of the system such that it returns to the equilibrium state within the new environment. This planning requires knowledge to select the right behavior from a large set of possible behaviors to counter the change. Finally, the motor neurons execute the selected change. Sensing, Analyzing, Planning, Knowledge and Execution are thus the keywords used to identify an autonomic computing system. A common model based on these ideas was identified by IBM Research and defined as *MAPE* (Monitor-Analyze-Plan-Execute) [23]. There are, however, a number of other models for autonomic computing [24,25]—in addition to work in the agent-based systems community that share commonalities with the ideas presented above. There has been significant work already undertaken toward autonomic grid computing [26–29]. In [26], an architecture to achieve automated control and management of networked applications and their infrastructure based on XML format specification is presented. Liu and Parashar [27] present an environment that supports the development of self-managed autonomic components, with the dynamic and opportunistic composition of these components using high-level policies to realize autonomic applications, and provide runtime services for policy definition, deployment and execution. In [28], an autonomic job scheduling policy for grid systems is presented. This policy can deal with the failure of computing resources and network links, but it does not take the network into account in order to decide which computing resource will run each user application. Only idle/busy periods of computing resources are used to support scheduling. Finally, in [29], a simple but effective policy was formulated, which prioritized the finishing and acceptance of jobs over their response time and throughput. It was determined that due to the dynamic nature of the problem, it could be best resolved by adding self-managing capabilities to the middleware. Using the new policy, a prototype of an autonomous system was built and succeeded in allowing more jobs to be accepted and finished correctly.

3. Autonomic Network-aware Meta-scheduling (ANM)

The availability of resources within a grid environment may vary over time—some resources may fail whereas others may join or leave the system at any time. Additionally, each grid resource must execute a workload that combines locally generated tasks with those that have been submitted from external (remote) user applications. Hence, each new task influences the execution of existing applications, requiring a resource selection strategy that can account for this dynamism within the system. Our autonomic approach uses a variety of parameters to make a resource selection, such as network bandwidth, CPU usage or resource goodness, amongst others. A motivating scenario for an autonomic network-aware meta-scheduler architecture is depicted in Fig. 1 and includes the following entities [6]:

- *Users*, each with a number of jobs/tasks to run.
- *Computing resources*, which may include clusters running a Local Resource Management Systems (LRMS), such as PBS [14].
- *GNB* (Grid Network Broker), an autonomic network-aware meta-scheduler.
- *GIS* (Grid Information Service), such as [30], which keeps a list of available resources.
- *Resource monitor(s)*, such as Ganglia [31] or Iperf [32], which provide detailed information on the status of the resources.
- *BB* (Bandwidth Broker), such as [33], which is in charge of the administrative domain and has direct access to routers, mainly for configuration and topology discovering purposes.
- *Interconnection network*, such as a Local Area Network (LAN) or the Internet.

The interaction between components within the architecture is as follows:

1. Users ask the GNB for a resource to run their jobs. Users provide features of jobs (the "job template"), that includes the input/output files, the executable file and a deadline, amongst other parameters.