A Dynamic Co-allocation Service in Multicluster Systems

Jove M.P. Sinaga, Hashim H. Mohamed, and Dick H.J. Epema

Faculty of Electrical Engineering, Mathematics, and Computer Science
Delft University of Technology
P.O. Box 5031, 2600 GA Delft, The Netherlands
D.H.J.Epema@ewi.tudelft.nl
www.pds.ewi.tudelft.nl/~epema

Abstract. In multicluster systems, and more generally in grids, jobs may require *co-allocation*, i.e., the simultaneous allocation of resources such as processors in multiple clusters to improve their performance. In previous work, we have studied processor co-allocation through simulations. Here, we extend this work with the design and implementation of a dynamic processor co-allocation service in multicluster systems. While an implementation of basic co-allocation mechanisms has existed for some years in the form of the DUROC component of the Globus Toolkit, DUROC does not provide resource-brokering functionality or fault tolerance in the face of job submission or completion failures. Our design adds these two elements in the form of a software layer on top of DUROC. We have performed experiments that show that our co-allocation service works reliably.

1 Introduction

Computer systems consisting of multiple clusters, and more generally grids, offer the promise of transparent access to large collections of resources for very demanding applications. In fact, the needs of a single application may exceed the capacity available in any subsystem making up such a system, and so *co-allocation*, i.e., the simultaneous access to resources of possibly multiple types (processors, data, network bandwidth) in multiple locations, managed by different resource managers [1], may be required. Then, the jobs executing such applications consist of multiple components, with each component using resources in a different subsystem. When co-allocation is not used in multiclusters and grids, such systems only act as large load-balancing devices with higher-level schedulers trying to find good single locations for jobs to run. The real challenge of using such systems is in trying to achieve good mechanisms and policies for co-allocation.

Among the simplest types of applications that need co-allocation are parallel applications that require the simultaneous allocation of processors managed by different schedulers. The feasibility of running such applications in multicluster systems with their relatively slow wide-area connections has been demonstrated for instance in [2]. One of the main problems of processor co-allocation is to achieve the simultaneous availability of the processors managed by different local schedulers. The basic mechanisms for processor co-allocation have existed for a number of years in the form of

D. Feitelson, L. Rudolph, and U. Schwiegelshohn (Eds.): JSSPP 2004, LNCS 3277, pp. 194–209, 2005.

the Dynamically Updated Request Online Co-allocator (DUROC) [1] component of the Globus Toolkit [3]. However, even though DUROC serves its basic purpose of submitting multicomponent jobs, it cannot be regarded as a co-allocating scheduler for general use in multiclusters or in grids. First, it lacks resource allocation policies by requiring jobs to specify exactly the sites where their components should run. Secondly, DUROC does not provide good fault tolerance in that it may wait for enough available resources for an unspecified amount of time, and in that it requires the user's intervention when the submission or completion of a job fails.

In previous work we have studied processor co-allocation by means of simulations [2,4,5]. In this paper, we extend this work by the design and implementation of a prototype called the Dynamic Co-Allocation Service (DCS) on our wide-area Distributed ASCI[1] Supercomputer (DAS, see Section 4.1) that implements mechanisms and policies for processor co-allocation in multicluster systems. Our design is built on top of DUROC and consists of a Scheduler that implements policies such as FCFS and a form of backfilling, a Resource Monitor that reports on the available resources, a Resource Broker that maps jobs onto the most suitable clusters, and a Co-allocator that submits jobs to DUROC. In particular, our DCS solves the two problems with DUROC mentioned above. The results of the experiments with our prototype show that it works reliably.

2 The Problem of Processor Co-allocation

In this section we formulate the problem of co-allocation in multiclusters, we discuss the DUROC component of Globus, and we describe the structure of multicomponent jobs and the placement and scheduling policies used in our design.

2.1 Processor Co-allocation

In itself, processor co-allocation is a simple notion: Assign processors in different systems in a multicluster or a grid to single jobs simultaneously. The potential advantage to a job is that it may employ more processors than available in a single cluster and so may experience a shorter runtime, and the potential advantage to the system is that the system load may be increased. Of course, due to the relatively slow wide-area communications, not all applicatons will benefit from using processors in clusters connected by a wide-area network, but some definitely will [2].

An important issue in processor co-allocation is that the processors in the different clusters have to be available at the same time. What would be very helpful to guarantee the simultaneous allocation of processors is a reservation mechanism of the local resource managers. However, hardly any of the popular local resource managers such as PBS [6] supports such a mechanism. A processor co-allocation mechanism built on top of resource managers without reservation capabilities cannot do anything else than repeatedly try to assess whether sufficient numbers of processors are available, and then claim these.

[1] ASCI is the acronym of the Advanced School for Computing and Imaging in the Netherlands.