Managing SLAs of Heterogeneous Workloads using Dynamic Application Placement

David Carrera
Technical University of Catalonia (UPC)
Barcelona Supercomputing Center (BSC)
Barcelona, Spain
david.carrera@bsc.es

Malgorzata Steinder
IBM T.J. Watson Research Center
Hawthorne
NY 10532
steinder@us.ibm.com

Ian Whalley
IBM T.J. Watson Research Center
Hawthorne
NY 10532
inw@us.ibm.com

Jordi Torres
Technical University of Catalonia (UPC)
Barcelona Supercomputing Center (BSC)
Barcelona, Spain
jordi.torres@bsc.es

Eduard Ayguadé
Technical University of Catalonia (UPC)
Barcelona Supercomputing Center (BSC)
Barcelona, Spain
eduard.ayguade@bsc.es

ABSTRACT

In this paper we address the problem of managing heterogeneous workloads in a virtualized data center. We consider two different workloads: transactional applications and long-running jobs. We present a technique that permits collocation of these workload types on the same physical hardware. Our technique dynamically modifies workload placement by leveraging control mechanisms such as suspension and migration, and strives to optimally trade off resource allocation among these workloads in spite of their differing characteristics and performance objectives. Our approach builds upon our previous work on dynamically placing transactional workloads. This paper extends our framework with the capability to manage long-running workloads. We achieve this goal by using utility functions, which permit us to compare the performance of various workloads, and which are used to drive allocation decisions. We demonstrate that our technique maximizes heterogeneous workload performance while providing service differentiation based on high-level performance goals.

Categories and Subject Descriptors: H.4 [INFORMATION SYSTEMS APPLICATIONS]: General

General Terms: Algorithms, Management, Performance

1. INTRODUCTION

In this paper we present a technique that allows the management to high-level goals of collocated long-running and transactional workloads in virtualized environments. We use utility functions to model the satisfaction of both long-running jobs and transactional workloads for a particular resource allocation – the different types of workload have different characteristics, and different performance goals, and utility functions offer a mechanism to make their performance comparable. We run both workloads inside virtual machines, in order to properly manage their performance, and our management also exploits the clustering nature of transactional workloads. A preliminary working prototype of our proposal that made use of a commercial middleware to enforce its decisions was described in [5]. The performance management aspects pertaining to transactional workloads were introduced in [2]. An extended version of this paper can be found in [1].

The explicit management of heterogeneous workloads was previously studied in [6]. This was a static approach, and did not run workloads within virtual machines. The use of utility-driven strategies to manage workloads was first introduced in the scope of real-time work schedulers [3]. In our work we use monotonic and continuous utility functions to represent the satisfaction of both transactional and long-running workloads, but other approaches have been studied in the literature [4].

HPDC'08, June 23–27, 2008, Boston, Massachusetts, USA.
ACM 978-1-59593-997-5/08/06.

2. PERFORMANCE MODEL

The major challenge of heterogeneous workload management is to provide performance predictions with respect to job completion time on a control cycle which may be much lower than job execution time. Typically, such a prediction would require us to calculate an optimal schedule for the jobs. To trade off resources among transactional and long-running workloads we would have to evaluate a number of such schedules calculated over a number of possible divisions of resources among the two kinds of workloads. The number of combinations would be exponential in the number of nodes in the cluster.We avoid this complexity by proposing an approximate technique described in this section.

While the actual utility achieved by a job can only be calculated at completion time (as a function of actual completion time and the objective completion time), the algorithm needs a mechanism to predict (at each control cycle) the utility that each job in the system will achieve given a particular allocation. And this is still true even for jobs that are

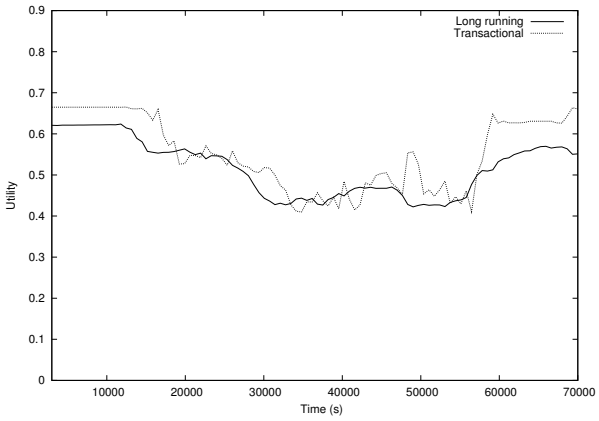

Figure 1: Actual utility for the transactional workload and average hypothetical utility for the long-running workload

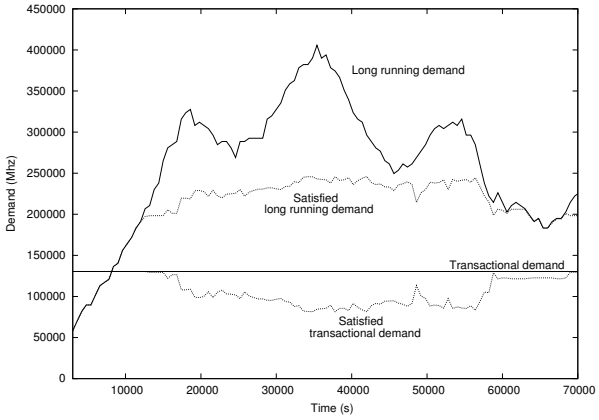

Figure 2: CPU power allocated to each workload and CPU demands to achieve maximum utility

not yet started, for which the expected completion time is still undefined. To help answer these questions we introduce the concept of *hypothetical utility*, for which we assume that all jobs can be placed simultaneously, and in which the available CPU power may be arbitrarily finely allocated among the jobs so that the expected utility is equalized amongst them. The algorithm operates by continously stealing resources to the more satisfied applications to later be given to the less satisfied applications.

3. EVALUATION

We consider a system of 25 nodes, each of which has four processors. To the system we submit 800 identical jobs. Each job's maximum speed permits it to use a single processor, and so four jobs could run at full speed on a single node. However, the memory characteristics of the system mean that only three jobs will fit on a node at once. Jobs are submitted to the system using an exponential inter-arrival time distribution with an average inter-arrival time of 260s. The system is configured to re-calculate application placement every 600 s.

The experiment starts with a system subject to a constant transactional workload used throughout, in addition to an insignificant number of long-running jobs already placed. In this state, the transactional application gets as much CPU power as it can consume, as there is little or no contention with long-running jobs. As more long-running jobs are submitted the hypothetical utility for those long-running jobs starts to decrease as the system becomes increasingly crowded. As soon as the hypothetical utility calculated for the long-running jobs becomes lower that the utility observed for the transactional workload, our algorithm starts to reduce the allocation for the transactional workload and give that CPU power instead to the long-running workload, until the utility each achieves is equalized. At the end of the experiment the job submission rate is slightly decreased, what results in more CPU power being returned to the transactional workload again. Figure 1 shows the utility for both of the workloads during the experiment. The utility for both workloads is continuously adjusted by dynamically allocating resources over time. Figure 2 shows the particular allocation at each moment of the experiment, as well as the CPU demand that would make each workload achieve its maximum utility. Notice how, as it was pursued, our technique makes an uneven distribution of resources in terms of CPU capacity, but it results in an even level of utility across the workloads.

Acknowledegments

This work is partially supported by the Ministry of Science and Technology of Spain and the European Union (FEDER funds) under contract TIN2007-60625 and by the BSC-IBM collaboration agreement SoW Adaptive Systems.

4. REFERENCES

[1] D. Carrera, M. Steinder, I. Whalley, J. Torres, and E. Ayguadé. Managing SLAs of heterogeneous workloads using dynamic application placement. Technical Report RC 24469, IBM Research, Jan. 2008.

[2] D. Carrera, M. Steinder, I. Whalley, J. Torres, and E. Ayguadé. Utility-based placement of dynamic web applications with fairness goals. In *11th IEEE/IFIP Network Operations and Management Symposium (NOMS 2008)*, Salvador Bahia, Brazil, 2008.

[3] E. D. Jensen, C. D. Locke, and H. Tokuda. A time-driven scheduling model for real-time operating systems. In *IEEE Real-Time Systems Symposium*, pages 112–122, 1985.

[4] C. B. Lee and A. E. Snavely. Precise and realistic utility functions for user-centric performance analysis of schedulers. In *HPDC '07: Proceedings of the 16th international symposium on High performance distributed computing*, pages 107–116, New York, NY, USA, 2007. ACM.

[5] M. Steinder, I. Whalley, D. Carrera, I. Gaweda, and D. Chess. Server virtualization in autonomic management of heterogeneous workloads. In *10th IEEE/IFIP Symposium on Integrated Management (IM 2007)*, Munich, Germany, 2007.

[6] Sun Microsystems. Behavior of mixed workloads consolidated using Solaris Resource Manager software. Technical report, May 2005.