QoS Support in the Context of Virtual Servers*

Juan A. Villar, Francisco J. Alfaro, José L. Sánchez
Dpto. de Sistemas Informáticos. Escuela Politécnica Superior
University of Castilla-La Mancha
02071 - Albacete, Spain
{juanan, falfaro, jsanchez}@info-ab.uclm.es

José Duato
Facultad de Informática
Univ. Politécnica Valencia
46022 - Valencia, Spain
jduato@disca.upv.es

1. Introduction

The last decade has witnessed a dramatic increment in the variety of computing devices as well as in the number of users of those devices. Moreover, connectivity has also dramatically improved. The traditional modem attached to the telephone line has been replaced by wired devices that offer much higher bandwidth, and wireless connections are becoming more available every day. The main reasons for the widespread use of computing devices are the availability of cheaper and more powerful devices and, even more important, the huge amount of information and services available through Internet.

The information and services provided through Internet rely on applications executed in many servers all around the world. Many of those servers were originally based on personal computers, but the huge increment in the number of users worldwide quickly led to a dramatic increment in the number of clients concurrently accessing to a given server. As a result, the computing power and I/O bandwidth offered by a single processor and a few disks were not enough to provide a reasonable response time. Clusters emerged as a cost-effective platform to run those Internet applications and provide service to hundreds or thousands of concurrent users [3, 5].

2. Internet Servers

Clusters are chosen in most cases today as a platform to provide network services because of their important advantages. The traditional way of sharing a cluster among several applications consists in using some tool or middleware to distribute the load among the processors in the cluster [2]. These tools do not provide support for Quality of Service (QoS) by themselves, and thus, they cannot guarantee server response time.

*This work was partly supported by the Spanish CICYT under Grant TIC2003-08154-C06 and by the Junta de Comunidades de Castilla-La Mancha under Grant PBC-05-005-1.

Processors are underutilized in most current Internet servers because servers are usually overdimensioned in such a way that they are able to guarantee a short response time under peak load. As a consequence, server cost is significantly higher than strictly necessary.

2.1. Virtual Servers

We define a Virtual Server (VS) as a set of resources (processors, memory, disks, interconnects) that are devoted to run a set of applications. Those resources are mapped to some physical devices or part thereof in such a way that several Virtual Servers can coexist in a physical cluster, sharing the devices it contains.

Virtual Servers allow applications to provide more details on their specific requirements. By using the concept of VS, the requirements of each application can be identified, thus assigning a correctly dimensioned set of resources to the VS running the applications. Moreover, applications may specify requirements that vary over time. For example, an Internet server will likely receive more requests during day hours. Thus, it can be reconfigured during the night to perform some maintenance functions, like backup. For that purpose, the system can be dynamically reconfigured by assigning a different set of resources to each VS according to its particular needs. Obviously, reconfigurations cannot be too frequent. Otherwise, the overhead introduced by the reconfigurations would consume a significant fraction of the system resources.

Moreover, allowing a detailed specification of application requirements, Virtual Servers provide an excellent framework for increasing security and fault tolerance, and also for providing QoS guarantees. Effectively, as each VS uses only a given set of resources, it does not require access to the rest of resources. Thus, it is possible to implement some hardware and/or software mechanisms to prevent unauthorized accesses to resources not belonging to a given server. This is not a trivial task since many resources

may be shared by different Virtual Servers in order to use the system more efficiently.

All of these problems (i.e., security, fault tolerance, and QoS) are considered to be extremely important, given the role of servers in today's scenario. However, in order to limit the scope of this project, we need to focus on only one of them. We have selected the problem of providing QoS guarantees because it is the one most easily perceived by the end user, and also because of its dramatic impact on system cost.

Virtual Services [6] provide dynamic per-service resource partitioning and management in a manner completely transparent to applications. Virtual Services are a kernel-based work classification mechanism. This approach uses tags operating system entities like processes, sockets, etc., with Virtual Services information. The operating system tracks the members of a service without requiring continuous application or administrator intervention. Unfortunately, its implementation does not cover servers based on cluster systems.

3. Virtual Servers in Linux

Our research hypothesis is that it is possible to use servers in a much more cost-effective way, while still guaranteeing response time and throughput under peak load, for those applications requiring a certain amount of resources and/or bounded response time when processing client requests. In particular, we believe this is possible in the context of several Virtual Servers sharing a single physical server or a cluster. We base this hypothesis on the following observations:

- Applications running on different Virtual Servers usually have resource requirements that vary over time.
- Not all the applications running on a server at a given instant in time require bounded response time.
- Not all the applications that require bounded response time need to provide the same QoS to their clients.

Specifically, we plan to develop a processor scheduler together with a suitable admission control algorithm that will guarantee that all admitted applications (or Virtual Servers) will be classified into several categories, depending on their QoS requirements. We plan to implement support for the following categories:

- Dedicated Resources Time Sensitive. Applications that require a guaranteed set of (virtual) resources and guaranteed bounded response time (e.g., a video-on-demand server).
- Dedicated Resources. Applications that require a guaranteed set of resources but do not require bounded response time (e.g., a web server).
- Best Effort. Applications that do not require neither a guaranteed set of resources nor bounded response time, but require a significant amount of computing power, and thus, they require all the spare computing power available at a certain instant in time (e.g., any numerical computation).
- Challenged. Applications that will only be executed when overall load goes below a certain threshold or will be periodically scheduled when load is low (e.g., backup).

Our approach will need the intervention of the system administrator who must give an explicit definition of every VS in the system with its variation over time and member applications.

4. Work in progress

We are implementing the proposed processor scheduling strategy on a Linux kernel as well as the proposed admission control strategy. Moreover, we plan to evaluate the behaviour of both single-processor servers and cluster-based servers. Our studies also include processors with and without HyperThreading and MultiCore support [4, 1] because they are the natural environment of Internet servers.

References

[1] AMD Inc. `http://www.amd.com`.
[2] Condor Project. `http://www.cs.wisc.edu/condor`.
[3] Google Inc. `http://www.google.com`.
[4] Intel Co. `http://www.intel.com`.
[5] Source Forge. `http://www.sourceforge.net`.
[6] J. Reumann, A. Mehra, K. G. Shin, and D. Kandlur. "Virtual Services: A New Abstraction for Server Consolidation". San Diego, CA, U.S.A., June 2000. in Proc. of USENIX 2000 Annual Technical Conference.