Application Performance in the Cloud, and its Relationship to QoS

Fall 2010

First, Last
Someone@my.csun.edu
Student ID: xxxxxxxxx
September 22, 2010

Committee Chair: Dr. Shan Barkataki

Approved by: _____________________________________

Committee Chair
Objective

The objective of this project is to analyze the performance of virtual machine instances running on a single piece of hardware, and to establish measures for describing the effects of resource consumption on the service performance as experienced by the end user. Resource consumption will be broken up into several different categories. Stress conditions will be simulated on the CPU, disk, and network.

Introduction:

Cloud computing is the virtualization and on demand presentation of hardware and software resources to users outside of the cloud. Cloud computing simplifies the user's experience by providing certain services within the scope of the cloud, abstracting the implementation details. The term "cloud" as it relates to computer science is a very broad one. Cloud providers offer a large variety of services from virtualized machine instances within the cloud, such as Amazon EC2, to online office productivity and infrastructure application software, such as that provided by Google Docs and Zoho, with many others in between.

A driving force in the shift to cloud computing, data centers are increasingly moving towards virtualizing their environments, using software such as VMWare ESX, Xen, OpenNebula, Eucalyptus, among others, to reduce both hardware and maintenance costs, while increasing agility[1]. In doing this, the provider must be able to guarantee the end user a certain level of performance, in order to make this service commercially viable. In the case of virtual machine instances running on a single piece of hardware, the need arises to be able to track and monitor hardware performance of the physical server in order to help guarantee a certain quality of service. My project will focus on the effects of multiple virtual machine instances running on a single hardware server, how they affect each other, saturate the hardware they run on, and how this effects quality of service as perceived by the end user.

Technical Approach:

My goal with this project will be to test the performance of multiple virtual machines running within a single physical server/workstation, and analyze how consumption of certain resources, such as CPU, memory, and I/O (disk and network), affect the Quality of Service as perceived by the end user.

In order to accomplish this goal, I will setup multiple virtual machine instances on a single hardware server or workstation using the open source virtualization software, Xen. I will then establish a set of criteria which I will evaluate the system against. The criteria will involve stressing a particular resource of each virtual instance in unison, and then checking to see how this impacts the physical server’s resources, and QoS from the end user’s side. Performance will be measured using both tools local to the physical server, as well as tools used across the network from the client machine.

Xen was originally developed at Cambridge University, and is now licensed under the
GNU General Public License, and developed by the open source community. I’ve chosen to use Xen as opposed to some of the alternatives, such as VMWare, OpenNebula, Eucalyptus, and OpenQRM, mainly because I am familiar with it, it is freely available, and it is closely integrated with the Redhat Enterprise Linux operating system, for which it is the official virtualization option.

Xen works by running a “hypervisor” layer which runs between the guest operating system(s), and the server hardware. The hypervisor is responsible for apportioning system resources to all Guest Operating systems running on the host. Operating systems assume they control the underlying hardware, so a side effect of this setup is that the Guest OS sits on a virtualization layer rather than the hardware layer, and cannot execute privileged instructions on the CPU without help from the hypervisor. Certain kernel level instructions must run in privileged CPU mode, and cannot be virtualized. (Note: user level instructions can execute without hypervisor assistance in user mode)

There are two primary solutions to this problem; full virtualization and paravirtualization, of which Xen provides support for both. Full virtualization requires newer CPU extensions. For Intel, this would be the Intel Virtualization Technology (VT) extensions, and for AMD, it would be the AMD Virtualization (v) extensions. In full virtualization, the hypervisor runs on the underlying hardware. Each Guest OS has a corresponding Virtual Machine Monitor within the hypervisor. The Virtual Machine Monitor operates in privileged CPU mode, executing certain instructions which require translation, on behalf of the Guest OS. As a result, the Guest OS is unaware that it does not have control of the server hardware. The virtualization layer is transparent to the OS. Since no OS modification is required, any operating system can be supported (so long as it is compatible with the hardware) be it Redhat Linux 5 or Windows 2003.

In contrast, paravirtualization requires kernel modification. In paravirtualization, the Xen hypervisor also runs between the operating systems and the hardware. However, in this case the paravirtualized Guest OS is aware of the hypervisor, and makes special calls to the virtualization layer to have these OS instructions translated. Since significant changes to the OS kernel are required, only a limited number of operating systems, specifically Redhat 4.5 and later, are supported in this configuration. No versions of Windows can be run as paravirtualized hosts on Xen. Due to hardware constraints, this will likely be my choice for this project.

Once the hypervisor is installed, Xen boots to a privileged (not in the CPU sense of the word) Redhat domain called “Dom0”. This is the first virtualized guest domain, and it has special rights to administer and manage all the other VM domains, “Dom U”, on the server. It should be mentioned that security of Dom0 is crucial as its compromise results in the compromise of all VM's on the server. Finally, a xend daemon also runs on Dom0 to provide virtualization services.

I will be creating multiple DomU's on top of the Xen hypervisor using Redhat 5. The total number of these guests is yet to be determined, as it will require some hands on work to figure out a number of virtual machines the host can reasonably handle. This will be dependent on the hardware that is selected for the project.
Once the number of Guest OS’s is determined, I will conduct a set of tests against each virtual machine to consume various resources, and monitor the combined effect this has on the underlying hardware, and the response as perceived by the end user. Potential stress tests that will be run on the Guest OS’s include CPU consumption, memory consumption, and I/O activity on the local disk, as well as over the network.

Several tools I will use to conduct stress tests include Iperf and Izone. Iperf is a network intensive program that can be used to test over the wire performance. Iozone is a disk intensive program used to measure performance. Along with these tools, I will locate, or write, additional tools to consume both CPU and memory. I will then break up the tests in a way similar to the following:

Assuming a 3 Guest Configuration:
Guest 1 - Run network intensive test
Guest 2 - Run disk intensive test
Guest 3 - Run CPU intensive test

Each guest OS will be running a web site on Apache, as well as a backend database. I will measure user impact by observing the effect on both ping time, as well as the time to download a webpage from each guest while the stress tests run. By setting up a large set of tests, I hope to establish an idea of what activities have the greatest impact on the user’s experience. I will setup as many of these tests as possible.

Logistics:

Ideally, I'd like to have some type of modern, multi-core server with a CPU that supports VT functionality. Practically speaking, I may end up using my desktop, or attempting to borrow an older server, possibly from work. I would like to use a dual processor DL380 G3/G4 server class machine with at least 4 GB of RAM, so I can use more virtual instances. In the case that I am able to come up with a machine that supports VT functionality, it would allow me to make use of Xen’s full virtualization support. This means I could run Guest OS instances of both Linux and Windows operating systems, setting up tests with all Windows guests running, all Linux, and a combination of the two. However, due to financial constraints, I will probably need to use an older machine without VT technology, requiring me to use paravirtualization. This will limit my OS options on Xen to RHEL 4.5 and above. I will also need a second less powerful machine, to act as the client. This computer could be anything from a laptop to a desktop machine.

Additionally, I will need to finalize my toolset. For the stress test tools, I have the necessary software to do disk and network tests, but I need to obtain software to stress the CPU and memory. I will probably write a simple program to handle the latter two.

To measure response on the end user’s side, I will setup Apache web server as well as MySQL, and run some type of shopping cart or website on the Guest OS’s. The end user will then attempt to retrieve web pages, and ping the guest servers. Response times will be gathered from the ping and wget utilities. Performance over a period of time may be monitored with scripts, or more complex tools such as Nagios or RRDTool.
I plan to take two semesters to carry out this project. A tentative schedule is included below.

<table>
<thead>
<tr>
<th>Tasks</th>
<th>Sep</th>
<th>Oct</th>
<th>Nov</th>
<th>Dec</th>
<th>Jan</th>
<th>Feb</th>
<th>Mar</th>
<th>Apr</th>
<th>May</th>
</tr>
</thead>
<tbody>
<tr>
<td>Formulate Proposal</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Research Test Cases</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Research S/w options to perform test cases</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Obtain/Write Necessary Software for toolset</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Finalize toolset</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Determine what services VM's will provide.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Finalize test cases</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Assemble hardware and testbed.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Install software, configure physical server, and VM's.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Run tests and analyze data</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Analyze results, draw conclusions on performance</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Thesis</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Prepare defense</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Criteria for Success:

- Completed literature survey on virtualization software and provided a critical analysis of the current literature (in thesis).
- Established and defined measures for expressing server performance as a function of resource consumption.
- Established and described a plan for performing the set of experiments for evaluating the performance of servers in a virtual environment.
- Obtained or developed necessary software to do stress testing and perform monitoring of server performance.
- Developed or obtained a testbed for conducting the performance tests.
- Created and validated test cases.
- Executed all planned tests and analyzed test results to draw conclusions.
- Completed and defended thesis.

References:

1 - VMWare White Paper. “Understanding Full Virtualization, Paravirtualization, and Hardware Assist”. VMWare.


8 – The Xen Hypervisor, Open Source Virtualization - http://www.xen.org

9 – Linux Vserver Virtualization for GNU/Linux Systems - http://linux-vserver.org

10 – VMWare Virtualization Software for CC Solutions - http://www.vmware.com

11 - Amazon Elastic Compute Cloud (EC2) - http://aws.amazon.com/ec2/