Bandwidth Limits for Network Usage by Virtual Machines

George Pandzik, Navid Pustchi, Morteza Shahriari Nia
University of Texas at San Antonio
gpandzik@cs.utsa.edu, npustchi@cs.utsa.edu, msnia@cs.utsa.edu

Abstract

Performance optimization and administration has always been a necessity in virtualized environments. In this paper, we present how to change a DomU's bandwidth limits under Xen, which is an open source virtualization Hypervisor. The approach we used is a static way to change the bandwidths limits, though we investigated the possibility of doing the same procedure in a dynamic manner. Our experiments shows that bandwidth utilization is limited by the specified rates.

1. Introduction

The basic motivation for virtualization is based on the same concept as multitasking in operating systems and the fact that computers have more processing power than one task needs. Virtualization helps with a variety features as cloning, migration and power usage. For example it allows a number of virtual servers to be consolidated to a single physical machine without losing the security gained by having completely isolated environments.

There are approaches like binary rewriting that has been used by VMWare. This approach imposes performance penalties, particularly when you have anything I/O intensive. Performance is about 80-97% of the host machine since you have the overhead of the underlying operating system.

The other approach is paravirtualization (PV) which is a technique that allows the operating system to be aware that it is running on a hypervisor instead of base hardware.

Xen is an open source hypervisor that is paravirtualized and make it possible to run multiple operating systems on hardware without the performance downgrades of the previous approaches on CPU, I/O and Memory as mentioned.

It has tiny hypervisor model, 2MB executable, no device drivers and keeps domain /guests isolated. Xen separates the hypervisor execution from management OS, management stack, device drivers, and guests. Guests are interchangeable, which means you can choose the best OS to support your needs. It has strong isolation between all guests assisted with modern hardware and domains can restart without taking out full system.

A critical benefit of the Xen Hypervisor is its neutrality to the various operating systems. Due to its independence, Xen is capable of allowing any operating system (Linux, Solaris, BSD, etc) to be the Domain0 thereby ensuring the widest possible use case for customers. This separation of hypervisor from the Domain0 operating system also ensures that Xen is not burdened with any operating system overhead that is unrelated to processing a series of guests on a given machine. In fact, this will lead into a series of guests each with a specific purpose and responsibility which drives better performance and security in a virtualization environment.

As for DomainU, also called Unprivileged Domains or guests, Xen has supported most operating systems, such as many Linux flavors that come with their own PV drivers as part of the kernel, and Windows, which requires additional PV drivers to ensure that the broader enterprise computing industry is able to deploy their OS of choice as a guest on Xen.

Xen has ensured security as hypervisor via a variety of methods and features:
• Guest Isolation – every DomainU guest is isolated from other DomainU guests with no way to access each other's memory or networking connections.
• Privileged Access – only the Domain0 or single purpose control guests are given the ability to communicate with the hardware via the hypervisor.
• Small Code Base – the Xen hypervisor contains a “tiny” code footprint which limits the areas for attack.
• Operating System Separation – by separating the hypervisor from an operating system, the Xen hypervisor cannot be used to attack an operating system; e.g. Xen cannot attack the host operating system as there is no host operating system to attack.

Xen also has been an appealing option for cloud computing and it the most used virtualization platform in the cloud computing space. Xen Cloud Platform (XCP) offers ISVs and service providers a complete cloud infrastructure platform with a powerful management stack based on open, standards-based APIs, support for multi-tenancy, SLA guarantees and detailed metrics for consumption based charging. But XCP is beyond the scope of this research effort.

2. Problem statement and proposed solution

Working in a shared environment, it is often necessary to cap or restrict the ability of any one guest to dominate its peers. Unfortunately, Xen does not by default limit the network bandwidth utilization of the DomUs on the system.

Xen does, however, provide a tool which can be used to define a limit on a DomU's network bandwidth, which we propose to use in this project. This limit is called the 'rate=' parameter of the vif= settings in a DomU's configuration files (by default, /etc/xen/<DomU name>). The rate parameter can be set in the configuration file and the DomU restarted to read the new value. The rate parameter on the vif statement is translated into a bits-per-period limit and length-of-period key-pair. This provides a readily-usable way to limit out-going bandwidth usage on a per-DomU basis.

3. Procedures

The way to set or change a bandwidth limit is as follows (note, we will make some assumptions; namely, the paths are Xen-on-CentOS-v5.5-default and our DomU names are 'fedora01' and 'fedora02'):

1. Acquire root-level privileges (using the `su` or `sudo` commands)
2. Edit /etc/xen/<DomU name>, such as `vim /etc/xen/fedora01`
3. Add a 'rate=' parameter to the end of the vif= line (inside the quotes and parentheses, if appropriate), such as `,rate=20Mb/s`
   a. Note: The rate number must be an integer
   b. The units can be in [K|M|...][B|b]/s, where Kb/s implies Kilobits per second, MB/s implies Megabytes per second, and so forth
4. If the DomU is currently running, get its domain ID (DomID) using the `xm list` command
5. Stop the DomU using either the `xm shutdown <DomID>` command or through the DomU's operating system itself. Note that a reboot is not sufficient. E.g., if fedora01's DomID is 3, then issue `xm shutdown 3`.
6. Once the DomU is down, or if it was not running at the time of the change, issue the `xm create <DomU name>` command, such as `xm create fedora01` to start the DomU
7. Verify the limit is in effect using various standard tools, such as a bandwidth-testing website or `iperf`

4. Results

To demonstrate the effectiveness of the adopted approach on limiting the network bandwidth of DomUs, we setup a test bed consisting of a CentOS-5.5 system along with Xen as Dom0 and two Fedora 12 systems which were installed as DomUs on top of CentOS. To measure the bandwidth, we used two network speed testing websites: speedtest.net and speakeasy.net. We also tested our experiments via *iperf* network testing tool (more specifically, its GUI interface *jperf*.) with 10 seconds as the duration of each trial. We observed that the workings of neither DomU affected the bandwidth limits of the other DomUs in the system -- all limits were effective at all times.

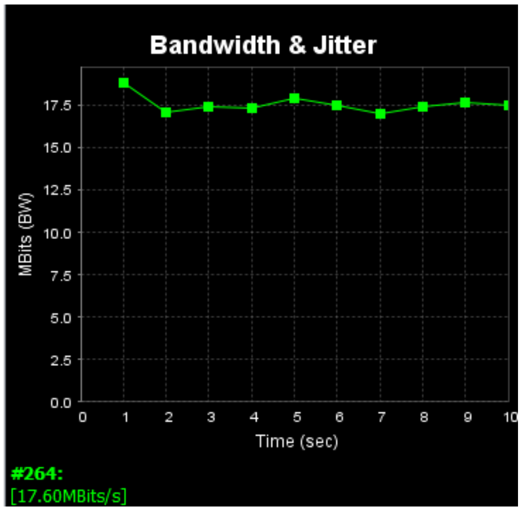

Figure 1- Practical bandwidth of a DomU with a set 20Mbps of bandwidth limit

In an unlimited bandwidth setup, both of the DomUs struggled for more bandwidth and the bandwidth was distributed in almost a fair and equivalent manner. However, in the case of a 20Mbps bandwidth limit on one of the DomUs the actual limit seemed to be around 17Mbps-18Mbps. See Figure 1.

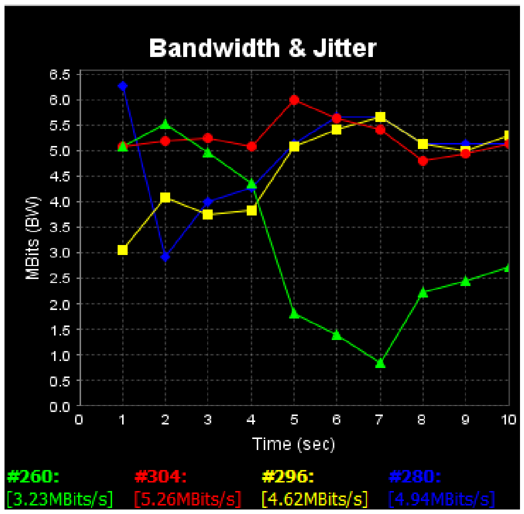

Figure 2 - Demonstrating the individual bandwidth of parallel streams outgoing from a DomU with a set limit of 20Mbps which accumulates to 17.9

Determining the impact of multiple parallel streams from one DomU was also important, the compatibility of which with the bandwidth limit is demonstrated below. With a bandwidth limit of 20Mbps the accumulative limit of all the streams summed up to around 18.4 Mbps which seems quite reasonable. See Figure 2.

We conjecture that the reason the limits were never fully attained could be due to a number of reasons, such as competition for the single physical network device among the DomUs during the tests or the limits of the TCP/IP framework and its check-and-resend properties.

5. Limitations and future research opportunities

We have found that the default behavior of Xen is to require the shutting down and restarting of a DomU to process the new network bandwidth limits. This presents a significant impediment to dynamic limiting, as it requires the complete shutdown of a guest, which can take minutes.

We have found how to change the rate variable on-the-fly, but the new rate doesn't go into effect until a "watch" function is triggered. We found many resources for how to trigger watches, but all appear to require kernel modifications, which are beyond the current scope of our project due to time constraints. This does, however, provide a good source of material for future research opportunities.

Another limitation to the current rate limits is the fact that only the out-bound transmissions are limited. This is fine for a file server, but does nothing for a DomU whose user is streaming video files, for instance. A future research opportunity would be to modify the Xen Dom0 code itself to add the ability to queue or discard incoming traffic as a brake on DomU downloading.

We also observed that multiple threads within a DomU can see vastly varying upload speeds, which can lead to some threads using almost the full allotment of upload "space," while other threads starve. Some method of sharing the upload resources more equitably would also be of interest, probably through modifications to the network front-end drivers within a DomU.

6. Conclusion

In this paper, we describe the procedure used to set static network bandwidth limits on Xen DomU guests. We verified the procedure works through various tests using iperf and other methods. As part of future work, we intend to investigate kernel-level modifications to implement dynamic rate changes using xenstore watches and modifications to the Xen source code to limit incoming transmissions, in addition to the current outgoing limits on DomUs.

7. References

[1] C. Takemura, L.S. Crawford, *The book of XEN a Practical Guide For the System Administrator*, No Starch Press, San Francisco, 2009

[2] D. Chisnall, *The Definitive Guide to the Xen Hypervisor*, Prentice Hall, Upper Saddle River, NJ, 2008

[3] XenStore Reference, http://wiki.xensource.com/xenwiki/XenStoreReference

[4] XenBus, http://wiki.xensource.com/xenwiki/XenBus

[5] *What is Xen*, http://www.xen.org/files/Marketing/WhatisXen.pdf

[6] *Why Xen*, http://www.xen.org/files/Marketing/WhyXen.pdf

[7] *How Does Xen Work*, http://www.xen.org/files/Marketing/HowDoesXenWork.pdf

[8] *Speedtest*, http://www.speedtest.net/

[9] *Speakeasy*, http://www.speakeasy.net/speedtest/

[10] *Iperf*, http://iperf.sourceforge.net/

[11] *Jperf*, http://code.google.com/p/xjperf/