An Empirical Evaluation of Methods for Improving Efficiency in Xen Virtual Machine CPU Scheduling

James Devine

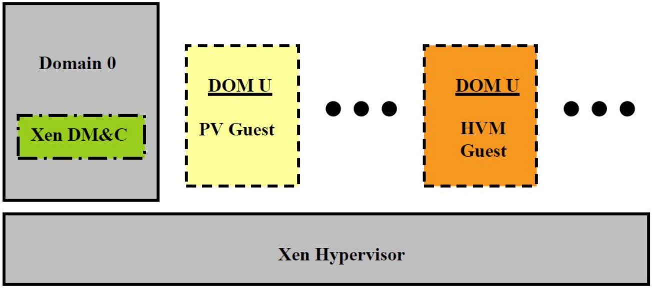
April 23, 2010

Thesis

An efficient virtual machine scheduling algorithm is important for increased throughput and decreased response time. Given varying workloads, there will be a particular scheduling algorithm that is more efficient at scheduling virtual machines for particular types of workloads. Thus, it is possible to fine tune the Xen hypervisor to maximize throughput and minimize response time with specific types of workloads.

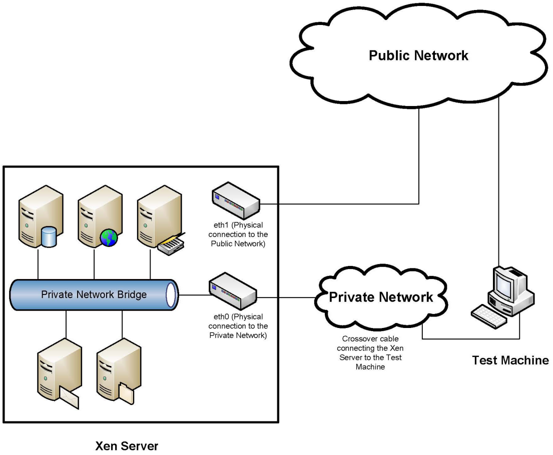
Xen Components

- Hypervisor
- Domain 0
- Domain Management and Control
- Domain U PV Guest
- Domain U HVM Guest



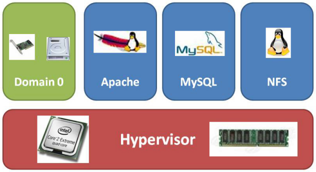
- Default Xen scheduler
- Proportional fair share CPU scheduler
- Each domain (including Domain0) is assigned a weight and a cap
- Weight
  - Domain with a weight of 512 will get twice as much CPU as a domain with a weight of 256
  - Default weight is 256
  - Weights range from 1 to 65535
- Cap
  - Optionally parameter - fixes the maximum amount of CPU a domain can use
  - Default weight cap is 0 (ie no cap)
  - Percentage of a whole CPU (ie 100 is 1 CPU, 200 is 2 CPUs)

- Dynamic priority real-time scheduling policy
- Processes placed in a priority queue
- Processes closest to their deadline execute first
- Per domain parameters
  - **period** - time interval during which a domain is guaranteed to receive its allocation of CPU time
  - **slice** - time per period that a domain is guaranteed to run for
  - **latency** - (unused by the currently compiled version)
  - **extra** - flag (0/1), which controls whether the domain can run in extra-time
  - **weight** - mutually exclusive with period/slice and specifies another way of setting a domains cpu slice


Public Network
eth1 (Physical connection to the Public Network)
Private Network Bridge
eth0 (Physical connection to the Private Network)
Private Network
Crossover cable connecting the Xen Server to the Test Machine
Test Machine
Xen Server

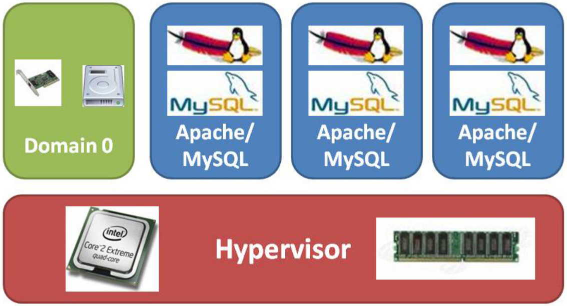
- Xen server and test machine were separate machines
- Private network was used for testing
- Public network was used for management

- A simple web application using Apache, SQL, and NFS Servers was created
- JMeter was used to simulate a user load of 25, 50, 100, 500, and 800 users
- For each user load 5 sEDF and 5 credit configurations were used and run 5 times each
  - Credit weights of 256, 512, 1025, 2048, and 4096 were used

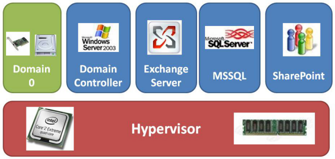


Configuration (name)	Domain0 Slice (ms)	MySQL, Apache Slice (ms)	NFS, VM	Period (ms)
Default (sedf-20)	15 (20ms period)	20		100
Equal Slices (sedf-2.5)	2.5	2.5		10
More Weight to Main VMs (sedf-2.8)	1.6	2.8		10
Equal Slices w/ Large Period (sedf-250)	250	250		1000
Greater Main VMs Slices w/ Large Period (sedf-280)	160	280		1000

- Uses same web application but running on multiple simple LAMP instances
- Due to time constraints this test could not be completed

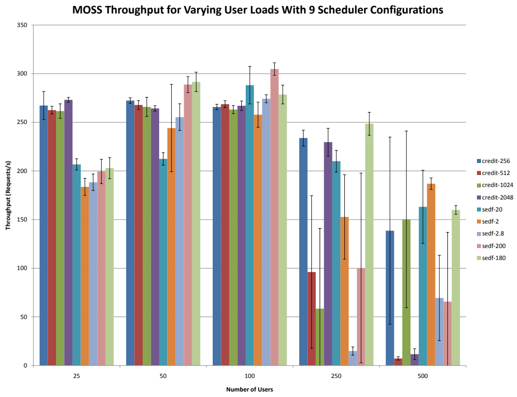


- Windows Domain Workload was created consiting of Active Directory, Exchange, MSSQL, and Sharepoint
- JMeter used to generate MOSS load and LoadGen used to generate Exchange load
- User loads of 25, 50, 100, 250, and 500 simulated
- For each user load 5 sEDF and 5 credit configurations were used and run 5 times each
  - Credit weights of 256, 512, 1025, and 2048 were used

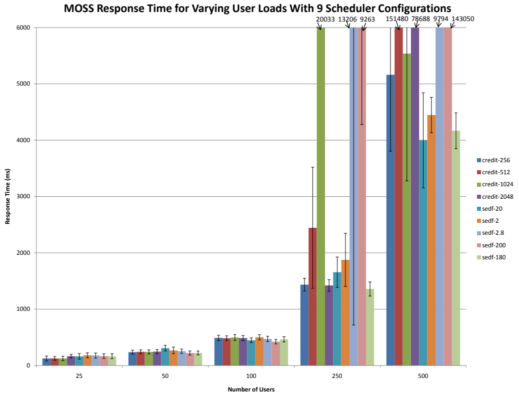


Configuration (name)	Domain0 (ms)	Slice	Main VM Slice (ms)	Other VM Slice (ms)	Period (ms)
Default (sedf-20)	15 (with period)	20ms	20	20	100
Equal Slices (sedf-2)	2		2	2	10
More Weight to Main VMs (sedf-2.8)	2		2.8	1.8	10
Equal Slices w/ Large Period (sedf-200)	200		200	200	1000
Equal Slices w/ Large Period (sedf-175)	175		175	175	1000

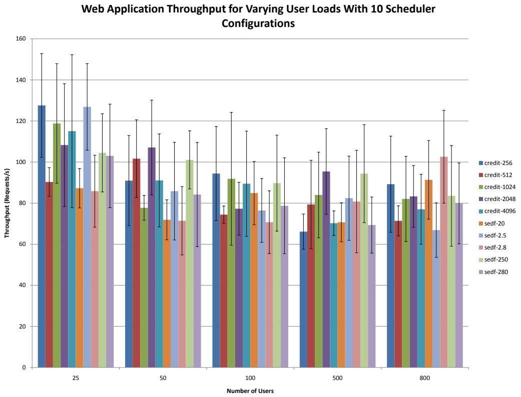
- JMeter log files were collected and placed into a single folder using powershell scripts
  - Log files were process with a python script to obtain response time for each request
  - Processed output was imported into Excel to perform statistical analysis
- Xenmon was run in the background during each test
  - R function created to take a given xenmon log file and output the wait, execution, and block time
  - Processed output was put into a table



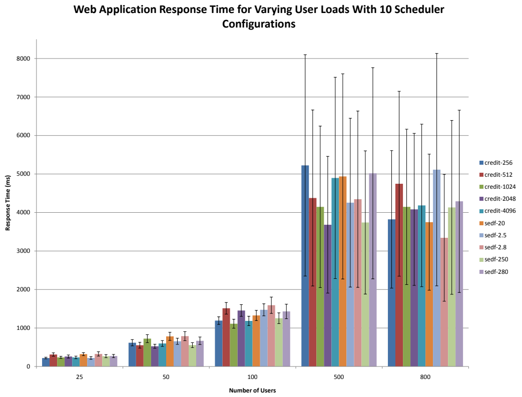
- Scheduler performance was largely determined by amount of users
- Default configuration of sEDF and credit schedulers did not perform best
- sEDF overall lead to higher levels of throughput



- Consistent response time for 25, 50, and 100 users
- Some configurations that had highest throughput did not have highest response time
- xenmon results indicate MOSS VM was CPU bound and MSSQL VM was CPU and I/O bound



- Similar variability in best scheduler configuration as seen with MOSS
- Default configuration of sEDF and credit schedulers did not perform best
- Higher scheduling preference to Domain0 (sedf-20 and sedf-280) did not optimize throughput or response time



- Scheduler configurations that had high throughput had low response times
- In general a doubling of response time up to 100 users
- Large jump in response time form 100 to 500 users

- CPU scheduler performance is both application and user load specific
- The default credit and sEDF scheduler configurations frequently lead to suboptimal levels of throughput
- In multiple cases the sEDF scheduler outperformed the credit scheduler
    - Interesting given the fact that the option to use sEDF will eventually be removed

- Three case study applications were created
  - Open source case study applications can be released for future research
  - The methodology for the Domain case study can be released so that it can be repeated
- The performance of two scheduling algorithms in various configurations was measured
- The "best" CPU scheduler configuration for each synthetic workload with a given target amount of users was determined

- Investigate more scheduler configurations
- Use multiple clients to generate user loads
- Determine why the “best” scheduler configuration varied with user load
- Determine if a higher throughput storage system shows similar trends to those found in this study
- Construct a system to automatically tune the Xen CPU scheduler based on the workload