

PERFORMANCE COMPARISON - XEN VS KVM

Since virtualization allows us to effectively utilize the system resources available, more and more people are trying to put their systems under virtualized environment. Selecting the right virtual system that suites your current system environment will be a difficult process as there are number of options available like xen, kvm, vmware etc. Here we are trying to get close with two prominent virtual system softwares in the market; Citrix's xen and Redhat's kvm. Before going to the benchmark tests that compare the performances of these two systems, it will be better to understand the basic idea of how they are providing these virtual environments. KVM is a hypervisor that is in the mainline Linux kernel. It is loaded to the Linux kernel as a module. It runs only on systems with hardware supporting virtualization extensions. On the other hand Xen is an external hypervisor; it assumes control of the machine and divides resources among guests and paravirtualized machines can be created without the support of hardware virtualization extensions.

Test Objectives

We are trying to closely watch the performances of file system and cpu operations of virtual machines that have been setup in xen and kvm virtual environment.

Test Environment

The base system that is used to setup virtual machines is having the following hardware specification.

Main Board : ASUS M2N68-AM Plus
CPU : AMD Athlon(tm) II X2 245 Processor
#cores : 2
cpu MHz : 2913.342
cache size : 1024 KB
Ram : 4GB DDR

Each virtual machines created in different virtual environments are using centos 64 bit operating system. Each virtual machine created is having the same amount of system resources like memory, storage etc, with the only difference that one is in a different virtual environment compared to the other. The test is done through benchmarking of file system and cpu operations on the two different virtual environment.

Tools used for benchmarking

We are using two tools to do the performance evaluations of various hypervisors. They are Bonnie for file system benchmarking and Nbench for cpu benchmarking. Various benchmarking tests will be performed and the comparisons will be shown with the help of plotted graphs. I will be explaining each attribute plotted in the graph, what exactly they mean. First we can go for file system benchmarking using Bonnie file system benchmarking application that will measure the performance of various file system operators.

File system benchmarking

Bonnie performs a series of tests on a file of known size. If the size is not specified, Bonnie uses 100 Mb; but that probably isn't enough for a big modern server - you want your file to be a lot bigger than the available RAM. Bonnie works with 64-bit pointers if you have them. For each test, Bonnie reports the bytes processed per elapsed second, per CPU second, and the % CPU usage (user and system). In each case, an attempt is made to keep optimizers from noticing it's all bogus. The idea is to make sure that these are real transfers between user space and the physical disk. The tests are:

Sequential Output

Per-Character : The file is written using the `putc()` stdio macro. The loop that does the writing should be small enough to fit into any reasonable I-cache. The CPU overhead here is that required to do the stdio code plus the OS file space allocation.

Block : The file is created using `write(2)`. The CPU overhead should be just the OS file space allocation.

Rewrite: Each Chunk (currently, the size is 16384) of the file is read with `read(2)`, dirtied, and rewritten with `write(2)`, requiring an `lseek(2)`. Since no space allocation is done, and the I/O is well-localized, this should test the effectiveness of the filesystem cache and the speed of data transfer.

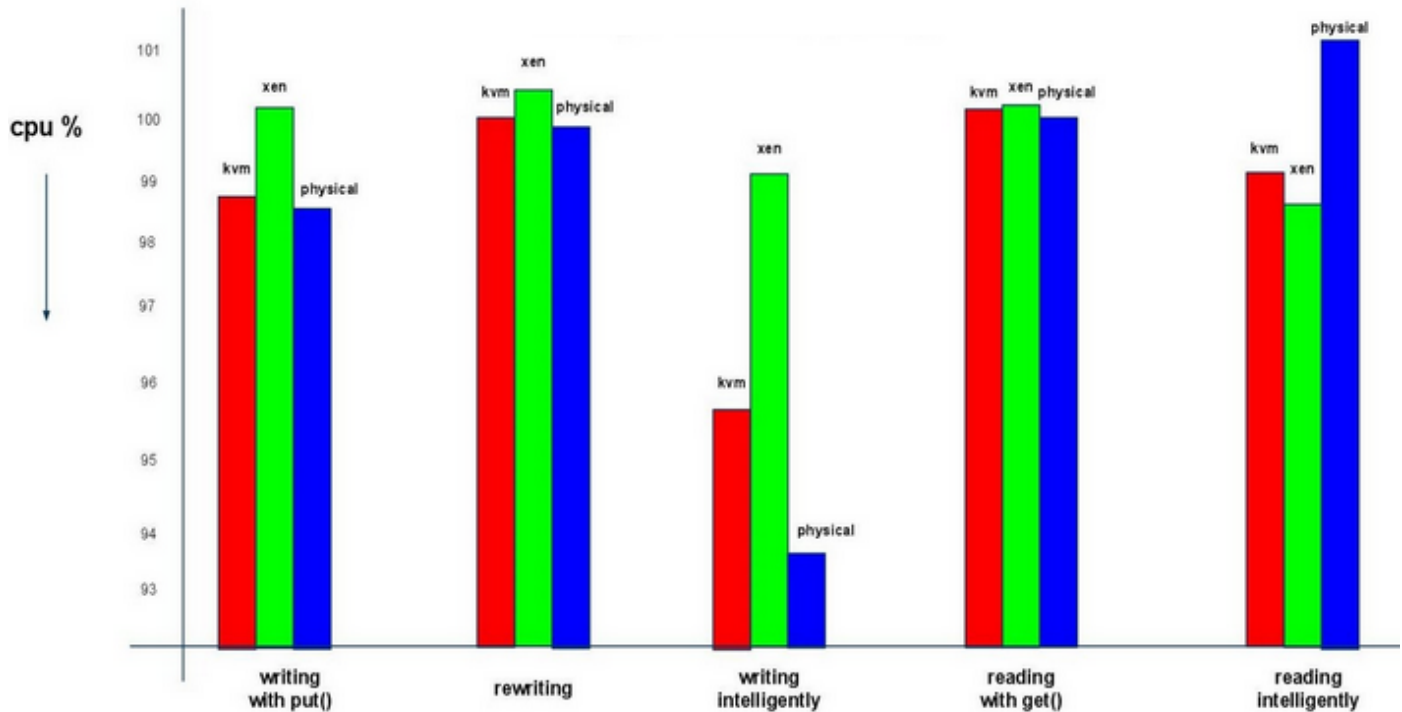
Sequential Input

Per-Character: The file is read using the `getc()` stdio macro. Once again, the inner loop is small. This should exercise only stdio and sequential input.

Block : The file is read using `read(2)`. This should be a very pure test of sequential input performance.

Results of tests done on different virtual systems and a non virtual system is shown below.

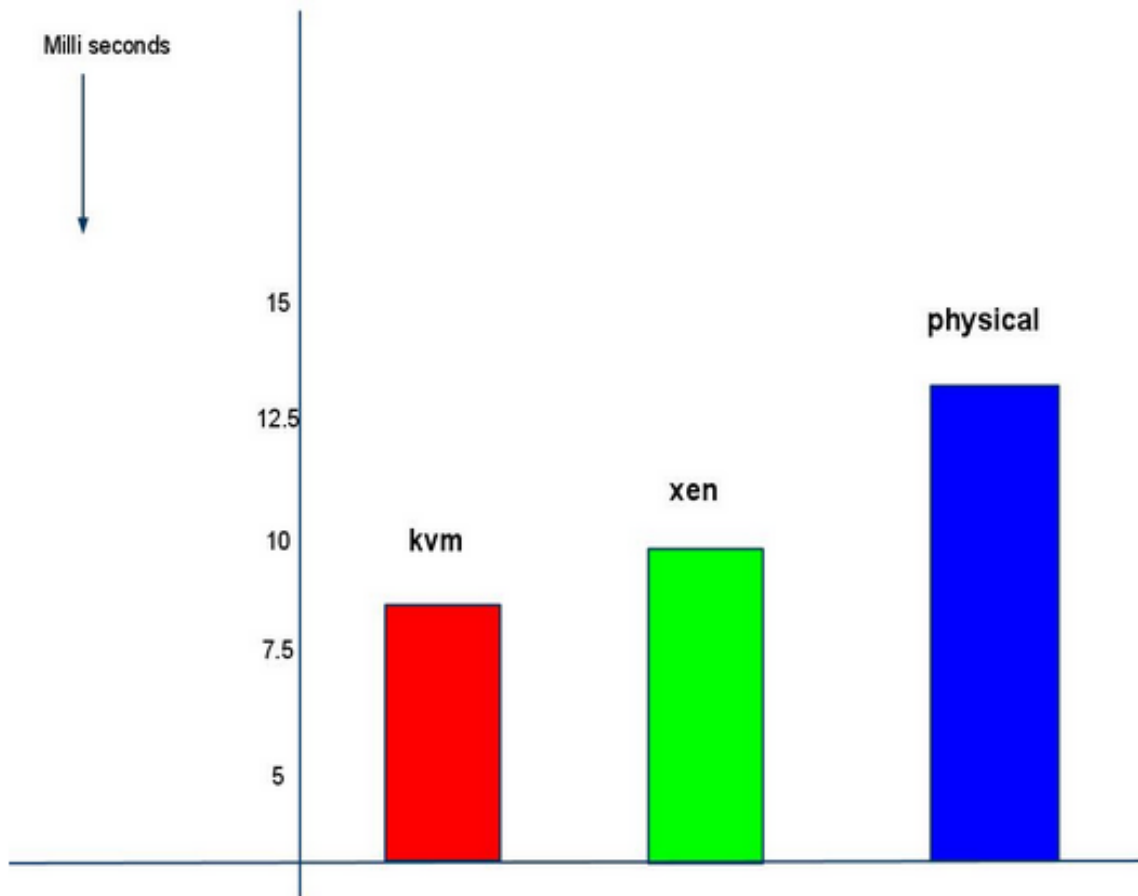
File System Benchmarking



Random Seeks

This test runs SeekProcCount (currently 4) processes in parallel, doing a total of 4000 lseek()s to locations in the file computed using by random() in bsd systems, drand48() on sysV systems. In each case, the block is read with read(2). In 10% of cases, it is dirtied and written back with write(2). The idea behind the SeekProcCount processes is to make sure there's always a seek queued up. The size of the file has a strong nonlinear effect on the results of this test.

Random access time

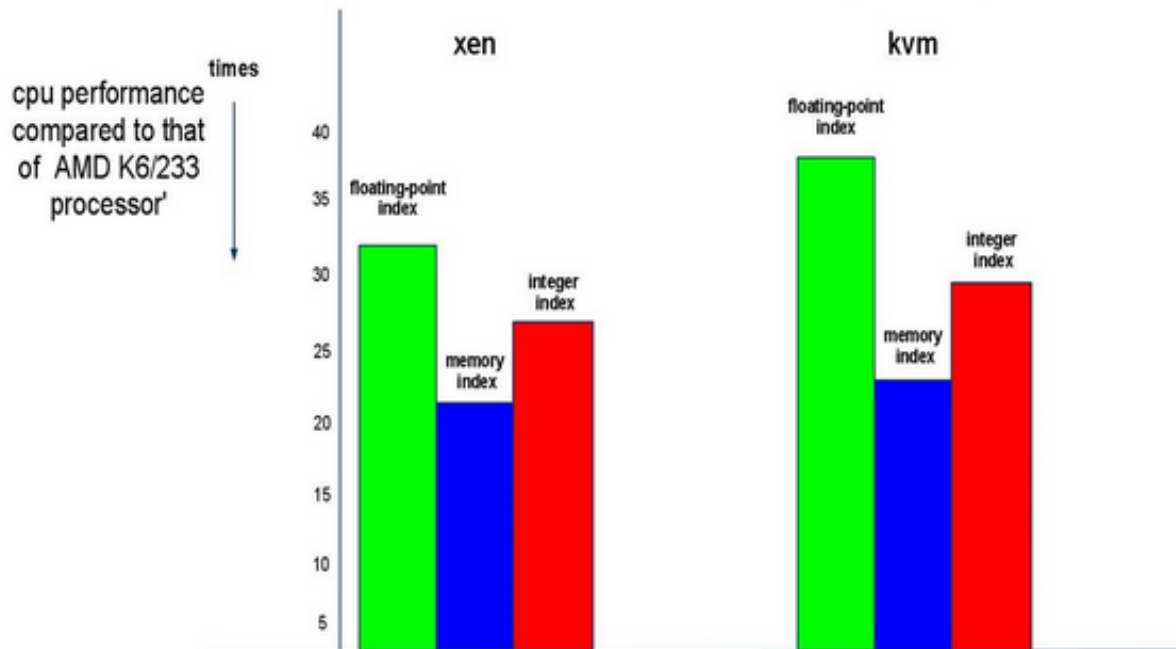


CPU Benchmark

In this test we are trying to see how efficiently cpu works in various virtual machine environments. The benchmark program run (on most machines) and compares the system it is run on to two benchmark systems (a Dell Pentium 90 with 256 KB cache running MSDOS and an AMD K6/233 with 512 KB cache running Linux). If you run all the tests the BYTEmark will also produce two overall index figures: Integer index and Floating-point index. The Integer index is the geometric mean of those tests that involve only integer processing while the Floating-point index is the geometric mean of those tests that require the floating-point coprocessor.

CPU BENCHMARKING

[Baseline (LINUX) :AMD K6/233*, 512 KB L2-cache, gcc 2.7.2.3, libc-5.4.38]



Above graph shows how many times the cpu operations in virtual machines are better than the base system. As you can see it from the above graph, cpu performance in kvm virtual environment is slightly better than that in xen environment.

Conclusion based on test results

In both the file system and cpu benchmarking, kvm system shows slightly better performance than the xen machine. I am not trying to make the point that kvm is better than xen, but just showing how they performed in various test environments that have been setup using applications like bonnie and nbench.