IPC: Measuring speed of Virtual Machine/Application

Janardhana Reddy Naredula

Abstract

Virtual machines or Application in the cloud typically run on existing general-purpose traditional operating systems(OS) such as Linux. The speed at which the instructions are executed by the CPU depends on various factors like memory speed,degree of locks usage, syscalls, hyperthreading..etc. This paper describes the details of all the parameters governing the speed at which instructions are completed/executed inside the cpu. Linux and Jiny kernels[1] are used to describe these parameters. Speed of cpu can be measured in the units of Instruction Per Cycle(IPC), higher value of IPC means better speed or cpu is running at high speed.

1 Introduction

IPC depends on various parameters. It varies from approx 0.01 to 4.50. The following are different type of workloads, These workload described below are homogenous, But In the actual application workload will be heterogeneous containing combination of all the below workloads:

1. **Memory Intensive**: writing or reading large amount of memory causes the IPC to slow down. Here it depends on the size of the memory and page size. If the page size is bigger and memory is smaller, the speed will be better. Speed is proportional to (pagesize/memory size accessed).
2. **Syscall Intensive**: During syscall, some of the costly instructions will be executed, due to this IPC will slows down.
3. **Lock intensive**: locking of the memory bus causes the IPC to slow down.
4. **Hyperthread**: Hyper Threading efficiency depends on type of workload on the logical cpus of a physical core. If all logical threads executes cpu intensive workloads then the efficiency will falls down. If the logical threads executes memory/io intensive and cpu intensive, then the instructions from logical threads will get interleaved and get the maximum efficiency.
5. **Interrupts**: -TODO-
6. **Network packet process**: -TODO-
7. **Multicore, IPI,Hlt,..etc** : -TODO-

Applications on the host or Application within virtual machine: The workloads described above and corresponding tests in the next section are very homogeneous in nature,so that test application stresses on a particular workload. But the practical application contain mix of all the

above workloads to various degree, due to this IPC of an real time applications will be combinations of all the corresponding IPC's.

2. Test Setup:

Perf tool[2] is used to to measure IPC on the host. VM are created of different os's like linux and Jiny kernel[1]. Jiny kernel is used in all the below tests, since it as features to disable/enable lot of OS features for the performance tests. The results will be similar to linux VM. Stress test app like "stress" is used to generate various loads like memory,syscall,.. etc inside the vm. and at the same time perf tool on the host will be used to measure the IPC for the specified VM on physical host. IPC value mentioned in the below test is for entire VM not the test application inside VM.

1. Perf tool: Runs on Host to measure IPC of vm(Jiny kernel/linux) in terms of Instruction Per Cycle (IPC).
2. Application inside the Jiny kernel to generate various work loads:
    a. Memory intensive test using stress[3]: memory size from 128k to 500M is used.
    b. Syscall intensive test using stress[3]: calling syscall in a tightloop, and the syscall does very minimal work inside the the os.
    c. Lock intensive test using Semaphore[5]: Two threads in tight loop acquire the lock and unlock.

2.1 Tests:

The speed depends on OS overhead, hardware speed, hypervisor overhead, and the actual app. For the same app, the speed between different os's/hardware will be relatively similar, means speed is mostly driven by the application logic and os configuration.
IPC is measured for vm and application from host using linux perf tool as below:

perf kvm --guest stat -e cycles,instructions -p <pid of vm>
or
perf stat -e cycles,instructions -p <pid of application>

Test#		**Description (commands = speed)**	**IPC score**	**Reason**
1.1		Memory/cpu test with 250 bytes without syscalls: ./stress --vm 1 --vm-bytes 250 --vm-keep=4.60	4.60	Baseline for all test, it is best speed in all the test. It does not have any memory,syscall or cpu overheads, it just does tight accessing the same memory.

2.1	memory	memory 4k pages : touching the memory in tight loop. ./stress --vm 1 --vm-bytes 128k --vm-keep=0.98 ./stress --vm 1 --vm-bytes 30M --vm-keep=0.16 ./stress --vm 1 --vm-bytes 128M --vm-keep=0.15 ./stress --vm 1 --vm-bytes 500M --vm-keep = 0.14 ./stress --vm 1 --vm-bytes 1500M --vm-keep=0.12	0.12-0.16	As the memory size increases, IPC decreases.
2.2		Memory Hugepages: ./stress --vm 1 --vm-bytes 30M --vm-keep=0.19 ./stress --vm 1 --vm-bytes 128M --vm-keep=0.18 ./stress --vm 1 --vm-bytes 500M --vm-keep = 0.18 ./stress --vm 1 --vm-bytes 1500M --vm-keep=0.18	0.18	Gain because of less ptes, due to less TLB misses.
2.3		Hugepages (1 process) + cpu intensive (2nd process) ./stress --vm 1 --vm-bytes 1500M --vm-keep + ./stress --vm 1 --vm-bytes 256 --vm-keep	2.40	This is better than previous test, because 2nd process run at high speed, so the average of two improves.
3.1	syscalls	syscallintesive : 1 process	0.59	This is less when compare to the baseline because of the context switch at every syscall. Here the syscall is almost empty, it does not consume much inside the os.
3.2		syscallintesive : 2 or more identical process	0.49	It is less when compare to above because of extra memory pressure during context switch. In the above memory is not switched since there is only one process.
4.1	spinlock	Semaphore and spinlock test: system calls are very minimum, two threads are in tight spin to acquire the lock. ./sem4 100000000 10000000000 =1.16	1.16	It is less when compare to the baseline because it does the tight spin with neighbouring thread.
5.1	Hyperthreading impact	Two vcpus from different vm's on the different core = 4.6 (same as test1) ./stress --vm 1 --vm-bytes 250 --vm-keep=4.60 Two vcpus from different vm's on the same core, both are cpu intensive: ./stress --vm 1 --vm-bytes 250 --vm-keep=2.3	2.6 to 4.6	50% loss because guest os is unaware of hyperthreading.

3.0 How to make use of IPC score

Application or vm will contain lot of modules or homogenous code, when IPC is calculated it will be average of all modules of the application. If the value of IPC is high or low it does not give any indication. Suppose for some application or module the IPC value is low then it does not indicate that there is room for improvement, because some application/module need some of low scoring IPC interfaces like syscalls,locks ,memory intensive or io ,..etc operations, due to this IPC score will be low. *So the absolute value of IPC does not give any indication, the relative value of IPC provides useful indications for performance improvement.* The following are the different ways, the IPC score can be used to improve the performance:

1. **Comparing IPC score from one Version to another version**: suppose some new features are added to the application that are not performance centric, then the IPC score of new version should not go down when compare to the older version. If the score is low then there will be a room to improve the performance.
2. **Capturing the IPC score module by module within application:** Usually the IPC score for each module may be homogenous as long as nature code is in similar lines, so the IPC score for each module gives the indication which module need to improve or which module contributes negatively in overall performance.

Calculating IPC for each module in application: IPC value can be calculated for entire application accurately. But the exact value of IPC for each module cannot be calculated since it cannot be run as individual process, but approximate value can be obtained by running the application and making code inside the required module active most of the time and rest of the code to a minimum degree, by using the overall IPC value obtained is approximate value for the module.

4.0 Summary:

Area	Difference in IPC	Reason
4K versus Hugepages(2M)	0.12 to 0.18 = 0.06	TLB lookup is fast in hugepages, due to this speed is higher.
Syscall overhead	4.60 to 0.59 to 0.49= 4.10	Switching from userspace to kernel. Slow down because of the following reasons: 1) context switch from userspace to kernel 2) when there are more than 1 processes, then additional slow down due to memory switch .
Spinlock overhead	4.60 to 0.83 = 3.80	Special instruction in implementing spin lock that lock the memory bus, due to this there is slow down.
Hyperthread friendliness	2.3(different cores) to 4.6 (same core) = 2.3	vcpu will be slowed down by 50% if the vcpu of different vm's falls inside the same core, assuming both the apps are cpu intensive loads. Means the app can run twice the speed if vcpu is alone inside the physical core and the load is cpu intensive.
Networking	TODO	TODO
Interrupts, IPI	TODO	TODO

References:

[1] Jiny Kernel: https://github.com/naredula-jana/Jiny-Kernel
[2] Perf: source code inside linux kernel repo
[3] stress test tool: http://people.seas.harvard.edu/~apw/stress/
[4] Paper from google: http://www.e-wilkes.com/john/papers/2013-EuroSys-CPI2.pdf
[5] semaphore test: