IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Optimization of Disk I/O Performance in Xen Virtualization

Dr.Rajendra H. Bele
Dept.of Comp.Sc. National Defence Academy Pune-411023

1. Abstract:

Virtualization is a prime factor in Cloud Computing; there are various virtualization technologies existing in the industry, nowadays virtualization is implemented with good performance benefits. It has achieved, very good overall performance, but it shows deflection in the system and IO performance concerning the workload running on virtual machines. If the VM has CPU intensive Workload then it produces considerable overheads in system performance if VM has a mix (CPU and IO intensive) type of workload, it produces a bit more overhead. However, IO intensive workloads like databases still suffer from performance degradation in virtualization. In this paper, we have analyzed disk IO handling mechanism in para-virtualization technology Xen. We have conducted experiments with database workload measured performance with different IO scheduling strategies and proposed a framework for improving disk I/O performance for database workload in a virtualized environment, which consists of a possible configuration of Linux IO scheduling strategies according to its features. The key idea is to make disk I/O requests handled by the correct scheduler instead of default scheduler shows better performance in IO Wait and read and write operations.

Keywords: I/O virtualization; Para-virtualization; file system; VM, VMM; I/O stack

2. Introduction:

Virtual Machine Monitor is a layer of abstraction between hardware resources and applications running on it. It is known as Hypervisor. Hardware resources like Memory, CPU, Disk and N/W, shared among all virtual machines. Hence the existing hardware is efficiently utilized. Thus, virtualization reduces hardware cost. Advantages like live migration, disaster recovery, high availability are also offered by Virtualization Technology for Data Centers in IT industry.

This paper has focused on open source database workload and Disk I/O virtualization. We measured the performance of I/O virtualization and analyzed the impact of I/O scheduling on Disk I/O in a virtual environment. Paper has organized in five different sections; the third section is experimental setup to measure the virtualization performance overhead. Section four is about disk I/O activity in the Virtual environment, section five experiments to check I/O performance in different virtual environment. Section five is comparative study and analysis of results, section six is a conclusion and future work.

3. Experimental Setup Part-I

In this section details are provided about the configuration of the Physical System and Virtualized System, Hypervisor, Database Application and the benchmark used for performance evaluation. Details about tools we used to monitor system performance are also provided here.

3.1 Machine Configuration

In the Experimental setup we use separate notebook PC with processor Intel core i5-3230m CPU@2.60Mhz, 1TB Hard Disk, 8 GB RAM, we used same machine for two experiments, one with Linux without virtualization and another with Linux and Xen virtualization. These machines are termed as native system and Virtualized system respectively.

1. Native System (non-virtualized) with Linux O.S. and
2. Virtualized System having host O.S.(Linux) and Xen Hypervisor and One guest O.S. (Linux)

Details of these systems are provided in following sections

3.1.1 Physical System Setup

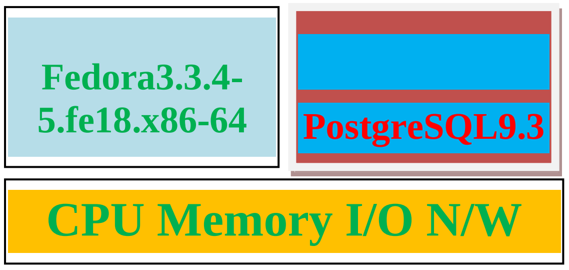

Figure I: Physical System Setup

This is physical machine with two 2.60 GHz Intel Dual core processors, 8GB memory, 1 TB of storage space. This is referred as physical system runs fedora18 with version 3.3.4-5.fe18.x86-64 of the kernel. We have installed latest open source database PostgreSQL 9.3 as a database application along with Fedora18.

5.1.2 Virtual System Setup

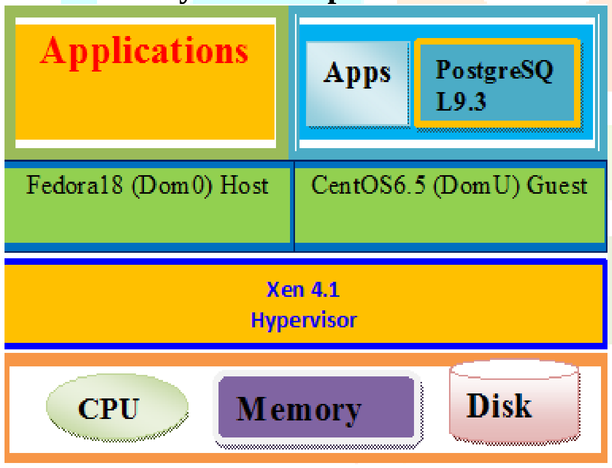

Figure II: Virtual System Setup

Hardware configuration of Physical Machine is kept as it is in Virtual System Setup, Xen 4.1 para-virtualization hypervisor installed on host O.S. (Dom0) Fedora18 with version 3.3.4-5.fe18.x86-64 of the kernel, and we have created guest Virtual Machine runs centOS-6.5 as guest OS(DomU), Here control domain is Dom0, and we allocate it 4GB of memory. We create a single VM (DomU) for running with CentOS-6.5 and postgreSQL9.3 Database system and we give it 3GB of memory and a single 40 GB partition hard disk to DomU contains [2] PostgreSQL9.3 and University Database. Dom0 and DomU use one virtual CPU each, and these virtual CPUs are mapped to different physical CPUs on the machine, which ensures that Dom0 has enough resources to do its work without controlling DomU or competing with it for resources. We refer this system in our experiments as the Virtual system. [4]

4. Disk IO Activity in Xen Para virtualization:

Xen architecture has developed from Hypervisor contained direct device access. This is split I/O architecture, primary goal of this architecture is to provide isolation from device drivers. Virtual machine in Xen do not have direct access to hardware devices, each device driver is organized to run in driver domain (Dom0) which has a backend driver to serve access requests from guest domains (DomU)[13].

In order to manage IO activity, I/O resources are managed by dom0 and provided to the guests, through the use of device abstractions and shared memory. Device access requests initiate at a guest domain and must go through dom0, via Xen. Dom0 is then responsible for handling communication with the actual device through the native operating system drivers. Hardware interrupts are received by dom0 through Xen, and the guest is notified using the event channel as an asynchronous virtual interrupt mechanism. This method is often referred to as Xen's split driver model. Dom0, in this case, is referred to as the device domain. By default dom0 is configured as the device domain, but a separate domain can also be used.

In above experiments We have observed that that there is big difference in performance, concern with IO handling in database workload inside virtual Environment ,In order to find possible cause we studied actual design and architecture of xen para virtualization.

I/O Handling Mechanism in Xen: A virtual machine monitor allows multiple operating systems to share a single machine safely. It provides isolation between operating systems and manages access to hardware resources. [7] Xen consists of two elements: the hypervisor and the driver domain. Hypervisor performs functions such as scheduling processors and allocating memory among guest domains. One of the major functions of the driver domain is to provide access to the actual hardware I/O devices. Hence all I/O traffic pass through the driver domain. [5]

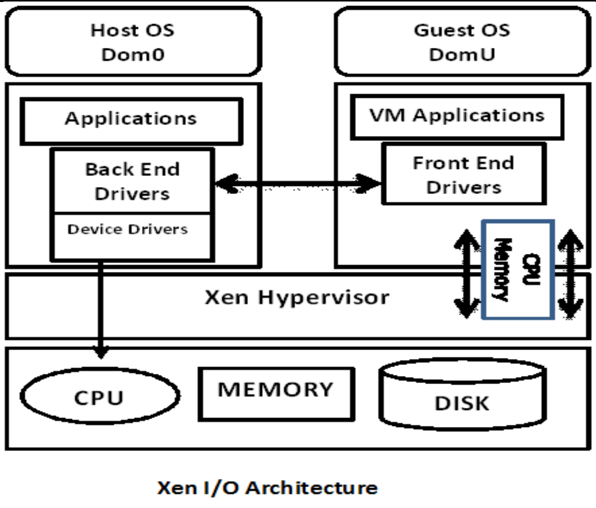

Figure IV: Xen I/O Model

Xen realizes the I/O virtualization mainly through the event channel, the I/O ring and grant table. Event channel mechanism is used between Domain and Domain or Xen and Domain communication mechanism have relatively lightweight implementations. I/O ring is a shared memory provided by Xen for DomainU and Domain0 (IDD) access and it provides two pair of producer-consumer pointers: domains place requests on a ring, advancing a request producer pointer, and Xen removes these requests for handling, advancing an associated request consumer pointer I/O ring is also the main factor to improve performance when Xen communicates with virtual machine.[21]

Xen uses grant table mechanism to achieve efficient transfer of data between DomainU and Domain0 or IDD. The front-end driver produces a request production pointer after receiving I/O request from Guest OS, and puts data into grant table. This Guest OS puts this pointer and grant table references into the I/O ring.[22] Then informs the back-end driver by event channel, the back-end driver takes I/O request and data out from I/O ring and grant table, and converts virtual address to physical address after checking the validity of this request. Then the back-end driver receives I/O request and maps to the physical device through Domain0 or IDD. Finally the device driver completes this I/O request by hardware. When I/O request is completed, the back-end driver would receive notification from Domain0 or IDD, and informs the front-end driver through event channel after putting I/O response into the I/O ring, and then the front-end driver takes out this I/O response from I/O ring and handles it. [23]

In physical system we need to monitor only the physical disk storing the database. However, for the Virtual System (Xen), we need to monitor both the physical disk accessed by Dom0 (on behalf of DomU) and the virtual disk inside DomU, with iostat tool. Observing disk activity at these two levels allows us to distinguish between the data read from the physical disk by Dom0 and the data read from the virtual disk in DomU.

Roles of two domains exists in IO handling, it is two tier scheduling. linux have four IO schedulers having different nature by changing schedulers at different levels, we check performance of these four schedulers for following parameters and compared results in both environment physical and virtual environment. Alternative IO Schedulers Existing in Linux, We can select appropriate scheduler at boot time to Optimize Performance [24].
Application can optimize the kernel I/O at boot time, by selecting one of four different I/O schedulers to accommodate different I/O usage patterns, The I/O scheduler can be selected at boot time using the “elevator” kernel parameter.

*Completely Fair Queuing—
elevator=cfq (default)
* Deadline—elevator=deadline
* NOOP—elevator=noop
* Anticipatory—elevator=as

5. Experimental Setup Part II

By keeping all above configuration (Hardware, Memory, CPU, Buffer and S/W, O.S, Set Of Queries) as it is (Part I Experiments) we conducted same experiments with different schedulers; Linux Schedulers are Used In Para virtualization Mainly there are four Linux IO scheduler’s, noop, anticipatory, deadline, cfq (Default) Now a day’s advanced version of linux replaces anticipatory (as) by completely fair queuing (cfq).

Hence in advanced operating system like fedora18 onwards there are three schedulers available

1. Cfq (completely Fair Queuing) is an I/O Scheduler and default under many Linux Distributors.
2. Noop(Noop) is the simplest I/O Scheduler for the linux kernel upon FIFO concept
3. Anticipatory scheduler (anticipatory) is an algorithm for scheduling hard disk input/output as well as old scheduler replaced by cfq.
4. Deadline(Deadline) It is attempt to guarantee a start service time for request

We have created one virtual Machine having CentOS- 6.7 as guest O.S in it called as dom1, postgreSQL- 9.3 Installed in it executed 12 set of queries and noted system values provided by IOstat Linux tool. We primarily use iostat to monitor disk activity precisely we gather the number of kilobytes read and the number of kilobytes written to physical (and/Virtual) disk. Mainly concentrated on following performance metrics:

1. tps: Indicates number of transfers per seconds that were issued to the device. A transfer is an I/O request to the device. Multiple logical requests can be combined into single I/O request to the device.

2. KB_read/s: Indicate amount of data read from the device expressed in kilobytes per second.

3. KB_write/s: Indicate amount of data written to the device expressed in kilobytes per second.

4. IOWait: Show the percentage of time that the CPU was idle during which the system had an outstanding I/O request.

IOStat provides many system details still we consider only above parameters because we need to monitor both the physical disk accessed by Dom0 (on behalf of DomU) and the virtual disk inside DomU. Observing disk activity at these two levels allows us to distinguish between the data read from the physical disk by Dom0 and the data required by each query and read from the virtual disk in DomU.

We selected different pairs of schedulers for DomU and Dom0 respectively executed queries individually and took readings, to count I/O Activity precisely we stopped services and flushed out buffer in Dom0 and in DomU.

Results displayed below.

Schedulers	Dom0			Dom1		
	tps	kb_reads/s	kb_writes/s	tps	kb_reads/s	kb_writes/s
(cfq, cfq)	5.30	101.53	16.18	0.11	0.85	0.31
(cfqnoop)	10.32	279.14	21.78	0.23	1.08	0.25
(cfq, deadline)	7.48	145.13	11.97	0.25	0.60	0.41
(noop, cfq)	16.98	456.01	22.49	0.59	1.38	0.97
(noop ,noop)	7.53	140.38	14.47	0.27	0.54	0.53
(noop, deadline)	12.15	316.76	23.34	0.37	0.85	0.62
(deadline, cfq)	17.77	488.60	31.84	0.55	1.36	0.85
(deadline ,noop)	11.07	275.36	20.29	0.34	0.95	0.56
(deadline, deadline)	6.34	123.12	17.26	0.17	0.33	0.34

Table1: Readings for Domain0 and Domain1 for Different Schedulers

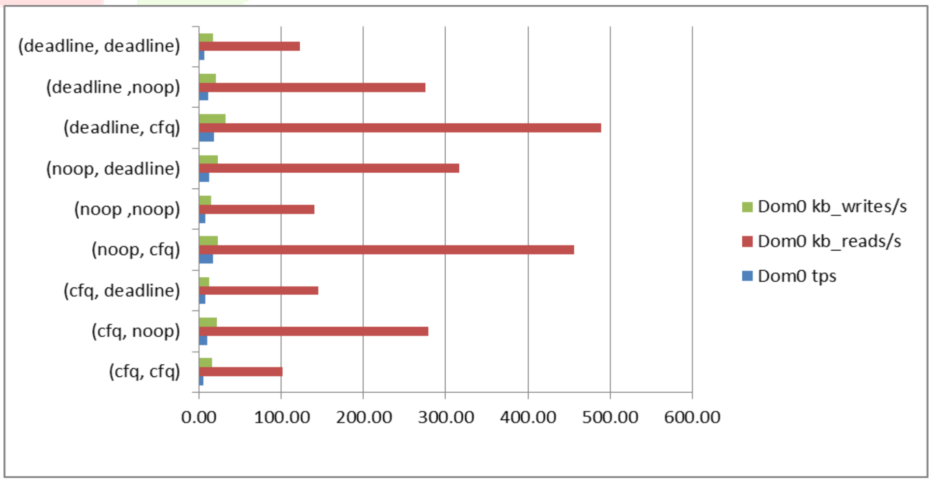

Figure 1: Graphical View for Domain0

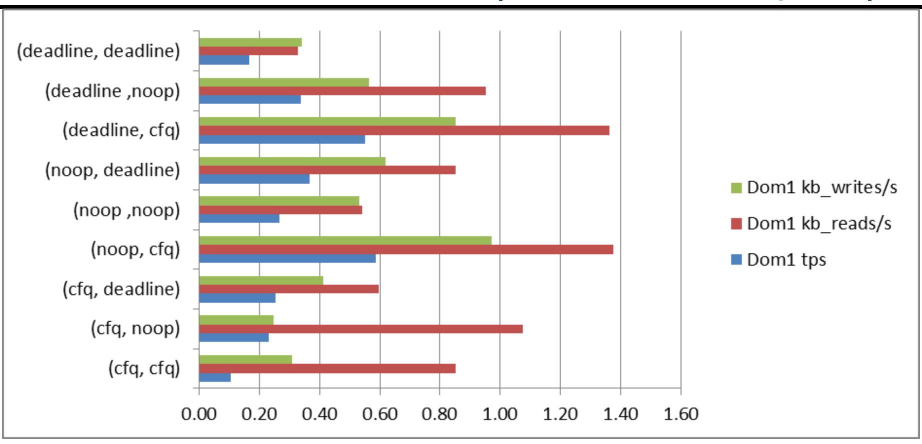

Figure 2: Graphical View for Domain1

5.1 Aggregate Results from All Combinations of Schedulers

Schedulers	user	system	iowait
(cfq, cfq)	0.99	3.69	1.75
(cfq, noop)	2.01	12.76	1.68
(cfq, deadline)	1.46	14.20	0.84
(noop, cfq)	2.27	20.66	1.59
(noop ,noop)	2.20	19.80	0.94
(noop, deadline)	2.28	20.04	1.21
(deadline, cfq)	2.60	21.04	1.63
(deadline ,noop)	2.78	21.02	0.94
(deadline, deadline)	2.49	20.33	0.80

Table 2: Average Results for All Combinations of schedulers

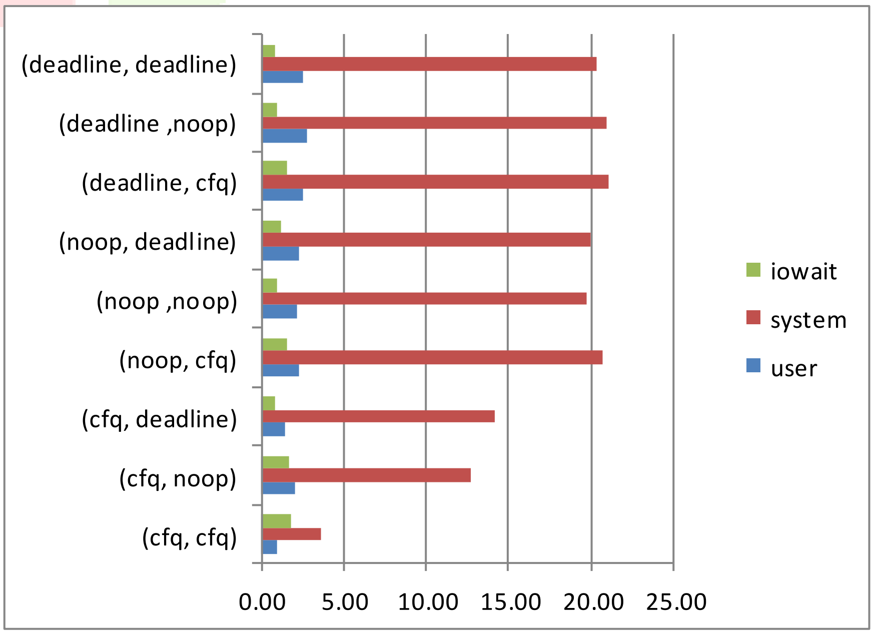

Figure 3: Graphical View for Domain1

6. Interpretations and Conclusions:

1. It is observed that selection of IO Scheduler presents variations in the results of tps(transfers per seconds), the amount of data read(kb_reads/s) and iowait time.

2. iostat with DomU (is Unprivileged Domain) displays results about virtual devices allocated to it. iostat with Dom0 (is privileged Domain) displays results about physical devices

3. Dom0 reads more data than required by DomU hence DomU reads more data automatically, which causes increase in iowait time .It is one of the main reason
Why database applications suffers in virtual environment.

4. Selecting scheduler at Dom0 plays significant role in performance optimisation for databases in virtual environment because it is host OS which provides all type of I/O services to guest OS existing in Virtual Machine.

5. If we observe aggregate results of all combinations of schedulers it interprets that it is deadline scheduler which provides optimum values for given parameters i.e. tps(6.34),kb_reads/s(123.12),kb_wrn/s(17.26),io wait(0.80)
Hence deadline scheduler is best for current scenario to reduce performance overhead on I/O handling

6. Default Scheduler do not provide best performance results in virtualization, These different schedulers having different characteristics which produces variable results in performance ,for this database state deadline scheduler shows overall optimum results still some queries perform well for other schedulers also.

7. I/O is a critical component for high performance applications which is particularly observed for high disk- IO workloads such as databases. Hence it is considered that I/O throughput was a limitation of virtualization. It happens due to workloads with large amounts of I/O. It is necessary to study characteristics of workload and configure scheduler IO Scheduler to enhance performance

7. References:

[1] Jeanna N. Mathews;Eli M.Dow;Todd Deshane;Wenjin Hu; ”Running Xen hands On Guide to the Art Of Virtualization”, Prentice Hall; 1 edition , April 2008,ch.1-3.
[2] Chirs Takemura and Luke S. Crawford, “The Book of Xen”, 2009.

[3] Eucalyptus open-source software, ”http://www.ecualyptus.com”

[4] David E. Williams, Juan Garcia, Simon Crosby ,” Virtualization with XenTi: Including XenEnterprise TM, XenServer TM, and XenExpress TM”, Syngress Publishing, Inc.,2007 chapter 1-10.

[5] Ahmed A. Sorory,Farooq Minhasy,Ashraf Aboulnagay,”Automatic Virtual Machine Configuration for Database Workloads”, Proceedings of the 2010 ACM SIGMOD international conference on Management of data , Pages 953-966 .

[6] Umar Farooq Minhas, Jitendra Yadav, Ashraf Aboulnaga, Kenneth Salem, ”Database System on Virtual Machines: How Much Do You Lose”, “International Conference On Data Environment”ICDE 2008.

[7] Dan Kusnetzly,”Virtualization A Manager’s Guide”,O’Really Publications, 2012

[8] Chaganti P.,” Xen Virtualization”, Packt Publishing Ltd, 2007.

[9] Tsuyoshi Tanaka, Toshiaki Tarui, and Ken Naono ,” Investigating Suitability for Server Virtualization using Business Application Benchmarks”, VTDC '09 Proceedings of the 3rd international workshop on Virtualization technologies in distributed computing ,2009 ACM, Pages 43-50.

[10] Diego Ongaro, Alan L. Cox Scott Rixner, “Scheduling I/O in Virtual Machine Monitors “ , Proceedings of the fourth ACM SIGPLAN/SIGOPS international conference on Virtual execution environments, Pages 1-10.

[11] https://en.wikipedia.org/wiki/Mendel_Rosenblum.

[12] Kihong Lee, Dongwoo Lee, Young Ik ,”A Para virtualized File System for Accelerating File

I/O" , 2014 International Conference on Big Data and Smart Computing (BIGCOMP) Publisher: IEEE ,Page(s):309 – 313.

[13] Umar Farooq Minhas; Jitendra Yadav, Ashraf Aboulnaga,"Database Systems on Virtual Machines: How much do You Lose?" 24th International Conference on Data Engineering Workshop ICDEW, 2008 IEEE, Page(s):35 – 41.

[14] Min Lee, A. S. Krishnakumar, P. Krishnan, Navjot Singh, Shalini Yajnik," XenTune: Detecting Xen Scheduling Bottlenecks for Media Applications", Published in: Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE, Page(s):1 – 6.

[15] David Chisnall,"The Definitive Guide to the Xen Hypervisor", Prentice Hall, 2008 ch.1-6.

[16] Mike Hogan," Head in the Cloud", July 2013, https://www.scaledb.blogspot.in/"

[17] Silberschatz, korth and Sudarshan ,"Database System Concepts",6 th Edition ,Chapters.1-10.
[18] Charles David' Graziano," A performance analysis of Xen and KVM hypervisors for hosting the Xen Worlds Project", A Thesis Submitted to Iowa State University, 2011.

[19] Walt Hubis, Rob Peglar,"Storage Virtualization, Why ,Where and How", http://www.snia.org/education/tutorials/virt-app.

[20] "VMware vSphere Hypervisor – Install & Configure", https://my.vmware.com/en/web/vmware/evalcenter.

[21] "PostgreSQL Database and Schema Management", http://www.postgresqltutorial.com/postgresql-administration.

[22] Avi Kivity, Yaniv Kamay, Dor Laor, Uri Lublin, Anthony Liguori, "Kvm: the Linux Virtual Machine Monitor", Proceedings of the Linux Symposium, 2007, Page(s) 225- 230.

[23] Jeremy Sugerman, Ganesh Venkitachalam, Beng-Hong Lim, "Virtualizing I/O Devices on VMware Workstation's Hosted Virtual Machine Monitor", Proceedings of the USENIX Annual Technical Conference, 2001 Page(s) 1-10.