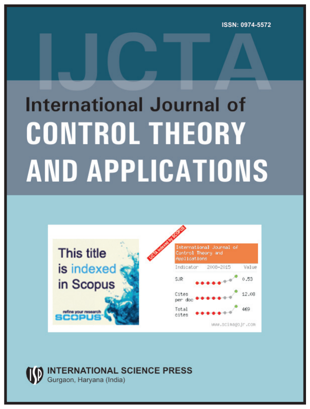

**International Journal of Control Theory and Applications**

ISSN : 0974-5572

© International Science Press

**Volume 9 • Number 44 • 2016**

# A Survey of Increasing I/O Latency in I/O Stack

**Ashutosh kumar Singh[a], Suhas Haribhau Patil[b] and Naveenkumar Jayakumar[c]**

*[a]Department of Computer Engineering, Bharati Vidyapeeth Deemed University, College of Engineering, Pune, Maharashtra, India. Email: ashutosh9316@gmail.com*
*[b,c]Prof. Department of Computer Engineering, Bharati Vidyapeeth Deemed University, College of Engineering, Pune, Maharashtra, India. Email: [b]shpatil@bvucoep.edu.in; [c]naveenkumar@bvucoep.edu.in*

***Abstract:*** Day by day increasing the size of data results from complexity in modern computing systems, a balanced approach of performance and manageability of data becomes a serious issue for achieving high-performance computing. The virtual machine provides many features that help in large-scale computing systems. I/O stack gets evolve by various reasons like higher software capability, insufficient memory management to handle various I/O requests, etc. There are many causes due to small manageability and low flexibility due to which present solutions cannot eliminate these reasons and hence it is required the new solution to remove this defects in I/O latency.

***Keyword:*** I/O Stack, I/O Latency, I/O Packet, Burst Buffer.

## 1. INTRODUCTION

**DEFINITIONS:**

**I/O STACK:**

**I/O:**stack is chain layer structure of drivers in which I/O request flow through different drivers like HBA driver, SCSI driver, etc

**I/O LATENCY:** The time taken to complete one I/O request in I/O stack is called I/O latency

**I/O REQUEST (PACKET):** The operations which are performed in I/O stack by I/O manager

**BURST BUFFER:** It is mid way high speed layer between application file system and parallel file system for high speed data execution in I/O stack

**Working:** A virtualized I/O stack is the layered structure of drivers. A request is sent from the application to target storage device. The file descriptor is used to target correct file system. After the file system translates the read request and it enters I/O media which represents the target content. The request travels down to storage driver after I/O media, and the storage driver provides protocol defined the format and passing the request below to SCSI device. SCSI device issues the request to target the device and pass the request to the logic unit number which is created by SCSI target device to assign its address. Then the request reaches to HBA driver who passes to the appropriate storage.The main difficulty is to that the traveling of the I/O requests two times first is physical I/O stack and then virtualized I/O stack. This leads to latency times increases. So, my approach to minimize the I/O latency in virtualized I/O stack

In this paper, we study the following

1. What are the virtues of hardware approach in I/O bottleneck?
2. What are the various methods of software based approach of improving I/O latency?
3. What are the existing literature survey of software based approach?
4. A neoteric approach to improve in I/O latency

## Types of Parameters which Affect I/O Latency

1. *Virtualizations:* This situation occurs when we create multiple VMware on single volume disks. As a result, it creates the tremendous amount of pressure on disk and application related to disc while VMware was running I/O intensive applications. If the VMware have small storage space the user wants to fit the data, then it is not possible for the VMware to accommodate all user requests
2. *Application*: Application which has heavy I/O packets cause I/O bottleneck situations when a large user base tends to access this application then process of slowdown take place
3. *I/O response time:* increase response time cause I/O bottleneck. When there is a queue, it causes an increase in latency. A busy storage drive is an also a reason for the increase in response time.
4. *Drastic RAID design:* This situation occurs when few redundant array of multiple handles many workloads. It is very evident when the storage system can’t process the amount of work that a user was trying to run.

## Solving These Parameters

### *A. Hardware Approach*

(i) *Hardware buffer:* It is some samples data the CPU catches and process the data at one time before split back for monitoring and recording. Large buffer size gives you better computer performance to process amount of data.

Some of the hardware cases:

- To eliminate the queue of I/O latency with the help of consolidation tools like CiRBA and PowerRecon which performs modeling for prospective virtual machine physical host server platforms. It analysis workload environments and available resources suggests best possible way use it.
- Tools such as VMware Inc’s Distributed Resource Scheduler (DRS), Virtual Iron Software Inc’s Capacity Management which utilizes in handling available resources.
- Then other tools are we can use Mellanox IO adapter for virtualisation based on this we can decide to track I/O latency which file is the busiest from a read and write process.

- We can eliminate by providing faster network and storage I/O by the hypervisor which can reduce latency by interconnecting point to point connection of node.
- Tools such as Akorri Network Inc.'s Balance Point and Tek-Tools Software Inc.'s Profiler which tracks I/O bottleneck.

(ii) *Playback buffer:* This based on RAM using a buffer for the hard drive that temporarily stores data that comes form and to the hard drive. This buffer referring those comes from the PCI card or USB and when the hard drives record it. The higher the buffer more data will record and received.

(iii) *Pro tools ram:* This deals with the adding more RAM which increases the performance of the computing system and leads high efficiency in processing of data.

### *(B) Software Approach*

(i) *Software buffer:* It is the area where the temporary data is stored in physical storage media when it moves from one place to another. it can be used with the processes within the computer, and it can be implemented in fixed memory of hardware and virtual memory of software. Typically buffer uses faster RAM to stores temporary data the main cause of buffers has used the time at which the data is being received and the rate at which the data is recorded.

(ii) Buffers are often used in joining with I/O to hardware, such as disk drive, sending or receiving data to or from a media network

**Examples:**

- The buffer is used as controls in DOS.
- The buffer between a serial port and a MODEM

**Comparison Between Hardware Approach And Software Approach**

| *Hardware Approach* | *Software Approach* |
|---|---|
| Less access time by hard disks | Much faster access time by RAM |
| Higher cost price | Less cost price |
| Main purpose is reduce access to both fast and slow storage | Main purpose to reduce access to the underlying slow storage |
| It does not do have very large capability multiple reads of same data | It does have very large capability multiple reads of same data |
| Because of not multiple read or writes feature they cannot be combined for batch processing | Because of not multiple read or writes feature they cannot be combined for batch processing |

## 2. LITERATURE SURVEY

As we do literature survey of various proposed models every time defects are found. This paper proposed a strategy that has been implemented by StarWind Software that shows considered promise for dealing effectively with the I/O Blender[1]. This paper does not examine the cause of poor application performance. The complicated task is doing the movement of one virtual host to another host. The primary cause of the problems the hosting environments for server configuration[2]. For server configuration, it has to accommodate maximum workload that is suitable for hosting virtualizations This paper explain does not explain the resource provisioning requirements to virtualize NIC and HBA. The idea of direct of storage I/O traffic create barriers to I/O performance describe by M.Principal in 2014[3].

It described between high-performance storage requirements and concepts from the networking space and identifies common high-performance I/O properties. This paper does not guarantee of giving performing I/O stacks. This paper does not explain the reconfigure of LUN. Configuring LUN for storage provisioning resource is necessary. This paper does not explain the rate of arrival of I/O packets at I/O devices. This paper does not describe the improvement in PCI interface and PCIe interface which results in the mismatch between CPU and I/O speeds. The result packet arrival is not determined which cause each network packet goes through an additional layer which results in an additional overhead. This paper does not explain the buffer management, I/O header processing, and I/O channel transfer describe by A. Trivedi, P. Stuedi, B. Metzler, R. Pletka, B. G. Fitch, and T. R. Gross.[4]

These paper also focuses on synergies between high-performance storage requirements and concepts from the networking space and identifies common high-performance I/O properties. This paper does not guarantee of giving performing I/O stacks.[5] This paper does not explain the parallel interface for high performance. This paper lacks in explaining the placement of driver software in the VMM. This paper does not define the error handling and device configuration to a avoid additional overhead.[6] It increases the Netfilter and bridge cost. Keeping the guest buffer in isolation stage would also cause which creates many copies of the I/O packet which results in processing of the packets and it also increases the CPU cycles. This paper explained the dedicated storage drive for synchronizing the propagation delay.[7]

This paper big data resources and their processing management capabilities. This paper explains the existing algorithms of I/O linear temporal logic[8].This paper describes the rationale of detection of all excepting cycles which is called as detecting all complete cycles. (DAAC) [9]. This paper does not explain correct semantics and interfaces in the processing of detection. This paper does not explain the criteria for detecting all cycles. This problem of detection sometimes violates of the linear logic of I/O subsystems. This paper demonstrates the computation technique and path management technique. This paper does not explain lowering of I/O complexity and additional overhead. Sometimes in path control causes space explosion problem

This paper focuses on improving reliability and security through device isolation using hardware assisted remapping and I/O performance and availability by direct assign of devices. This paper does not. This paper does not explain the management of resources which may help in application compatibility and reliability. This paper does not describe additional levels of manageability, security, isolation and performance. This paper explains the protection by restricting DMA.

It explains the architecture which assign one or more I/O devices for security domains. This paper lacks to explain the mechanism of accessing certain memory regions via address translation table. Sometimes in remapping structure, some I/O devices are not assigned described by B. T. W. Burger and A. March in 2016. [10]

This paper proposes the new algorithm the new algorithm has a lower I/O complexity than I/O efficient model checking algorithms, including detection accepting cycle, maximize allowing predecessors, and long depth-first search. This algorithm is not suitable for smaller systems, andit is only for large scale systems. This paper introduces an I/O efficient algorithm which is used for large-scale systems to lower I/O complexity and to reduce time. This paper does not explain the free space management. This paper describes the cache duplication detection technique which increases additional time and additional overhead both. This paper does not describe the processed I/O operation so as to reduce time efficiency described by Wu, L., Huang, H., Su, K., Cai, S., & Zhang, X. in 2015 so we overcome this types of demrit this paper proposed CORI burst buffers.

| *Titles* | *Authors* | *Merits* | *Demrits* |
|---|---|---|---|
| Eliminating the I/O Blender Effect and Realizing World Class Storage Performance in Virtual Storage Environments | M. Principal | Dealing effectively with the I/O Blender. | Does not explain the resource provisioning requirements to virtualize NIC and HBA |
| An Active Storage Framework for Object Storage Devices, | Michael T. Runde Wesley G. Stevens Paul A. Wortman John A. Chandy | Used Object file system OSD services and RPC services Proposed Execution engine | Unexplained Integration between OSD and Active storage. Asynchronous but can only operate on single object. |
| Unified High-Performance I/O : One Stack to Rule Them All.” | A. Trivedi, P. Stuedi, B. Metzler, R. Pletka, B. G. Fitch, and T. R. Gross | Focuses on synergies between high-performance storage requirements and concepts from the networking space | Does not explain the parallel interface for high performance |
| Database-Aware Semantically-Smart Storage Acm Sigmetrics 2006 | Muthian Sivathanu, Lakshmi N. Bairavasundaram | Find Communication Cost. Data transfer vs Application offload. Fused logic with file systems | Proved for majority of write operation. Needs modification to DBMS |
| Intel ® Virtualization Technology for Directed I/O : Enhancing Intel platforms for efficient virtualization of I/O | B. T. W. Burger and A. March | Focuses on improving reliability and security through device isolation using hardware assisted remapping and I/O performance | Does not explain the management of resources which may help in application compatibility and reliability |
| I/O Architectures for Virtualization | M. Mahalingam | Explains the performance in resembles to CPU utilization | Does not explain the TCP offload and DMA operations |
| I/O Virtualization Decoupling a logical device from its physical implementation offers many compelling advantages | M. Rosenblum and C. Waldspurger | This paper focuses on the decoupling of logical device of heavily workload at the host. | This paper does not explain the of offloading the heavy workload if high I/O intensive applications |

## 3. PROPOSED SCHEME

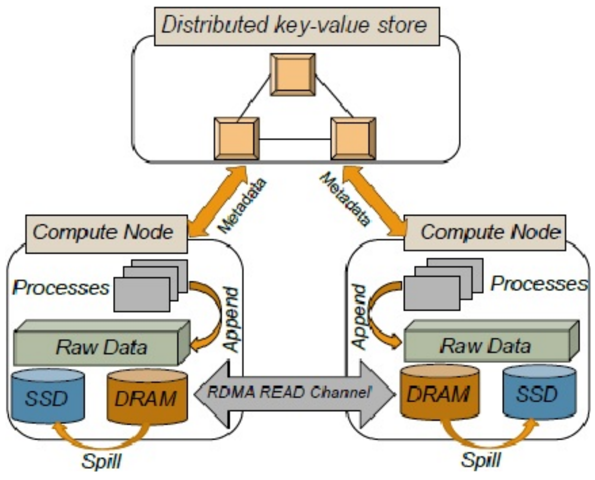

**Figure 1: BurstFS Architecture**

Our proposed system works on basically 3 modes:

1. Cleint Side
2. Server Side
3. Fault Tolerance

1. **Client Side:** In this, we will examine in 2 steps balancing offloading data, coordinated data flushing from the nodes. Balancing o data can be by offloading them to under utilized server. In most of the cases, the burst buffer has noncontiguous segment which has shared files which have 2 phase I/O. In first I/O all processes figured out sized of the data and in 2nd phase I/O the shared files are partitioned into n files.
2. **Server Side:** In this two steps will be done synchronization in the case of failure and member detection. Synchronization is done to maintain consistency among all servers using protocols using sending and receiving message to all servers and clients, and failure detection can be done by server side stabilization to avoid broadcast of the message.
3. **Fault Tolerance:** Fault tolerance can be achieved by data replication. in case of failure or recovery the data copy can be used in place of defective data

## 4. CONCLUSION

In this paper, we concluded the survey on the merits and demerits of an existing system of the existing system of increasing the I/O latency in virtualized stack. In this types, what are the limitation of hardware approach and what are the advantages of software approach and the points that software approach is better than hardware approach.

Hence e can conclude we can use burst buffer approach to minimize I/O latency execution, failure detection and better approach than other existing systems

## Acknowledgment

I would like to thank to my project guide, research coordinator, head of department, all teachers staff, parents and friends for their support and confidence in me.

## REFERENCES

[1] Julian Satran, Leah Shalev, Muli Ben-Yehuda, ZorikMachulsky," Well-Architected Way to Do Scalable, Secure and Virtualized I/O", *IBM Haifa Research Lab, Haifa, Israel,43,1-6,2014.*

[2] Mendel Rosenblum, Carl Waldspurger," I/O Virtualization",*acmqueue,11,1-17,2011.*

[3] Kihong Lee, Dongwoo Lee, Dong Hyun Kang and Young Ik Eom, A Paravirtualized File System for Accelerating File I/O, *College of Information and Communication Engineering, Sungkyunkwan University, Suwon 440-746, 2015.*

[4] Paul Willmann, Scott Rixner, and Alan L. Cox,"Understanding the sequences of DMA Protection Strategies for Direct Access to Virtualized I/O Devices", *Rice University,15-28,2013.*

[5] Nadav Amit, Muli Ben-Yehuda, Ben-Ami Yassour, ".IOMMU: Strategies for Mitigating the IOTLB Bottleneck", *IBM Research,1-12,2014.*

[6] White paper, "Eliminating the I/O Blender Effect and Realizing World Class Storage Performance in Virtual Storage Environments", *Toigo Partners International LLC,1-12,2014.*

[7] Animesh Trivedi, Patrick Stuedi, Bernard Metzler, Roman Pletka, Blake G. Fitch2 and Thomas R. Gross,"Unified High-Performance I/O: One Stack to Rule Them All", *IBM Research, 1-6, 2013.*

[8] Fernand Lone Sang, Vincent Nicomette and Yves Deswarte "I/O Attacks in Intel-PC Architectures and Counter measures", *LAAS-CNRS – Toulouse, France, 1-33, 2011.*

[9] Mallik Mahalingam, "Understanding suitable architectures I/O Architectures for Virtualization".*R &D –Networking,1-24,2006.*

[10] Chao Chen, Michael Lang, Latchesar Ionkov,"Active Burst-Buffer: In-Transit Processing Integrated into Hierarchical Storage",Los Alamos National Lab, 1-10, 2014.