RESEARCH ARTICLE OPEN ACCESS

A Model to Improve I/O Request Workflow in I/O Stack

*Ashutosh Kumar Singh [1], S.H. Patil [2]
Department of Computer Engineering, College of Engineering ,Bharati Vidyapeeth Deemed University Pune – India
Corresponding Author: Ashutosh Kumar Singh

ABSTRACT
Burst buffer is high bandwidth layer which is present between computing nodes and parallel file system which provide high bandwidth for improving the computational performance in I/O stack. However the existing model system have partially embraced the full potential of improving high i/o performance in i/o stack. In this paper we are simulating the burst buffer for improving the i/o latency in i/o stack by performing read and write operations.
Keywords: latch, burst buffer, i/o latency, i/o stack, SSD

Date of Submission: 26-06-2017 Date of acceptance: 9-10-2017

I. INTRODUCTION

In the field of high i/o performance the i/o gap between computing resources and disk storage is increasing day by day like data transferring speed of i/o nodes of Trinity system. People can quickly spot the asynchronous between the SSD with weak CPU and limited resources and the powerful PC with great amount of ram and large amount of computing resources available to it. Making some kind of cooperation interfaces between the two sides can help us to solve our problem of insufficient of memory resources inside the i/o stack with the help of SSB device

II. PROPOSED ARCHITECTURE

This section tells to the proposed methodology how it works for i/o request execution in i/o stack. The burst buffer is high speed layer which present between computing nodes and parallel file system. Fast SSD are used for enhance better performance

The flash memory have limitations over writing data and cycles per head that is why controller are used that has flash transition layer(software) that manages the flash memory chips. The flash memory are accessed normally block devices. The memory area is divided into blocks and each block contains of 128 pages .pages of size start from 512 bytes to 2k , 4k etc

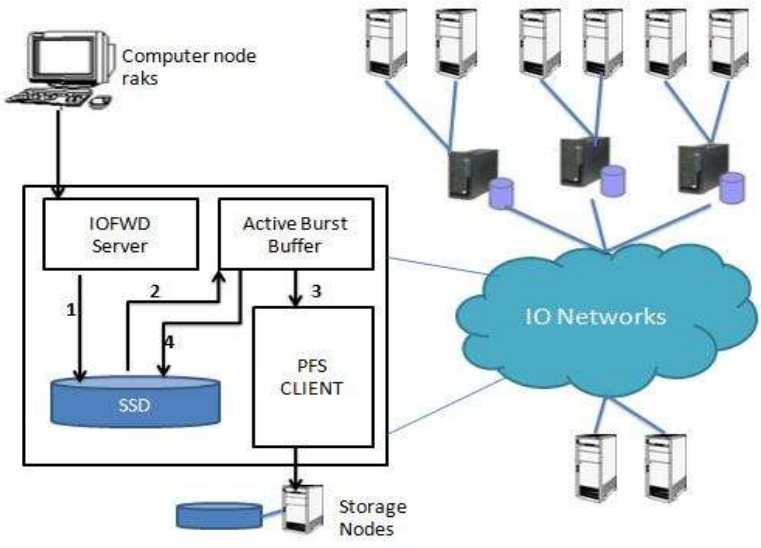


Fig 1.1 The proposed architecture

III. WORKING –

Our model based on works between computing nodes and parallel file system in which we simulate the burst buffer with the help of code. The burst buffer is attached to SSD and SSD device essentially contains two elements (1) controller and (2) flash memory. The controller is responsible for making a bridge of the flash memory components to the host computer. On the front-end the controller connects to computer bus using SATA, SCSI or other interface. On the back-end the controller it handles the demerits of flash memory chips.

The limitation of working with flash becomes complicated because the flash memory which support limited number of deleting cycles and no fine level of detail in a set of data. will be gained when overwriting data operation take place. Therefore the controller implements software layer called whose responsibility is to manage the flash chips and overcome the demerits of the flash memory.

This software layer has two main functions. Firstly it creates abstraction of range of pages which can be read, written and overwritten just like in usual hard disk device. Second, it manages to arrange data so that delete and re-writes operations are distributed across the flash memory. When the SSD receives I/O request from SCSI interface it needs to perform read operations mapping entries from the flash memory

In general latches are saved result of some computation which needs to be cached. Latches are an old caching technique used to speedup computations. Most of the time the latches used for no repeating computation which will lead to increase in performance. In our case the SSD device will send hint with mapping computation result related to particular LBA.

The interfaces which are present at the host part will cache these latches. Next time when the host issues I/O request with logical block addressing matching one of the latch stored in the host, it will send the latch using out of band channel and then invoke the I/O request itself. The SSD then can validate the hint information and use it to process incoming request without overhead of mapping computation.

The first stage of the mapping is of data sets in the SSD's RAM and the second layer is stored on the flash memory. For each I/O request the software layer needs to locate the actual physical address on the flash chip. This is done by using the first layer of the mapping in the RAM to locate the second layer on the flash chip. Then reading mapping entry from flash memory where second layer of mapping is stored.

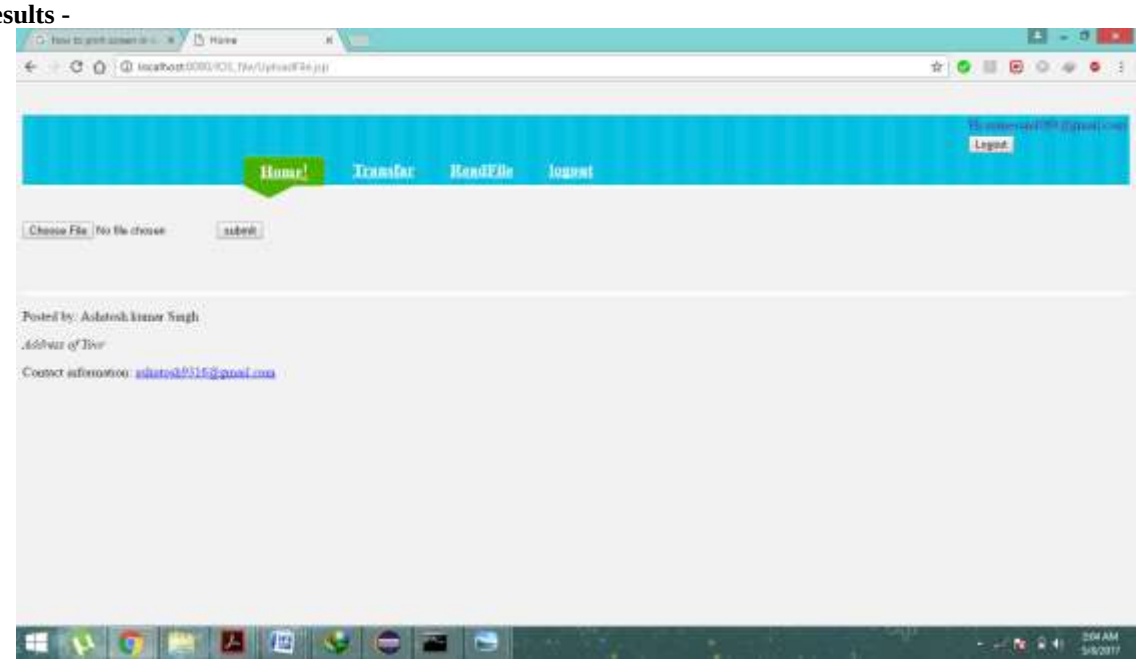
SCHEDULING PROCESS – The scheduling process is done in stage wise

1.**Step 1-** After the request get processed, the request needs prefetch data, such as information about files and input data from parallel file system to burst buffer

2. **Step 2-** The scheduler uses the no of computing nodes and amount of burst buffer required for processing. Then the request undergoes in processing stage.

3**Step 3** – When the request processing get finished it output data goes to external storage.

Results -



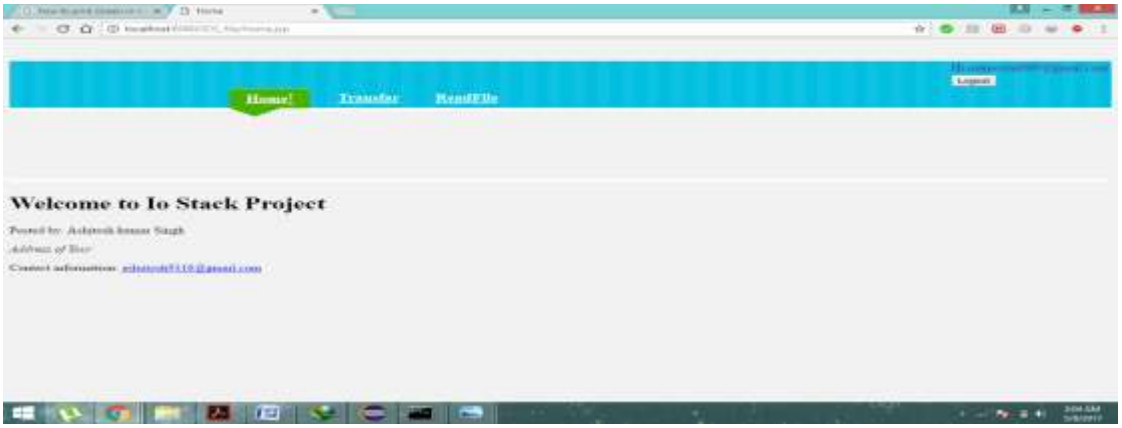


Fig 2.1 Modules implementation (1)

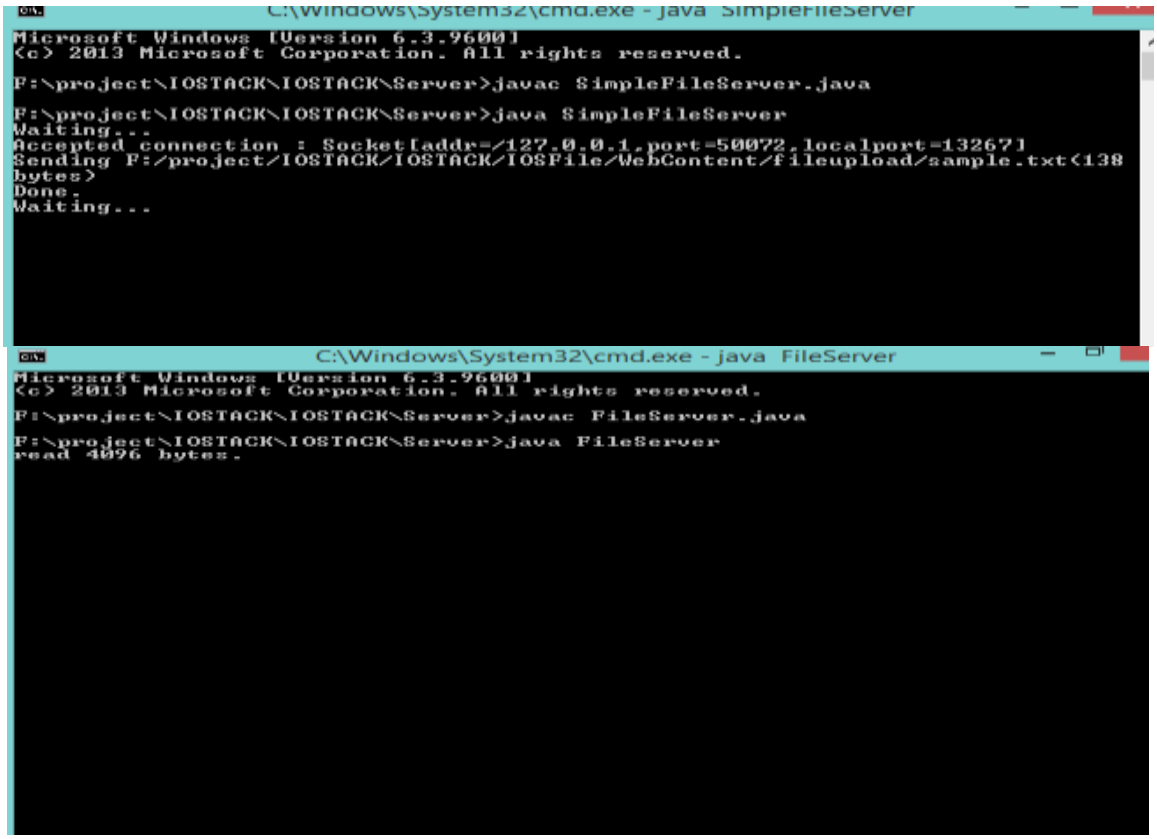


Fig 2.2 Modules implementation (2)

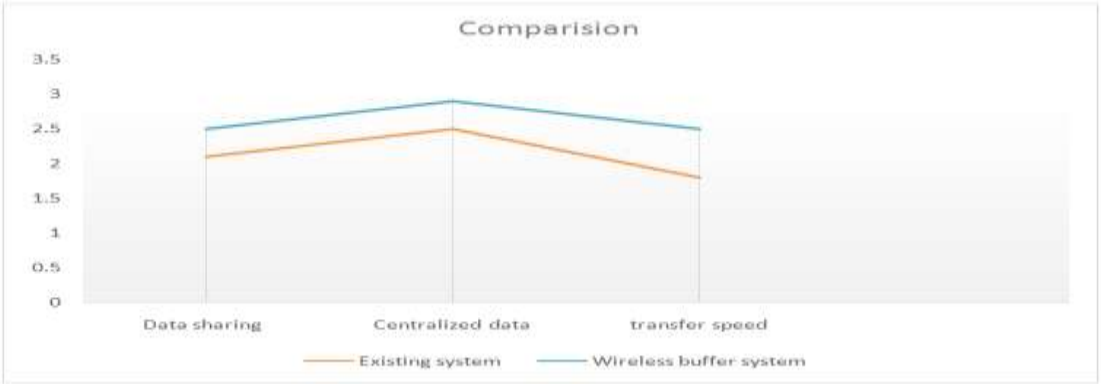


Fig 2.3 Difference between existing system and burst buffer model on the basis of data sharing, centralized data and transfer speed represented in bytes/sec (1 point= 1000 bytes/sec)

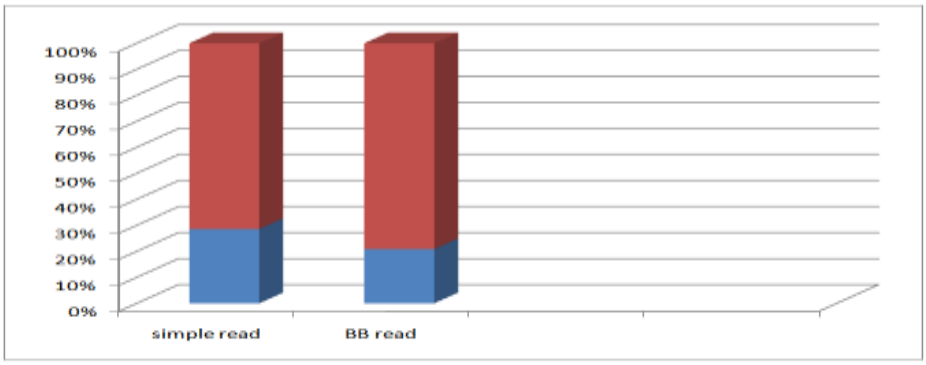


Fig. 2.4 Read Operations Chart Simple Read Operations Perform 70% and BB(Burst Buffer) Read Operations Perform 80% Represented By Red Box

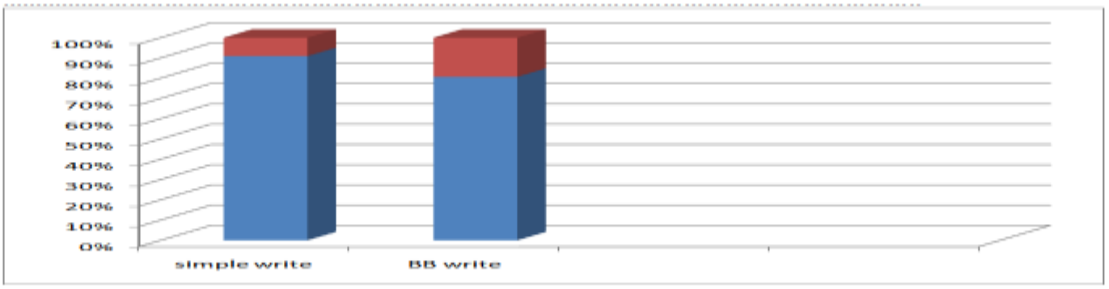


Fig. 2.5 Write Operations Chart Simple Write Operations Perform 10 % and BB Write Operations 20% Represented By Read

FORMULA

t(bb)= the time required to transfer data from computing nodes to burst buffer
t(pfs)=the time required to transfer data from burst buffer to parallel file system
a(bb)= amount of data transfer from computing nodes to burst buffer
a(pfs)=amount of data from burst buffer to parallel file system
s(bb)=the speed at which data from computing nodes to burst buffer
s(pfs)=the speed at which data from burst buffer to parallel file system

1. t(bb)= a(bb)/s(bb)
2. t(pfs)= a(pfs)/s(pfs)

CONCLUSION AND FUTURE WORK

Many previous existing solutions had concluded with their techniques, advantages, and limitations. After analyzing the existing models there is no models which is offering a fully potentials i/o performance in i/o stack. Every existing models scheme have one or more problem of i/o contention, complex resource management and dense overloaded overhead etc. This paper will help someone to determine the differences between burst buffer model and the existing models on i/o stack performance ,so that it make future improvements in those techniques so that we can get more secure, efficient and scalable scheme for enhancing better i/o performance in future

REFERENCES

[1] Ashutosh Kumar Singh, S.H.Patil, Naveenkumar Jayakumar," A Treatment for I/O Latency in I/O Stack," vol. 5 issue 2,pp 424-427,2017
[2] K. Singh, "A Study on Merits and Demerits of SAN Protocols," vol. 3, pp. 1–5, 2015.
[3] Ashutosh Kumar Singh, S.H.Patil, Naveenkumar Jayakumar," A survey of increasing i/o latency in i/o stack", IJCTA, May 2017
[4] N.k.Singh, A. K. Singh, "A Study on Virtue and Faults of Security in Cloud Computing," vol. 5, no. 1, pp. 93–97, 2017
[5] Feitelson, D. G., Corbett, P. F., Baylor, S. J., & Hsu, Y. Parallel I/O subsystems in massively parallel supercomputers. IEEE Parallel and Distributed Technology, 3(3), 33–47. http://doi.org/10.1109/M-PDT.1995.414842,1995
[6] P Wu, L., Huang, H., Su, K., Cai, S., & Zhang, X. An I/O efficient model checking algorithm for large-scale systems. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 23(5), 905–915. http://doi.org/10.1109/TVLSI.2014.2330061
[7] Zhou, Z., Yu, M., &Gligor, V. D. Dancing with giants: Wimpy kernels for on-demand I/O isolation. IEEE Security and Privacy, 13(2), 38–46. http://doi.org/10.1109/MSP.2015
[8] Computers, I. S. Disk System Architectures for High Performance Computing, 77(12), 1842–1858,1990
[9] Li, C.. High-speed optical interconnect for multimedia systems, 390–412
[10] W. Paper, "Solving I / O Bottlenecks to Enable Superior Cloud Efficiency," pp. 1–6.
[11] M. Principal, "Eliminating the I / O Blender Effect and Realizing World Class Storage Performance in Virtual Storage Environments," no. December, pp. 1–12, 2014.