Volume 5, Issue 2, February 2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Performance Impact Analysis of Application Implemented on Active Storage Framework

Naveenkumar J, Raj Makwana, Prof. S. D. Joshi, Prof. D. M. Thakore
Department of Computer Engineering, Bharati Vidyapeeth University,
College of Engineering, Pune, Maharashtra, India

Abstract— there is a tremendous increase in usage of digital devices and services provided. As the number of users increased so as the proliferation of data and need of information and knowledge. There has been increasing trend in need for storing and processing these growing data. With the development trend in technology, the storage capacity and processing power has kept well improving. But since the computing power is faster than the spinning device, there is a delay of providing data to processing component from the traditional spinning disks. This delay evolves with a Processing – I/O performance gap. This CPU-I/O gap worsens for application which are data intensive. Active storage framework are proposed to minimize this Processing – I/O performance gap. This research paper provides a test bed which analyses the performance impact on the application

Keywords— Active Storage Framework, Parallel and Distributed Systems, Storage Arrays, Storage systems, Compute Storage system, Storage Virtualization, Active Disks, RPC

I. INTRODUCTION

The end users need for information and services is exponentially increasing the data volume. As the research carried out by various researchers on data explosion claim that by 2020 the data size will be increased to forty thousand petabytes. The digital volume is doubling every year [19]. Storage vendors are manufacturing storage devices which can support this explosion of data capacity, Processor manufacturers are working building the compute power which are capable of processing these huge amount data as well network medium are very well developed and deployed as per the need of current infrastructure. When every component of the cluster are well advanced and running there is still performance issues faced at application end.

While scanning whole network components right from server to hard disks, some end to end application gets severe blow in their latency. The delay in latency is caused by each components of the network. The latency delay might be due to network bandwidth and traffic, spinning disks, processing time of data etc. Even though considering the network component used as infiniband and there is plenty of bandwidth available then also the application response time is hindered by the mechanical device down the I/O path.

In case of Data intensive or I/O intensive application data needs to be read or written in huge quantity making data processing the focal point. There are many architectures proposed for executing such applications in efficient manner. One of the architecture is active storage. In active storage framework, the application is moved near data and executed. The core idea is instead of moving data from spinning disks to application server, application is moved to the system where the data resides. The active storage reduces the heavy utilization of network and also reduces the wait cycles of the processors at both sides. This framework tightly binds the compute and storage together in single box. A set of data intensive tasks can be offloaded to this box for executing near data. This paper will discuss about the performance impact on application and as well as the storage systems by deploying the active storage framework. Section II will discuss regarding the existing studies carried out on active storage framework, the next section proposes active storage framework, continued with Test bed architecture and result analysis.

II. LITERATURE SURVEY

This research regarding active storage was focused on one of the problem in executing data intensive application. The data growth is the main challenger for application server. These data growth may lead to more storage and data processing, for processing, transfer of data will consume more bandwidth and increases the traffic. These factors directly impact the performance of software as well hardware. There is significant impact on the Application performance. In order to reduce the data movement among the data storage and application server researchers have proposed active storage framework in which the application or part of the application will be moved close enough to data where it is stored. The motive behind the active storage is improve end to end application performance and utilize the underutilized components of storage systems. [1] proposes Applications are becoming more data intensive. Small parts of huge multidimensional arrays are concurrently accessed by a large number of clients, both for reading and writing. The scalability of data management becomes a critical issue, as it affects the performance of the whole application. There is an increasing need to specialize the I/O stack to match the requirements of the application. SciDB does not address

chunking layout management which may be a problem in exascale. There is a gap between the multidimensional arrays (application model) and Parallel files systems and relational tables. [1][2][3] Proposes To reduce parallel access contentions the approach is chunking. To improve performance of strided access patterns propose to split the array into sub domains that are equally sized in each dimension. Using this approach, the neighbors of cells have a higher chance of residing in the same chunk irrespective of the query type, which greatly limits the number of chunks that need to be accessed

III. PROPOSED ACTIVE STORAGE FRAMEWORK

The active storage framework proposed here is implemented within the storage silos or storage arrays. A storage silos consists of multiple components. The components within the storage stack are –

Host end ports - These ports are the ports which gets connected to Host Bus Adapter's from various other hosts. These ports accept the requests from the clients or application server for the data. These ports are responsible for serving the required data back to the host.

Host End Processors – These are set of compute power attached to front end ports. The processor can accept and acknowledge the I/O request one at a time. If four processor are currently active then each processor can accept and acknowledge one request and service it per unit time. These processor are responsible for serving the request from host

Cache – This is the fastest memory which will satisfy many requests from the host. The Host end processor will service the request immediately if it gets the data from the cache. Cache is global memory which can be partitioned and allocated to individual processors for their use or it can also be used as one single large memory. This configuration varies according to various manufacture.

Disk End Processor – These processors are an interface between the host and the disks. These processors are the closest to the disks. These compute power try to fetch the data from disks and forward it to the host.

Disk End Ports – These are the interface between the Disks and the disk end processors.

Disks – These are the storage units which has been configured in various RAID as per the requirement of the application. These disks store all the data required for the processing as well as write data from the host.

The proposed active storage is erected or deployed within the storage silos. In the proposed framework, the disk end processors execute the offloaded application within the storage silos closer to data. The figure 1 depicts the active storage framework.

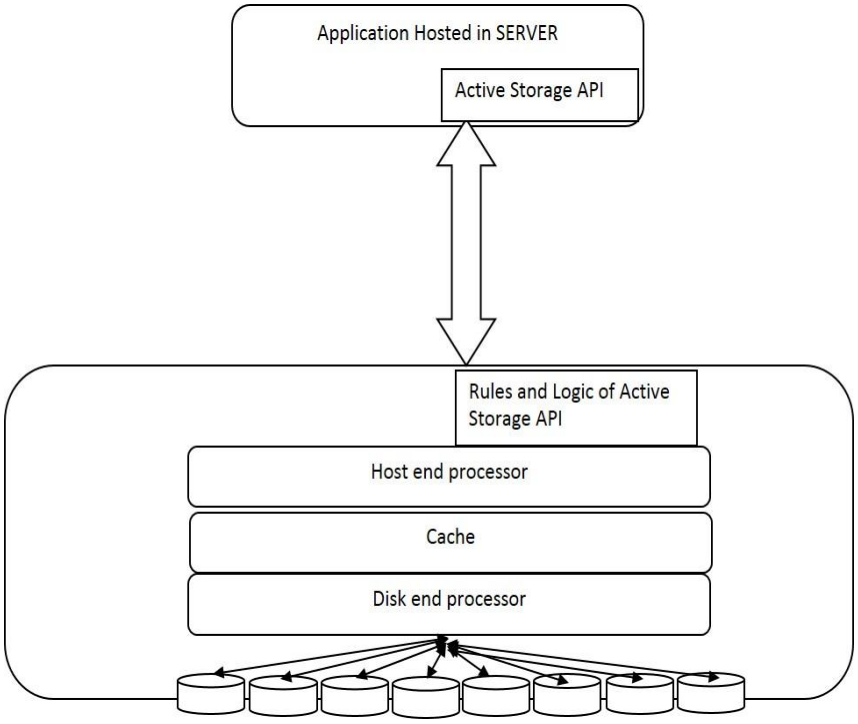

Figure 1 Active storage framework

In figure 1 the storage array components will incubate the file storage API and the set of rules and logics in order to accept and co-ordinate the offloaded application. The application or I/O intensive part of the application is downloaded or offloaded to the storage array from application hosting system generally application server. At the application server side, there is an file system API along with listener and decomposer which works on identifying the data intensive part of the application and decomposing it, storing the states of the application part and initializing the offloading part. The offloading happens through the API. The function is offloaded to the storage system and executed there close to data. This offloading process reduces the number of hops the whole data needs to be transferred between the application server and storage disks. Thus by reducing the performance impact on end user application. In order to check whether is really there a variation in performance of the application while implementing or executing the application in active storage framework. The following section discusses upon the test bed environment and results.

IV. TEST BED ARCHITECTURE AND ENVIRONMENT

The purpose of this research paper is to find the performance variation and component utilization of storage silos and end to end application. There are two test beds proposed here.

The first test bed represents the traditional way of execution. The server node hosts the application and the request for the data is sent to the storage nodes or a single storage silos. Requested data is fetched from tradition drives or cache in storage node and moved up to the server end disk or cache for processing. For test purpose the tool Iometer [20] is used. IOmeter is a disk storage benchmarking tool which has two component the GUI part and the dynamo part. The Dynamo is the heart of this tool. It is responsible for creating and writing the file and data into the file respectively. It also reads the data from the created file.

The second test bed is setup to simulate the active storage framework. In this test bed, the same tool is used and configured according to the active storage framework. The Dynamo part of the IOmeter is separated from the GUI part and is migrated to the storage node. Using the storage virtualization a Virtual machine is created running Linux. The Dynamo component is executed inside the VM at storage side. The GUI part is run at the application host side running windows. Since the dynamo performs more of a data intensive or I/O intensive tasks that particular component is brought at the storage side and read write tasks are executed very close to the spinning disks. The values are just forwarded to GUI.

The configuration of the systems at storage silos end are Xeon processor with four cores, 4GB RAM, 500GB 7200 HDD configured to two LUNs, with bare metal hypervisor running one Linux virtual machine. The configuration at host side is i7 processor, 2GB Ram with 160GB HDD.

The workload used for these tests were a combination of various access pattern as 100% sequential to 0% sequential for IO size ranging from 32KB to 64KB. The test bed architecture for both test environment is shown in figure 2 and figure 3.

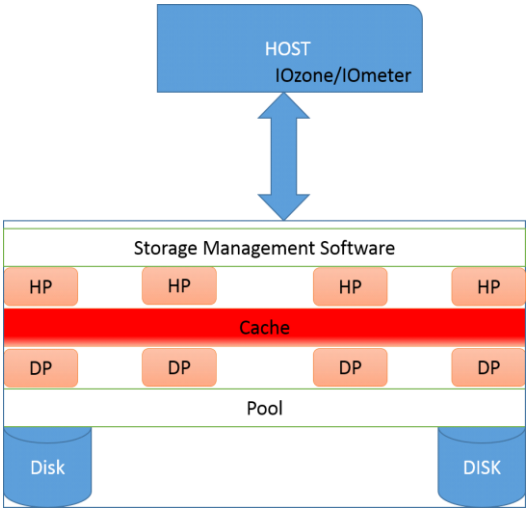

Figure 2 Traditional

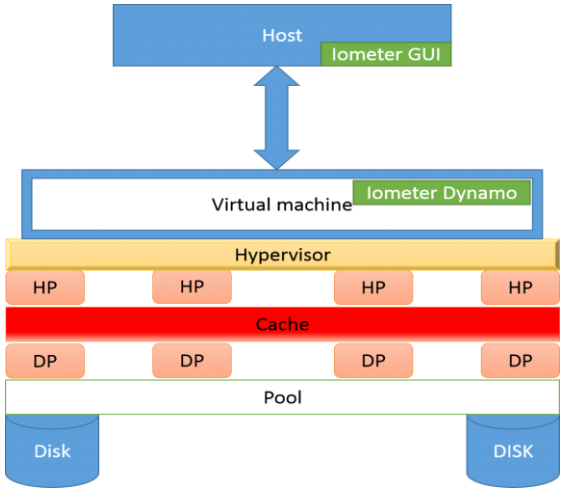

Figure 3 Active storage testbed

The metrics considered for evaluating these two testbeds are IOPS, Throughput, Response time, processor utilization. In general IOPS is the metrics which considered for benchmarking the storage systems. Throughout is the amount of data carried to and fro via network per unit time between server and storage. The latency decides the response time of the application. The processor utilization is measured for the storage systems. Since having the processors near to disks in storage system, it is seen in the tests in traditional system testbed that they are not utilized. So in order to utilize the unused processing power of storage systems, the active storage framework will help in offloading the application or tasks to the storage system and execute there.

Considering the graph in figure 4. the IOPS has been increased in the active storage framework than in traditional test bed. Since the compute as well as the I/O requests are getting executed or serviced inside the storage array.

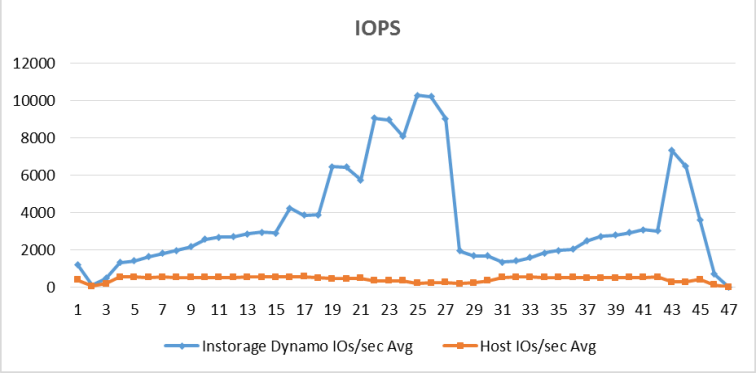

Figure 4 IOPS of storage system

Reviewing the graph of throughput in figure 5, it is expected that the bandwidth consumed should slope down heavily but in the graph shown it is reduced minutely because the luns are being accessed by other application as well. These access are also transferring data via network hence the bandwidth consumption is reduced with minor fraction.

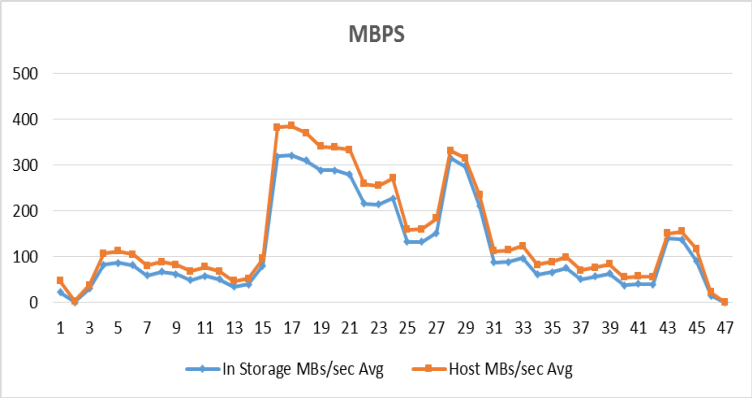

Figure 5 Throughput in active storage

Looking at the processor utilization figure 6 of both test bed, it is seen that cpu utilization in active storage test environment is increased because the cpu needs to process the I/O requests as well as the Compute/Instruction processing. The CPU accepts and acknowledges the I/O request one at a time and the CPU time is shared between servicing I/O request and processing the offloaded code.

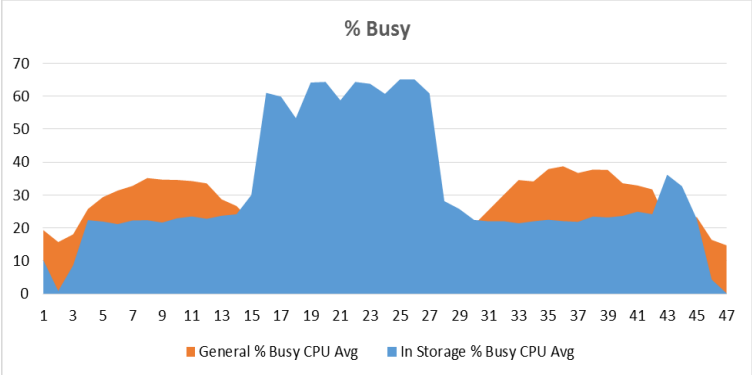

Figure 6 Processor Utilization

Analyzing the Latency of application figure 7, the protocol conversion has been reduced there by reducing the hops between the application server and storage systems, hence the response time of the application is improved by reducing it. In case of active storage testbed the response time is reduced than the traditional system testbed.

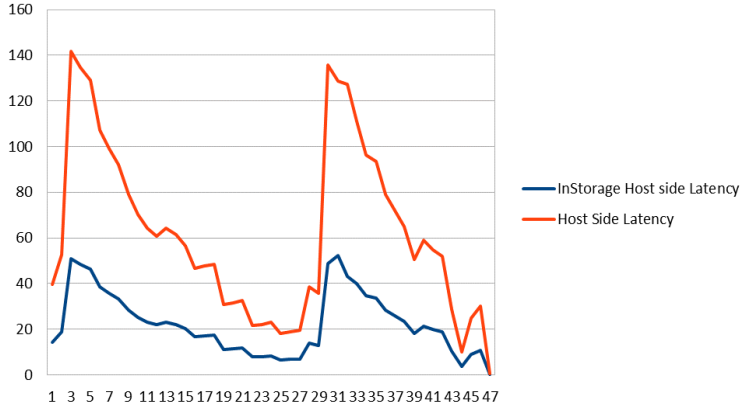

Figure 7 Latency

V. CONCLUSIONS

This paper revolves around the active storage framework performance impact on end to end infrastructure and application. Active storage framework is used to move the code near data storage for reducing the network traffic. In this paper it is analyzed and seen that in active storage performance the application response time is improved by reducing the bandwidth consumptions and increasing the CPU utilization at storage array. The underutilized components of storage array are also gets used. It is evaluated on this testbeds proposed in this paper that active storage framework will really impact the application performance positively and will be beneficial to be deployed for data intensive or I/O intensive application in an HPC environment.

REFERENCES

[1] Huang, Cheng, Minghua Chen, and Jin Li. "Pyramid codes: Flexible schemes to trade space for access efficiency in reliable data storage systems." Network Computing and Applications, 2007. NCA 2007. Sixth IEEE International Symposium on. IEEE, 2007.

[2] Tran, VietTrung, Bogdan Nicolae, and Gabriel Antoniu. "Towards scalable arrayoriented active storage: the pyramid approach." ACM SIGOPS Operating Systems Review 46.1 (2012): 1925.

[3] Tran, VietTrung, et al. "Pyramid: A largescale arrayoriented active storage system." LADIS 2011: The 5th Workshop on Large Scale Distributed Systems and Middleware. 2011.

[4] Acharya and J. Saltz. Active Disks: Programming Model, Algorithms and Evaluation. In Proceedings of the eighth international conference on Architectural support for programming languages and operating systems, 1998.

[5] S. Chiu, W.-k. Liao, and A. Choudhary. Design and Evaluation of Distributed Smart Disk Architecture for I/O-Intensive Workloads. In Proceedings of International Conference on Computational Science, 2003.

[6] G. Chockler and D. Malkhi. Active Disk Paxos with infinitely many processes. In Proceedings of the 21st CM Symposium on Principles of Distributed Computing, 2002.

[7] J. Felix, K. Fox, K. Regimbal, and J. Nieplocha. Active Storage Processing in a Parallel File System. In 6th LCI International Conference on Linux Clusters: The HPC Revolution, Chapel Hill, North Carolina, 2005.

[8] M. Franklin, R. Chamberlain, M. Henrichs, B. Shands, and J. White. An Architecture for Fast Processing of Large Unstructured Data Sets. In Proceedings of the IEEE International Conference on Computer Design, 2004.

[9] Anastasiadis, S. V., Wickremesinghe, R. G., & Chase, J. S. (2005). Lerna: An active storage framework for flexible data access and management. *Proceedings of the IEEE International Symposium on High Performance Distributed Computing*, 176–187. doi:10.1109/HPDC.2005.1520955.

[10] Arpaci-Dusseau, A. C., Arpaci-Dusseau, R. H., Bairavasundaram, L. N., Denehy, T. E., Popovici, F. I., Prabhakaran, V., & Sivathanu, M. (2006). Semantically-smart disk systems. *ACM SIGMETRICS Performance Evaluation Review*, *33*, 29. doi:10.1145/1138085.1138093

[11] Bechini, A., & Vetrano, A. (2013). Management and storage of in situ oceanographic data: An ECM-based approach. *Information Systems*, *38*(3), 351–368. doi:10.1016/j.is.2012.10.004

[12] Black, D. L. (2008). SNIA Legal Notice.

[13] Braam, P. J. (n.d.). Lustre : The intergalactic file system. *Computer*.

[14] Chandrasekaran, K. (n.d.). Analysis of Different Parallel Programming Models.

[15] Chandy, J. a. (2006). Active Storage Networks. *Development*.

[16] Chen, C., & Chen, Y. (2012). Dynamic active storage for high performance I/O. *Proceedings of the International Conference on Parallel Processing*, 379–388. doi:10.1109/ICPP.2012.22

[17] Cherkasova, L., & Gardner, R. (2005). Measuring CPU overhead for I/O processing in the Xen virtual machine monitor. *Proceedings of the USENIX Annual …*, 387–390. Retrieved from http:// static .usenix .org / events / usenix05 / tech / genera l/ full_papers /short_papers /cherkasova /cherkasova.pdf.

[18] hiu, S. C., Liao, W. K., & Choudhary, A. N. (2006). Distributed smart disks for I/O-intensive workloads on switched interconnects. *Future Generation Computer Systems*, *22*(5), 643–656. doi:10.1016/j.future.2005.09.007.

[19] http://www.emc.com/collateral/analyst-reports/idc-the-digital-universe-in-2020.pdf

[20] http://greg.porter.name/wiki/HowTo:iometer

