ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 11, November2015

Evaluation of Active Storage System Realized through MobilityRPC

Naveenkumar J [+], Prof.Dr.S.D.Joshi[*]

[+]Research Scholar, Dept. of Computer Engineering, BVDUCOEP, Pune, Maharashtra, India

[*]Professor, Dept. of Computer Engineering, BVDUCOEP, Pune, Maharashtra, India

ABSTRACT: Recent meliorations in storage technology and emerging high performance computing and interconnects have facilitated to construct systems scaling new high processing power by connecting thousands of compute and storage nodes. However, large-scale simulations and computations using this kind of environment, postulating the ginormous and raising masses of data remain, still a challenging problem. It has been noted that, the cost of bandwidth for moving data between the processing nodes and the storage node has not been significantly improved at the same time as the disk capacity. Because of huge quantity of Data intensive applications & I/O Applications, the storage and transfer of data remains a serious bottleneck. This paper evaluates the offset cost of hosting the applications would be offset, by the effective (aggregate) overhead involved in driving the end-to-end data transfer between the array and the applications

KEYWORDS: Storage Array, Active Storage, Application Offloading, Java, Mobility RPC

I. INTRODUCTION

The IT sector has been reinventing its business platform now and then. These reinvention and innovation changes are happening at much increased pace. The transformation of IT landscape has happened from first Platform to the third Platform.[1] The first platform was purely based on the mainframes and terminals, which were referred to as dawn of the computing era. The Second Platform was built on the foundation of the emergence of personal computers, servers, database systems and client-server models. With the advent of cloud, mobility, big data analytics and social networking the concept of third platform was emerged. The current trend in information technology embraces the terms like 'Big Data', 'Mobile Engineering', 'Cloud Computing' and Social Networking Platform. All these terms refer to technologies, which have been used by a mass of end users.[2]–[4]

To keep up with the innovation pace and sustain in the business, IT organizations are undergoing huge transformations. Applications have been developed which are capable of running on mobile platforms and accessing the remote compute and storage. As mentioned earlier, the scope of users and deployment model of application, the business model enables the need to migrate to the third platform. As the users scale-up, so the infrastructure and the data grow exponentially. There is an outbreak of data from multiple spectrums of devices that are required to be analysed and processed for extracting useful information.

In order to meet performance demands, it is essential to optimize the processing and I/O subsystems. One promising approach to optimize performance is to use "active storage". The advent of low latency, high bandwidth network architectures and embedded processing having enabled hard disks to function as active storage devices which is discussed in this paper.

II. RELATED WORK

Existing Storage system is an example of ad-hoc, proprietary based approach to realize a form of active storage, whereas, approach to embed a guest operating system is the other spectrum of Active Storage enabler. While, what is desirable is something like Hadoop ecosystem, which fits in as midway between these two, the Hadoop based ecosystem allows offloading of compute/data storage IO intensive IO tasks among the data servers in the Hadoop cluster.[5][6][7]

Proper characterization of these models as enablers of active storage system is not available from the literature review. These approaches are developed and deployed on ad-hoc basis as and when needed. [8]

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 11, November2015

Some storage industry commodities are proprietary intelligent storage system, which comprises the offloading of tasks near to the data. However, these offloading are not full-fledged and are ad-hoc in manner. Those storage components can be partitioned into three compartments like Database servers, Network fabrics and Storage servers also called storage cells. The more storage cells more will be the capacity and high bandwidth. The offloading process happens between the database servers and storage cell through iDB (intelligent database protocol). The iDB works as function conveyance to transparently represent the database operations to ExaData operations. It is also used for transferring data between database node and storage cell. iDB is implemented using LIBCELL and in turn, ExaData binaries are linked up with the LIBCELL to facilitate cell communication. The heart of the ExaData storage cell for providing the unique features and services is CELLSRV. The CELLSRV provides majority of services. Cell Offload significantly reduces the IO transfer over storage network, memory and CPU utilization database tier nodes. Since direct path read mechanism is implemented the buffer cache will not be saturated for large IO requests. [9][10][11][12], [13]

DDN also emerged with its own storage fusion architecture (SFA) which was designed for multi-processor and multi core systems. The processors inside the storage were divided into application processor and RAID processors. The Application processors were dedicated to execute tightly coupled applications within the storage subsystem. With the help of virtualization tool the applications were brought into storage subsystem and the application processor were used to execute them. Operating system with the SFA acts as hypervisor to control processors/cores, memory, IO and virtual disk allocations. This also takes care of applications running in embedded space, hence cannot hinder block operations memory space and that the applications only utilize the processing and IO resources that have been coupled with. [14][5], [15], [16]

III. PROPOSED REALIZED ACTIVE STORAGE TESTBED

A. *Testbed Considerations:*

The evaluation is conceded over two altered set of setups with two different Testbeds and evaluating the performance of two frameworks with same synthetic workloads representing real time application profile. In other words, embedding the frameworks on both the Testbed and evaluating it by executing applications on it. This evaluation will reveal the behaviour of system to the applied workload.

The two sets of Testbed are, firstly without active storage Testbed i.e. the server node is used in its native way. Secondly, with active storage Testbed i.e. the server offloads the compute component of the application or the application itself nearer to the storage node, in other words the compute framework is embedded in active storage.

B. *Description of the Testbed:*

The ‘mobilityRPC is a Java Implemented Library. The mobilityRPC is used to develop the framework deployed here, which facilitates in offloading the code dynamically between the nodes. [https://code.google.com/p/mobility-rpc/wiki/WhatIsCodeMobility]. In this evaluation, this framework is modified slightly with respect to interfaces and developed new listeners in order to embed it inside the proposed model.

There are again two set of deployments, application running with mobility RPC and the same applications running without RPC using generic socket and ports.

The below table shows the hardware and software used for evaluating both the Testbeds. The hardware section in table show two sets of resources being configured, the first column in which Xeon processor is used is the storage node and the second column with i3 processor is the application host node. These two nodes are directly connected via network switch and configured to share the drives in Testbed 1. Testbed 1 is a native storage node – application host node layout pattern in which the data is stored on the storage node and the application is executed on application host node. For the data required by the application, it needs to fetch the data from storage node, transfer to application host node buffer and then perform the execution on it.[17], [18]

In case of Testbed 2, the hardware is invariant compared to Testbed 1. The software here used is ESXI 5.5 hypervisor, which is bare metal hypervisor, and it creates virtual resources of all hardware components of storage node. The ESXI 5.5 is chosen since, it is the mostly used and deployed enterprise level virtualization technology in current time as well it supports real-time performance monitoring of the resources. In this deployment, the hardware is virtualized and multiple virtual machines can be created on which the storage services can be run on separate virtual

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 11, November2015

machines and the offloaded applications can be run of other virtual machines, which are attached with virtual disks. These virtual disks store the data required by the application.[8], [19] These underlying disks can be configured as (BoD) bunch of disks or RAID. The disks when combined together from RAID, the logical entity are created which is referred to as LUNs. These LUNs can be used as Data stores for storing virtual machine data and virtual disks can be created from these and provisioned to application for use. The disks are configured with RAID 5 in both the deployments. Here after the two testbeds will be regarded as Architecture model 1 Testbed and Architecture Model 2 Testbed that are shown in below Fig.1 and Fig. 2.

Table 1: Software and Hardware of testbed

SYSTEM WITH TESTBED 2	
HARDWARE	
1 Intel Xeon Processor 2.4 GHz 1 6 GB RAM 1 Gigabit Ethernet network card 3 300GB HDD SAS 2.5”	1 i3 Processor 2.4 GHz 1 4 GB RAM 1 Gigabit Ethernet network card 1 500GB HDD
SOFTWARE	
ESXI 5.5 3 Virtual Machines (Windows / Ubuntu) MobilityRPC Framework IO Meter Hadoop Net Beans IDE 8.0.2	

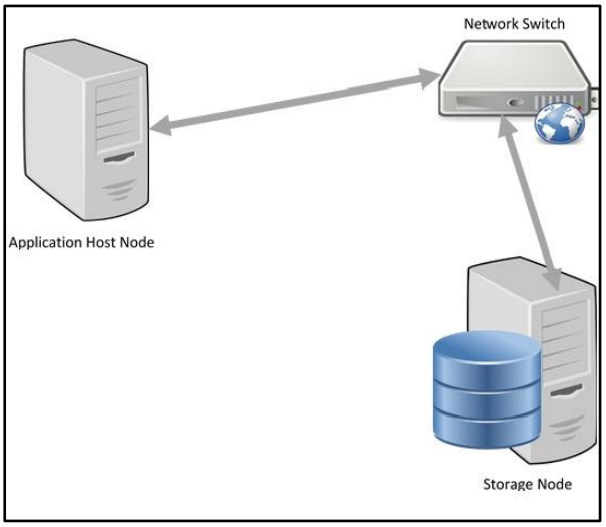

Fig.1: Architecture Model 1 testbed

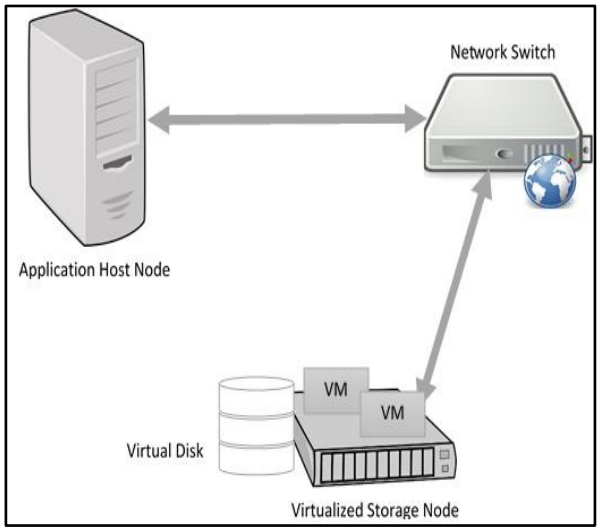

Fig.2: Architecture Model 2 testbed

C. *Architecture Model 1a Testbed*

The socket programming has been used to deploy the traditional way of interaction between the application host node, storage array node and application execution pattern. The application execution style implemented here requires the data to be moved from the storage node to the application host node. This uses network bandwidth and increases the time of application execution completion since, the waiting time increases for the application to receive the data required for processing.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 11, November2015

The purpose of the application developed and deployed here is to identify and display the line from the paragraphs in each file of the dataset, which has the maximum number of words in it. This program is not completely I/O bounded but it needs a large amount of files for processing. This application purely exhibits the read operations and the data are scattered across the two disks and one disks making the read operation to be random as well as sequential. The application is developed using the Java Program.

Reuters dataset has been considered here which constitutes approximately 15000 text files. The goal of developed and implemented application is to read these individual files from dataset, scan it, identify the line in each paragraph with maximum number of words and display it.

The application was developed with the sense of making the magnetic heads to move back and forth at the same time little bit processing of counting and comparing is given to processors.

D. *Architecture Model 1b Testbed:*

The mobility remote procedure call libraries are embedded into the virtual machine that is hosted on storage node. The application is hosted at other node, which is referred to as application host node. These two nodes are connected via Ethernet. The application starts running on the application host node, the application itself uses the libraries and API of the mobility remote procedure call to identify and offload the data intensive component of the application.

There are listeners, which are used for identifying the data intensive part of the application, which helps in segregating, and offloading that particular component to data storage node using the Application-programming interface embedded on both nodes. The listener needs to be developed for various applications. The behavior of individual applications varies with its design and development pattern. Since the behaviors are different and the components designed will interact in a specific pattern to achieve the purpose of the application, it becomes necessary to understand the programming model of that application.

The core idea of the programming model for this kind of implementation where it is needed to migrate the components or application closer to data it becomes necessary to design the application in a way that the application should fundamentally migrate the I/O intensive application or component closer to data keeping the processor bound component as it is.

There are listeners that can actually identify the part of functions and statement in the code that can be offloaded to the storage node. This is not the actual partitioning of the application but implementation is closer enough to show the partitioning of the application.

This implementation shows the application-programming interface of the mobility remote procedure call to accept the parameter from the listeners and then offload that object.

This research work includes the development, implementation and testing of listeners related to text file and used along with it also proposed skeleton of listeners for video and other files which are elaborated below.

The concept of partitioning of application depicts how the data intensive part of an application is identified and segregated. By data intensive, it is meant to be the part that interacts with the data. The part of the application that interacts with the data is only migrated. The part that fetches the data from the remote location and brings back the result to the local station. The application partitioning logic for all the above-mentioned data items is generalized. Since the motive of the research to identify the cost of storage for embedding the application and observing the enhancement in the application performance, the application partitioning has not been researched out in depth. The mobilityRPC supports the dynamic offloading of the application but does not has inbuilt application partitioning logic hence this implementation uses the Listener logic using the buffer and drivers to identify the data intensive part.

The partitioning listeners not only identify a connection to the remote data source but also wrap the connection code into objects. This object is migrated to the remote location to fetch the requested data. The impact of application partitioning can only be measured during object creation. It depends upon how many methods from the application-programming interface are being utilized for the connection and the how many external connections are to be made for partitioning.

IV. EVALUATION RESULTS

Processor utilization - In the curve of testbed 1b shown in Fig. 3, at initial stage there is a steep increase that rises to the peak value of approx. 77% of the utilization and fluctuates between 77% - 75% but stays constant within the same range. Here the overall aggregate CPU utilization is measured around 38%, the aggregate is considering all the

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 11, November2015

cores of the system. On the other hand, when considering the curve for testbed 1a, fluctuations in the utilization curve is seen which reflects the pattern of the term used in management studies head and shoulders. The frequent fluctuation is because of accepting the I/O request, acknowledging the I/O and connecting with the application node. Since the testbed 1a curve is the result of socket programming implementation, the ports are the component through which the data are sent back to the application host node. This task of acknowledging the I/O request, processing request and sending the required data consumes the cycles of the processor at various intensity, which is seen in the curve.

The curve in the testbed 1b shows decent improvement in the processor utilization of storage node compared to the testbed 1a curve. Thus, the graphs and tables shown below prove that active storage node increases utilization of underutilized resources and varies the utilization of processors combined with ports. This is the cost of embedding the application in storage node.

Throughput Analysis - Since the Testbed 1b embeds the computation closer to data, the I/O request as well as the offloaded application component needs to be serviced by the processor at storage node. Once the processing is completed, the result needs to be transferred back to the application host end for further processing or to send back to client. This is measured by throughput parameter in KBPS. Observing at testbed 1a throughput curve from Fig. 4, since the dataset contains various sized text files, these text files were streamed out of the storage node in sets and the set size varied internally. This variation in set size varied the curve. There is a steep increase from the regular throughput measuring at 0 and 3KBPS to directly to 3171KBPS and peaking to 11110 KBPS. This drastic increase in the throughput is due to the socket connection and huge amount of text files of the dataset being initiated to transfer. As the dataset files being transferred the remaining dataset is reduced in quantity, but the reason behind the dropping of the throughput is the size of files being transferred. As in the dataset, size of the files varies hence the quantity decreases the total file sizes to be transferred also decreases thus by reducing the throughput. The sloping down of the throughput is because of the files size is reduced after some time and reaches 0Kbps after transferring all files.

In case of testbed 1b, since the results are only needed to send back to the application host node, the throughput is tremendously dropped down. The result set size is very less compared to the dataset size. Thus, it is comprehended from the testbed 1b throughput curve that the drastic reduce in the throughput decrease the network traffic.

Elapsed & Latency - Considering the third parameter of runtime or elapsed time and latency, the end-to-end performance is measured through the response time of the application. The response time of the application is measured at the application host end when all the data is received from the storage node. The latency for testbed 1a is measured at the application host side through the net beans runtime when the execution stops at end of file. The elapsed time is measured at the storage node when the storage processors complete transferring the files. On the same note, the testbed 1b parameter is also measured.

The latency is improved in testbed 1b implementation model compared to other observed from bar chart in Fig.5. Thus by proving that the time required transferring data would be compensated by increasing the processor utilization of storage there by affecting the application response time. If the throughout is more the response time of application is becoming less as observed from the below bar chart shown in Fig.6.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 11, November2015

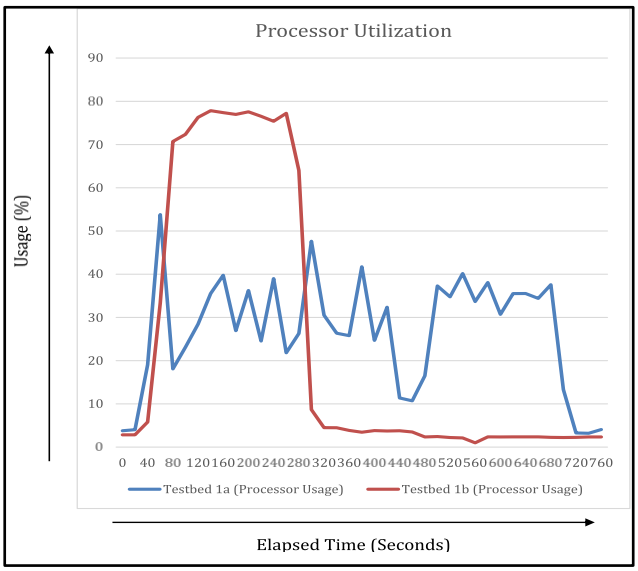

Figure 3 Processor utilization Comparison

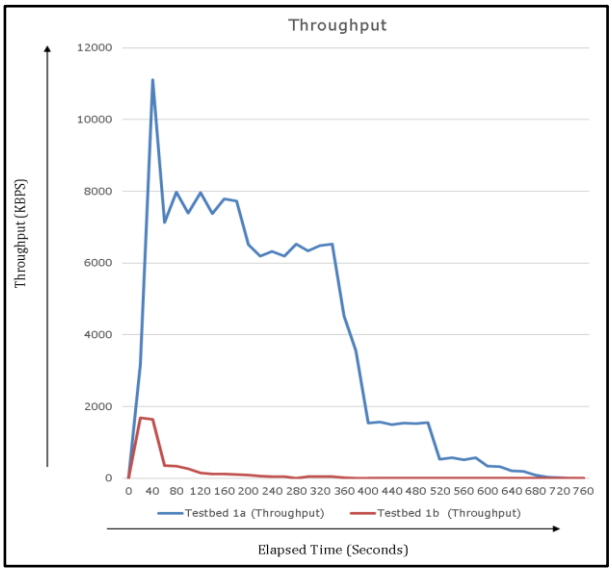

Figure 4 Throughput Analysis

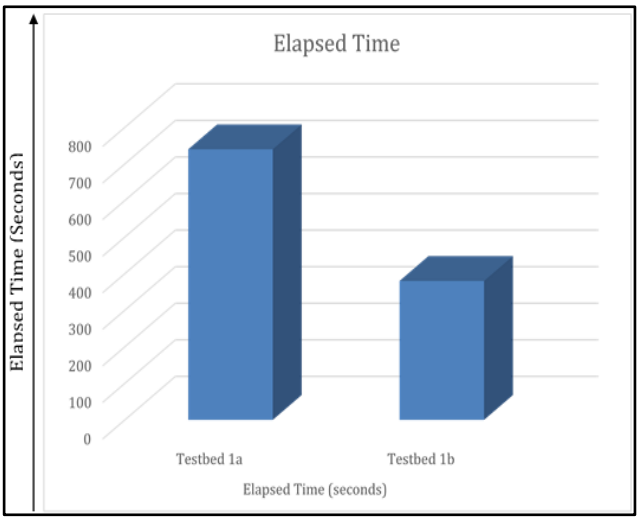

Figure 5 Elapsed Time Comparison

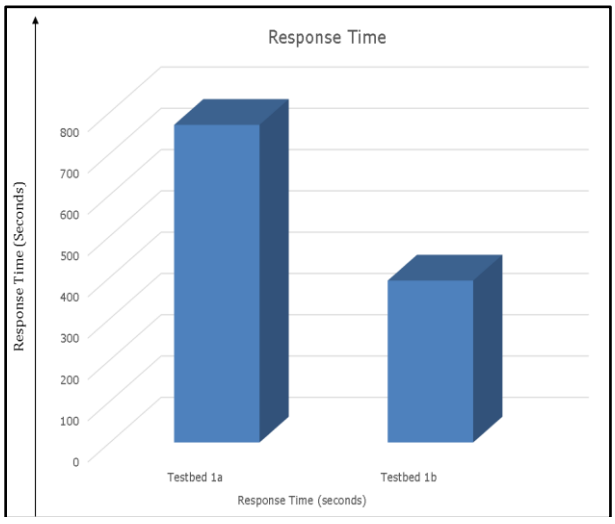

Figure 6 Response Time

V. CONCLUSION AND FUTURE WORK

Comparing the results of the two testbed execution for mobility RPC, it is very much observable that the second testbed i.e. realizing proposed model have increased utilization of the processors which is actually the cost for embedding the application inside the storage. In this evaluation, the testbed 1a have been deployed using traditional socket programming implementing same application with same data similarly as the mobility RPC implemented in testbed 1b. It is very clear from the above evaluation results that there is some cost of embedding the application within the storage array which also compensates the cost of data transfer rate which is verified through the latency and elapsed time of application.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 3, Issue 11, November2015

REFERENCES

[1] A. Mcafee, "The third platform," 2013.
[2] T. Myth and P. Revolution, "The Myth of the 3rd Platform Revolution," no. July, pp. 1–7, 2015.
[3] A. Konary and R. P. Mahowald, "The Coming of the 3rd Platform and What This Means for Software Business Models," no. May, 2013.
[4] F. Gens, "The 3rd Platform: Enabling Digital Transformation," *Idc*, no. November, pp. 1–13, 2013.
[5] J. Coomer, "Big Data Evolution," 2012.
[6] N. J, "Evaluation Parameters of Infrastructure Resources Required for Integrating Parallel Computing Algorithm and Distributed File System Evaluation Parameters of Infrastructure Resources Required for Integrating Parallel Computing Algorithm and Distributed Fil," *Int. J. Sci. Technol. Eng.*, vol. 11, no. 12, pp. 251–254, 2015.
[7] S. C. Chiu, W. K. Liao, and A. N. Choudhary, "Distributed smart disks for I/O-intensive workloads on switched interconnects," *Futur. Gener. Comput. Syst.*, vol. 22, no. 5, pp. 643–656, Apr. 2006.
[8] N. Jayakumar, F. Zaeimfar, and S. D. Joshi, "Workload Characteristics Impacts on file System Benchmarking International Journal of Advanced Research in Workload Characteristics Impacts on file System Benchmarking," *Int. J. Adv. Res. Comput. Sci. Softw. engiinerring*, vol. 4, no. 2, pp. 39–44, 2015.
[9] D. T. Jayakumar, Naveenkumar, Raj, SDjoshi, "International Journal of Advanced Research in Computer Science and Software Engineering," *Int. J.*, vol. 2, no. 9, pp. 62 – 70, 2012.
[10] W. Paper, "DEPLOYING ORACLE DATABASE 11 g ON EMC SYMMETRIX VMAXe," no. June 2011.
[11] A. Oracle and W. Paper, "A Technical Overview of the Sun Oracle Database Machine and Exadata Storage Server," *Components*, no. June, 2010.
[12] C. L. Philip Chen and C. Y. Zhang, "Data-intensive applications, challenges, techniques and technologies: A survey on Big Data," *Inf. Sci. (Ny).*, vol. 275, pp. 314–347, Aug. 2014.
[13] T. Leyden and M. Wos, "A Beginner ' s Guide To Next Generation Object Storage," 2013.
[14] S. Microsystems, "LUSTRE ™ FILE SYSTEM," no. December, 2007.
[15] P. Pendle, "IMS on z / OS Using EMC Symmetrix Storage Systems."
[16] "OFFLOADING COMPRESSION AND DECOMPRESSION LOGIC CLOSER TO VIDEO FILES USING REMOTE PROCEDURE CALL," no. August, 2015.
[17] "A GENERIC PERFORMANCE EVALUATION," no. FEBRUARY 2014, 2015.
[18] J. Naveenkumar, R. Makwana, P. S. D. Joshi, and P. D. M. Thakore, "Performance Impact Analysis of Application Implemented on Active Storage Framework International Journal of Advanced Research in Performance Impact Analysis of Application Implemented on Active Storage Framework," *Int. J. Adv. Res. Comput. Sci. Softw. engiinerring*, vol. 5, no. 2, pp. 1–6, 2015.
[19] https://github.com/npgall/mobility-rpc
[20] B. Hedlund, "Understanding Hadoop Clusters and the Network," Stud. Data Cent. Networking, …, 2010.
[21] G. Porter, "Decoupling storage and computation in Hadoop with SuperDataNodes," ACM SIGOPS Oper. Syst. Rev., vol. 44, no. 2, p. 41, 2010.
[22] J. Issa and S. Figueira, "Hadoop and memcached: Performance and power characterization and analysis," J. Cloud Comput. Adv. Syst. Appl., vol. 1, no. 1, p. 10, 2012.
[23] H. Performance and D. Division, "Hadoop MapReduce over Lustre *," 2013.
[24] D. W. Zhang, F. Q. Sun, X. Cheng, and C. Liu, "Research on Hadoop-based enterprise file cloud storage system," in Proceedings of 2011 3rd International Conference on Awareness Science and Technology, iCAST 2011, 2011, pp. 434–437.
[25] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The Hadoop distributed file system," in 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies, MSST2010, 2010.
[26] H. S. Brief, "What is Hadoop ? Why virtualize Hadoop nodes ? How will Hadoop Nutanix & Hadoop = Enterprise Grade Big Data."
[27] J. Buell, "A Benchmarking Case Study of Virtualized Hadoop Performance on VMware vSphere 5," Tech. white Pap. VMware, Inc, 2011.
[28] Nutanix, "Hadoop on Nutanix Reference Architecture," 2012.
[29] Emc, "Hadoop on Emc Isilon Scale-Out Nas," 2012, no. December 2012.
[30] D. Kimmig and A. Schmidt, "DataSys 2013 - Tutorial The Hadoop Core – Understanding Map Reduce and the," 2013.
[31] D. Heger, "Hadoop Performance Tuning-A Pragmatic & Iterative Approach," C. J., pp. 1–16, 2013.
[32] C. W. Olofson and D. Vesset, "Worldwide Hadoop-MapReduce Ecosystem Software 2012 – 2016 Forecast," no. May 2012, 2016.
[33] VMware, "Virtualized Hadoop Performance with VMware vSphere 5.1 - Performance Study - TECHNICAL WHITE PAPER," pp. 1–20, 2012.
[34] Sun Microsystems Inc., "Using Lustre with Apache Hadoop Overview and Issues with Hadoop + HDFS," System, pp. 1–25, 2010.
[35] M. Dunn, "Parallel I / O Testing for Hadoop," 2010.
[36] "Understanding_Hadoop_Clusters_and_the_Network-slides_and_text.pdf." .
[37] A. F. Different, A. To, and E. Analytics, "Modernizing Hadoop Architecture for Superior Scalability , Efficiency & Productive Throughput . The Impetus For Today ' s Hadoop Design."
[38] L. X. and J. L. and J. Wei, "FMEM: A Fine-grained Memory Estimator for MapReduce Jobs," 10th Int. Conf. Auton. Comput. ICAC'13, San Jose, CA, USA, June 26-28, 2013, pp. 65–68, 2013.
[39] Y. Kang, Y. S. Kee, E. L. Miller, and C. Park, "Enabling cost-effective data processing with smart SSD," in IEEE Symposium on Mass Storage Systems and Technologies, 2013.
[40] J. Lin and C. Dyer, "Data-Intensive Text Processing with MapReduce," Synth. Lect. Hum. Lang. Technol., vol. 3, pp. 1–177, 2010.
[41] M. Mihailescu, G. Soundararajan, and C. Amza, "MixApart: Decoupled Analytics for Shared Storage Systems," Proc. 11th USENIX Conf. File Storage Technol., pp. 133–146, 2013.
[42] M. Zaharia, D. Borthakur, J. Sen Sarma, K. Elmeleegy, S. Shenker, and I. Stoica, "Delay scheduling: a simple technique for achieving locality and fairness in cluster scheduling," EuroSys, pp. 265–278, 2010.