International Research Journal of Computer Science (IRJCS)
Issue 05, Volume 3 (May 2016)
ISSN: 2393-9842
www.irjcs.com

A PROPOSED AGENT BASED STORAGE VIRTUALISATION APPROACH: USING THE MAP-REDUCE PROGRAMMING MODEL

BENARD ONG'ERA OSERO
Chuka University, Department of Computer Science,
Chuka, Kenya

Abstract — Increasing performance and decreasing cost of microprocessors are making it feasible to move more processing power to the data source. This allows us to investigate new methods of storage delivery and storage management that were not plausible in the past. Our architecture, inspired by agent-based techniques and active disk technology, promotes an open storage management framework that embeds functionality into storage devices. We use local agents to implement self-control with automated capability that can be dynamically adapted to meet the storage management through improved search capabilities in the virtual environment and also retain capabilities like: security, performance and availability requirements.

Keywords: Mobile Agents, Map reduce, metadata, Storage Virtualization, data latency

I. INTRODUCTION

Storage virtualization seems to be a silver bullet squarely aimed at intractable cost-efficiency issues prevalent across most data centers today. Amid the ever-rising complexities, risks and expenses of managing data storage resides the promise of a real remedy that achieves business benefits through a more economical IT architecture [1].

Efficient, cost-effective information management is one of the top measures of business health. As IT leaders persistently pursue healthier data infrastructures, storage virtualization has evolved into a proven method for getting there faster [1].

A. BACKGROUND

Virtualization is a powerful feature that plays a role in the current success of storage arrays. By design, virtualization manages where data is located and controls access to data for users and applications. The value of storage has moved from disk drives to the array controller as more features and data protection capabilities have been added over time from the array to the point of virtualization [2].

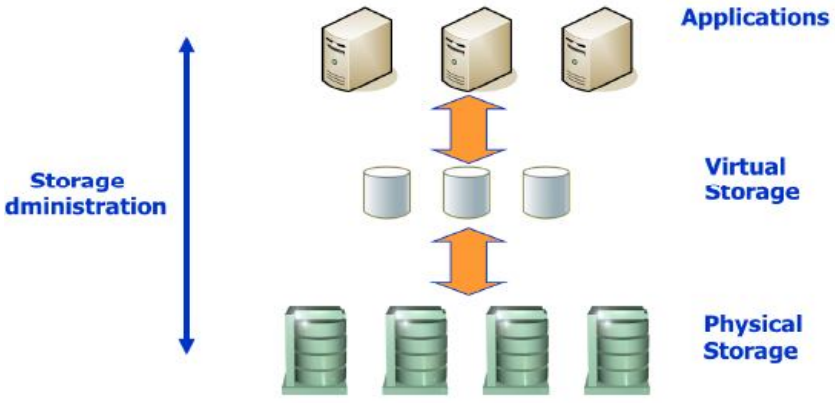

Fig.1. SAN virtualization source. [2]

Virtualization provides logical representations of physical resources while preserving the usage interfaces of those resources. Virtualization techniques can remove resource limits while improving utilization. Virtual memory and virtual networks have existed in the technology industry for years [3].

II. CURRENT NETWORK ATTACHED STORAGE (NAS) SYSTEMS.

A. NETWORK-ATTACHED SECURE DISKS (NASD).

Network-Attached Secure Disks (NASD), which separates store and forward copying work and management. By modifying the interface for commodity storage devices, we eliminate the server resources mainly required for data movement.

Figure 2 the major components of NASD ARCHTECTURE are outlined in the figure below [8][23].

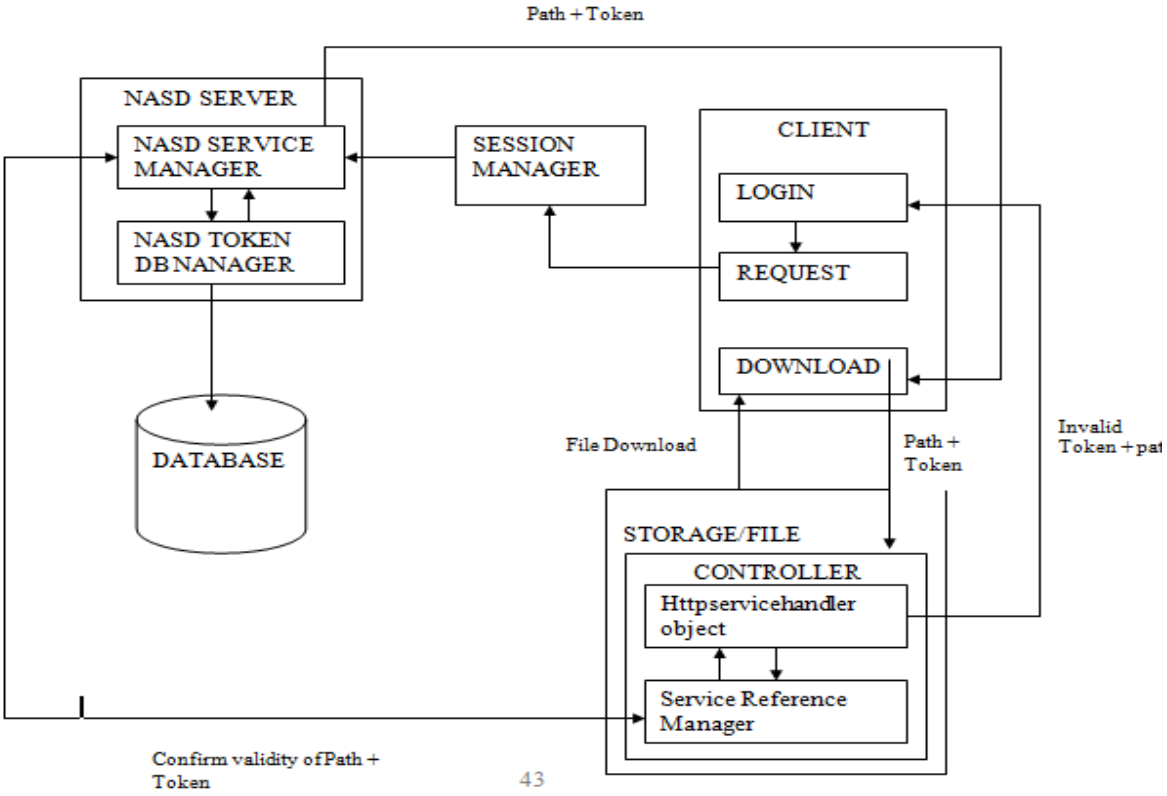

Fig.2. The major components of NASD ARCHTECTURE

III. HOW THE ABOVE NASD MODEL WORKS.

NASD reduces the overhead on the file server (file manager) by allowing storage devices to transfer data directly to clients. Most of the file manager's work is offloaded to the storage disk without integrating the file system policy into the disk. Most client operations like Read/Write go directly to the disks; less frequent operations like authentication go to the file manager. Disks transfer variable-length objects instead of fixed-size blocks to clients. The File Manager provides a time-limited cacheable capability for clients to access the storage objects. A file access from the client to the disks has the following sequence [7][8]:

1. The client authenticates itself with the file manager and requests for the file access.
2. If the client can be granted access to the file requested, the client receives the network location of NASD disks and their capability.
3. If the client is accessing the disk for the first time, it receives a time-limited key for the establishment of secure communication to the disk.
4. The file manager informs the corresponding disk using an independent channel.
5. From now on, the client directly accesses the NASD disks by giving the capability it received and further data transfers go through the network, bypassing the file manager.

The above architecture shows a framework of a NASD model, when the client requests data from a storage area then the request is sent through the controller to the file-server which has the file manager network protocol and the access and security policy control semantics. If the client satisfies the security requirements then it will be allowed to get an access path from the network storage pool in the file server and then eventually download the file from the storage area directly to the client [7][8].

IV. OBJECT-BASED STORAGE (OSD)

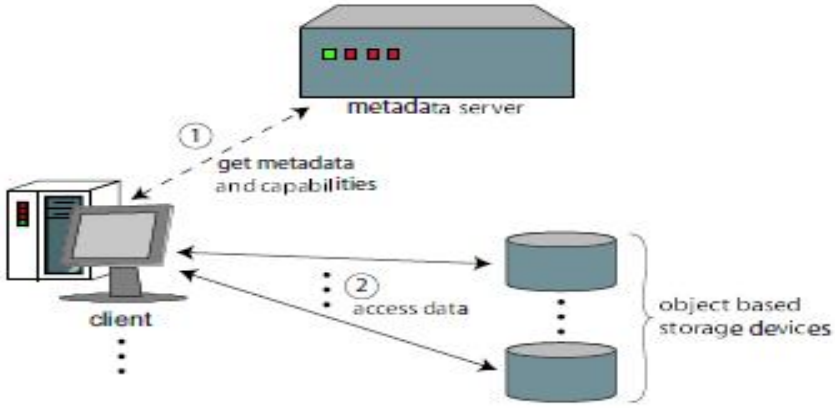

Fig 3: Direct client access.

First, a client interacts with the metadata server to obtain mapping information (e.g., which object IDs to access and on which storage devices) and capabilities (i.e., evidence of access rights). Second, the client interacts with the appropriate storage device(s) to read and write data, providing a capability with each request [9].

Although metadata is usually less than 10% of the overall storage capacity of a file system, its operations represent 50–80% of all the requests. Metadata operations are also very CPU consuming, and a single metadata server can easily be overloaded by a few clients in a parallel file system. Hence, to improve the performance and scalability of metadata operations, a cluster of servers is needed. PVFS and Ceph, for instance, use a small set of dedicated servers as metadata cluster, and Lustre expects to provide a similar production-ready service for version 2.2[16].

The metadata cluster of FPFS uses OSD+ devices to provide a high performance and scalable metadata service. It also profits the enhanced intelligence of the OSD+s to tackle with directory renames, links and permission changes, in a consistent and atomic manner. Communications between clients and OSD+s are established via TCP/IP connections and request/reply messages. Each OSD+ will launch one thread for attending the requests of a client, and for performing the operations on the local disk on behalf of the client. This way, the workload generated by the clients is reflected on the servers, which can be configured accordingly. Note that file systems like Lustre limit the number of threads on the metadata server, and that this number is usually much smaller than the number of clients; this decision distorts the clients' workload that the server sees [16].

1. **SOLVING THE PROBLEMS IN OSD:** Server-driven metadata prefetching and namespace flattening can address the above problems with minimal changes to the object-based storage architecture [9].Server-driven metadata prefetching: Rather than returning metadata for just one object, when queried, the metadata server should return metadata for other related objects as well. By doing so, it allows the client to populate its cache with additional mapping information and capabilities—a form of prefetching, but orchestrated by the metadata server. The client still determines what to keep and replace, but the server determines what to prefetch. When the necessary metadata is in its cache, a client can access the storage device immediately. Thus, if the right additional metadata is returned, the number of metadata server interactions should drop dramatically.

V. AGENT BASED DISTRIBUTED NETWORKS

Mobile agents are considered a very interesting technology to develop applications for mobile, pervasive, and distributed computing. Thus, they present a combination of unique features, such as their autonomy and capability to move to remote computers to process data there and save remote communications. Many mobile agent platforms have been developed since the late nineties. While some of them have been outdated, others continue releasing new versions that fix bugs detected or offer new interesting features. Moreover, other new platforms have appeared in the last few years. So, a common problem when one wants to benefit from mobile agent technology to develop distributed applications is the decision about which platform to use. In this research paper, we provide an up-to-date evaluation of existing Mobile Agent platforms [18].

A Mobile Agent is an emerging technology that is gaining momentum in the field of distributed computing. The use of mobile agents can bring some interesting advantages when compared with traditional client/server solutions, it can reduce the traffic in the network, it can provide more scalability, it allows the use of disconnected computing and it provides more flexibility in the development and maintenance of the applications. In the latest years, several commercial implementations of mobile agent systems have been presented in the market [18]

A. SEVEN GOOD REASONS FOR USING MOBILE AGENTS.

Although mobile agent technology sounds exciting, our interest in mobile agents should not be motivated by the technology *per se*, but rather by the benefits they provide for the creation of distributed systems. So here are seven good reasons for you to start using mobile agents [19]:

i) They reduce the network load. Distributed systems often rely on communications protocols that involve multiple interactions to accomplish a given task.

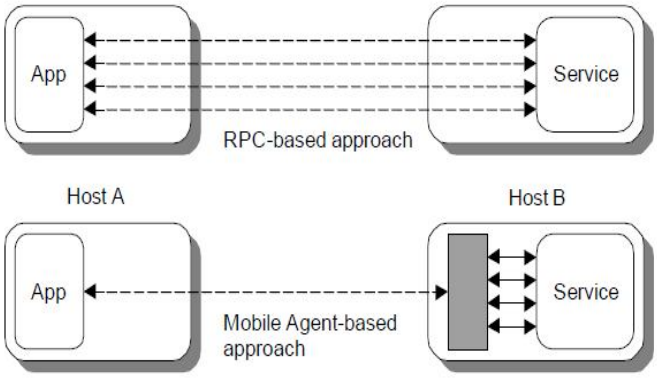

Fig.4. Mobile Agents Reduce Network Load [19]

ii) They overcoming network latency. Critical real-time systems such as robots in manufacturing processes need to respond to changes in their environments in real time. [19].

iii) They encapsulate protocols. When data are exchanged in a distributed system, each host owns the code that implements the protocols needed to properly code outgoing data and interpret incoming data, respectively. [19].

iv) They execute asynchronously and autonomously. Often mobile devices have to rely on expensive or fragile network connections. That is, tasks that require a continuously open connection between a mobile device and a fixed network will most likely not be economically or technically feasible. [19].

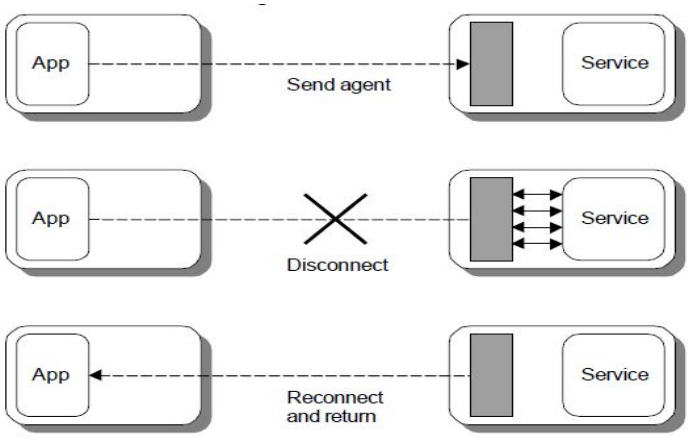

Fig.5. Mobile Agents Allow Disconnected Operation [19]

v) They adapt dynamically. Mobile agents have the ability to sense their execution environment and react autonomously to changes. [19].

vi) They are naturally heterogeneous. Network computing is fundamentally heterogeneous, often from both hardware and software perspectives. As mobile agents are generally computer- and transport-layer-independent, and dependent only on their execution environment, they provide optimal conditions for seamless system integration [19].

vii) They are robust and fault-tolerant. The ability of mobile agents to react dynamically to unfavorable situations and events makes it easier to build robust and fault-tolerant distributed systems. If a host is being shut down, all agents executing on that machine will be warned and given time to dispatch and continue their operation on another host in the network [19].One of the main characteristics of the mobile agent systems is the ability to transport not only the data but also the code to the different places of the agent's itinerary. Thereby, the system should be able to fetch the agent code in a flexible and efficient way. [20].

B. The Limitations of Client/Server

Historically, distributed applications such as these are created with "client/server" programming. In this model, an operation is split into two parts across a network, with the client making requests from a user machine to a server which services the requests on a large, centralized system. A protocol is agreed upon and both the client and server are programmed to implement it. A network connection is established between them and the protocol is carried out.

The client/server model has the advantage of enabling the removal of the client to smaller, remote machines, and it works well for certain applications. However it breaks down under other situations, including highly distributed systems, slow and/or poor quality network connections, and especially in the face of changing applications. In a system with a single central server and numerous clients, there is only a problem of simple scaling. When multiple servers become involved, the scaling problems multiply rapidly, as each client must manage and maintain connections with the multiple servers. [6].

C. Mobile Agents to the Rescue

Mobile agents overcome all these inherent limitations in client/server [6]. The Current management models based on client/server suffer from scalability and flexibility problems. Furthermore, the staleness of gathered data (due to network latency involved) and probable error in the selection of management task being carried over (owing to the staleness of data) reduces the reliability of the management applications. In this sense, use of mobile agents offer many possibilities for designing the next generation of distributed network management systems by reducing the management traffic around the management station and distribute processing load. [21] Various mobile agents based network management models, the key advantages of mobile agents in the distributed network management systems and present a mathematical model for the purpose of comparing client/server vs. mobile agent paradigms in terms of responsiveness and traffic generated around management station [21].

Server administration becomes a matter simply of managing systems and monitoring local load. The problem of robust networks is greatly diminished, for several reasons. The hold time for connections is reduced to only the time required to move the agent in or out of the machine. Servers can be upgraded, services moved, load balancing interposed, security policy enforced, without interruptions or revisions to the network and clients. All in all, a significant advantage in Mobile Agents [6]

VI. MAP REDUCE ALGORITHM

Map Reduce is a framework for processing and managing large-scale datasets in a distributed cluster, which has been used for applications such as generating search indexes, document clustering, access log analysis, and various other forms of data analytics. Map Reduce adopts a flexible computation model with a simple interface consisting of *map* and *reduce* functions whose implementations can be customized by application developers. Since its introduction, a substantial amount of research effort has been directed toward making it more usable and efficient for supporting database-centric operations [52].

The fundamental idea is to simplify the parallel processing using a distributed computing platform that offers only two interfaces: map and reduce. Programmers implement their own map and reduce functions, while the system is responsible for scheduling and synchronizing the map and reduce tasks. The Map Reduce model can be used to solve the “embarrassingly parallel” problems,1 where little or no effort is required to partition a task into a number of parallel but smaller tasks. Map Reduce is being used increasingly in applications such as data mining, data analytics, and scientific computation. Its wide adoption and success lie in its distinguishing features, which can be summarized as follows [52]:

i) Flexibility. Since the code for map and reduce functions are written by the user, there is considerable flexibility in specifying the exact processing that is required over the data rather than specifying it using SQL. Programmers can write simple map and reduce functions to process petabytes of data on thousands of machines without the knowledge of how to parallelize the processing of a Map Reduce job.
ii) Scalability. A major challenge in many existing applications is to be able to scale to increasing data volumes. In particular, *elastic scalability* is desired, which requires the system to be able to scale its performance up and down dynamically as the computation requirements change. Such a “pay-as-you-go” service model is now widely adopted by the cloud computing service providers, and Map Reduce can support it seamlessly through data-parallel execution. Map Reduce was successfully deployed on thousands of nodes and able to handle petabytes of data.
iii) Efficiency. Map Reduce does not need to load data into a database, which typically incurs high cost. It is, therefore, very efficient for applications that require processing the data only once (or only a few times).
iv) Fault tolerance. In Map Reduce, each job is divided into many small tasks that are assigned to different machines. Failure of a task or a machine is compensated by assigning the task to a machine that is able to handle the load. The input of a job is stored in a distributed file system where multiple replicas are kept to ensure high availability. Thus, the failed map task can be repeated correctly by reloading the replica. The failed reduce task can also be repeated by re-pulling the data from the completed map tasks.

Map reduce has been applied in solving various distributed systems problems such as Data mining models for Internet of things [53] Data Mining [54] Internet of things IOT [55].

VII. PROBLEM STATEMENT

Like the client server problems discussed in [6], The NASD and OSD systems[7] [8][9][10][11](12)[13][14][15] discussed in the literature, fall short of handling high latencies involved in the interactions between the client and the virtual objects in the server because they employ granularly threaded transactions between the client and server thus wasting the bandwidth and also encountering high latencies because every client has to make a request for the resource from the virtual servers. Although [16] tried to provide solution to bandwidth problem that is encountered when the client server makes interactions by the use of, directory management scheme, this is a partial solution although it improves on bandwidth utilization but not latencies. [7][10][9]The employ unsorted storage metadata blocks which are quite inefficient in data handling; since much time has to be taken in searching and mapping the metadata information to the required client.

Various attempts have been made by [17] in metadata prefetching and catching but the current predictive prefetching algorithms are for data but not metadata. Although the metadata prefetching is better than centralized client –server systems, since the algorithm uses a metadata relationship graph to assist prefetching decision making. The relationship graph is used to dynamically represent the locality strength between predecessors and successors in metadata access streams

The metadata prefetching algorithm can be improved by use of mobile agents as indicated by [18] [19][20][6][21]. Only [21] has attempted to apply agents on distributed network environment but not on distributed objects such as NASD and OSD models. In their conclusion [21] shows that Mobile agents offer an easy re-configurable, flexible and scalable solution to the management of today's complex telecommunication networks thereby reduces the number of necessary human interactions. By using mobile agents on a distributed network [6][21] shows that it can solve the problems of high bandwidth and latency requirements, which is a penalty to the performance of a distributed object based parallel network, encountered during the interaction between the Client and virtual Server.

VIII. PROPOSED SOLUTION

From the above study we propose a method of using search mechanism to the existing metadata resource storage pool enhanced by the map reduce algorithm that will sort the metadata blocks according to the client IP address domains before mapping them to the mobile agent code and eventually migrated to one elected client that now will take over as a local server, this will improve on the locality of reference problem. The mobile agent will fetch the sorted domain based resource pool and migrate it to the one of the elected local servers where it will execute henceforth, this will be terminated if this particular local server terminates normally or it is terminated by the parent server if in case the; the local server uses the resource not allocated to it or issues instructions beyond its allocated mandate or a critical unrecoverable event happens.

The clients within a particular domain will get the resource path indicating where a certain physical resource is located in the storage area network physical disks as long as the requests are valid. The local server will have the potential of enforcing their local security mechanisms to be able to protect the clients within a particular domain. We will carry out various experiments to prove whether our solution is better than the existing methods as indicated in the methodology.

IX. CONCEPTUAL FRAMEWORK

To address the gaps that exist between mobile agents and network attached disks that have not yet been fully exploited; a more intelligent, self managed and secure storage environment has been proposed as shown in the following model.

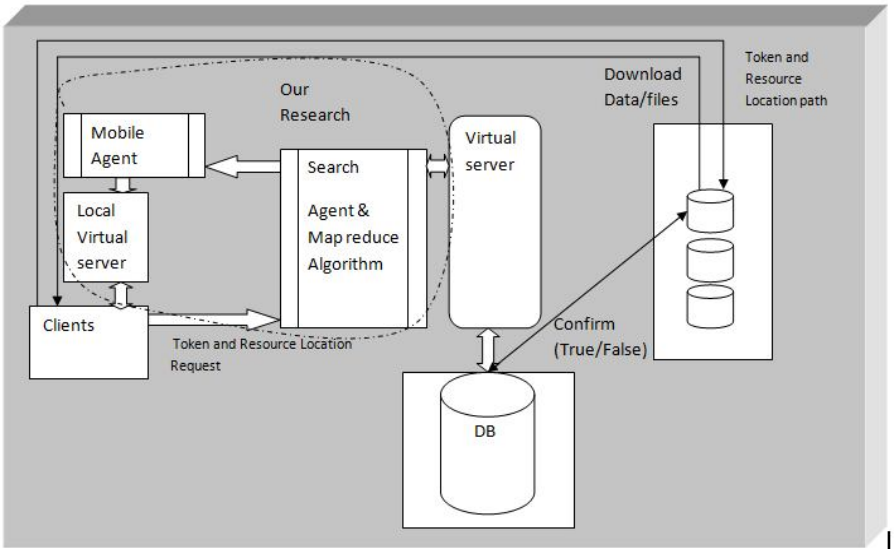

Fig. 6. A proposed intelligent object based system using agents.

A. IMPORTANCE OF THE PROBLEM

1. It will give a basis for the study of the need for mobile agents on distributed systems environments.
2. Internet of things (IOT) is an emerging trend to the future object interactions on the web, this particular technology will require that every object will interact with every other object, hence massive physical memory/storage requirements. Therefore intelligent virtualized storage solutions will come in handy to mitigate such huge memory demands.
3. Bandwidth like storage is a scarce resource that needs to be utilized, hence the need for mobile agents which can easily work offline.
4. This research will open up more space for distributed storage and give direction to future storage systems.

REFERENCES

[1]. Hitachi data systems. Storage virtualisation:How to capitalize on its economic benefits. April 2016.
[2]. al, Randy et. San virtualisation evaluation guide. s.l. : Evaluator group, 2011.
[3]. EMC2. Where information lives:current benefit and future potential technology concepts and business considerations. s.l. : EMC2, January 2008.
[4]. Mobile agent enabled application mobility for pervasive computing. al, J ma et. Springer-Velag Berlin Heidelberg : UIC 2006, 2006. LNCS.
[5]. Storage Virtualisation, What is it all about? al, Mark et. s.l. : IBM International technical support, 2000.
[6]. Concordia White paper. Concordia. [Online] Upenn.edu, March 17, 2016. [Cited: 3 17, 2016.] https://www.cis.upenn.edu/bcpierce/629/papers/Concordia-Whitepaper/.
[7]. Osero, B.O. Network Storage Virtualisation and management.[online]www.ijern.com. Chuka, Kenya : International Journal of Education and Research, IJER, 2013. ISSN:2201-6333 (Print) and 2201-6740 (online).
[8]. osero. Storage virtualisation and management. nairobi : University of Nairobi, 2010.
[9]. Improving small file performance in object based storage. al, James et. CMU-PDL-06-104, Pittsburg, PA 15213-3890 : Parallel Data Laboratory, 2006.
[10]. Object Storage:The future building block for storage systems. Michael Factor, kalman Meth, Dalit Naor, Chad Rodeh, and Julian Satran. Sardinia, Italy : IEEE 2005, 2005.
[11]. Object based storage. Mike Meisner, Gregory R. Ganger and Erick Riedel. s.l. : IEEE,Communications Magazine, August 2003.
[12]. OSBF:A file system for object based storage devices. Feng Wang, Scott A. Brandt, Ethan L. Miller and Darrel D.E.Long. Adelphi MD : NASA Goddard/IEEE Conference on Mass storage systems and technologies, 2004.
[13]. Lustre. http://www.lustre.org.
[14]. INC, Panasas. http:www.panasas.com.
[15]. Solutions, Permon. Parmabit INC,http://www.parmabit.com.
[16]. Scalable metadata management through OSD+devices. Ana Avile's-Gonza'lez, Juan Piermas,Pilar gonzale'z Fer'ez. Murcia, Spain : Springer Science+Business Media LLC 2012, July 2012.
[17]. Peng Gu, Ju Wang, Yifeng Zhu, Hong Jiang, Penglu Shang. A novel weighted-Graph Based grouping Algorithm for metadata prefetching. University of Nebraska-Lincoln : CSE JOURNAL, 2010.
[18]. Analysis of mobile agents. Prakash v. Rajguru, Sushant B. Deshmukh. Hingoli,Marahashtra India : Journal of global research in Computer Science,, 2011, Vol. Volume 2.
[19]. Mobile objects and mobile agents:The future of Distributed computing. [book auth.] Lange Oshima. Promming and deploying Java mobile agents with aglets Addison wesley 1998 (ISBN:0-20-201-32582-9). Sunnyvale, Carlifonia : General Magic Inc, 1998.
[20]. Optimizing Migration of Mobile agents. Guilherme Soares, Louis Moura Silva. Coimbra, Portugal : MATA , 1999.
[21]. Application of Mobile Agent in Distributed Network Management;YMCA University of science and technology, Conference on Communication systems and network. Atul Mishra, A.K. Sharma. Haryana, India : IEEE, 2012. DOI 10.1109/CSNT.2012.198.
[22]. C.R, Kothari. Reseach Methodology:Methods and techniques. Daryaganj,New dheli : New age international publishers, 2004.
[23]. A cost effective high bandwidth storage archtectures. Arctectural support for programming languages and operating systems. Garth A. Gibson, David F. Nagle, Khalil Amiri, Jeff Butler, Fay W. Chang, Howard Gobioff, Charles. San Jose CA 3-7 : SIGPLAN Notices 33[11], 1998.
[24]. Information technology-SCSI object-Based storage device commands (OSD). weber, Ralph O. s.l. : Ralph O. Weber, July 2004.
[25]. CORP, EMC. Content addressed storage systems http:www.emc.com/products/systems/centera.jsp?openfolder=platform. s.l. : EMC.
[26]. Active Disks Remote Execution for Network Attached Storage. Riedel, Erik.
[27]. "CATALINA":A smart application control and management". al, Salim Hariri et. s.l. : AMS, 2001.

[28]. Self Managing Storage System-Design and evaluation,http://acl.ece.arizona.edu/projects/old/smp/smppaper.pdf,downloaded 25/3/2016; 1:30 PM. Srinivas Singavarapu, Salim Hariri, Mazin Yousif. Arizona : s.n.
[29]. "low complexity video coding for receiver driven layered multicast" . Steven McCanne, martin Vetterli and Van Jacobson. s.l. : IEEE journal on selected areas in communications, vol 16, 1997.
[30]. Al-Shishitwy, Ahmad. Self-Management for large scale distributed system. Stockholm, Sweden : Royal institute of Technology, 2012.
[31]. Saulsbury, Ashley. Attacking Latency bottlenecks in distributed shared memory. 1999.
[32]. Simsarian, Kristian. Toward Human Robot Collaboration . 2000.
[33]. Towards a unified object storage foundation for scalable storage systems. al., Cengiz Karakoyunlu et. 978-1-4799-0898-1/13, USA : IEEE, 2013.
[34]. A Scalable High performance distributed file system.[online] available http://www.lustre.org/docs/whitepaper.pdf. Lustre. s.l. : Cluster file systems Inc., 2002.
[35]. GPFS:"A shared disk file system for large computing clusters," in proceedings of 1st USENEX conference on file and storage technologies. R.Haskin, F schumuck and. Berkeley,CA USA : USENEX association, 2002.
[36]. Scalable Performance of the Panasas parallel file system-Usenex conference on file and storage technologies pp 17-33. B welch, M. Unangst, z. Abbasi, G. gibson, B.mueller,J.Small, J. Selenka and B.Zhou. s.l. : FAST, 2008.
[37]. A parallel file system for linux clusters, in proceedings of 4th Annual linux showcase and conference . P.H Carns, W.B. ligon III, R.B. Ross, and R. Thakur. Antlanta : USENIX Association, 2000.
[38]. Ceph: A scalable high performance distributed file system, 7th symposium on operating systems design and implementation . S.A Weil, S.A Brandt, E.L Miller, D. E Long and C. Maltzahn. s.l. : OSDI, 2006.
[39]. Amazon simple storage system, [online]. Available http://aws.amazon.com/s3/. Amazon.
[40]. Swifuret: A storage archtecture for large objects. Long, L-F Cabre and D.D. E. Santa Cruz : Tech-Rep, 1990.
[41]. librados. librados API documentation.[Online] Availabe http://ceph.com/docs/master/api/librados/.
[42]. "RADOS: A fast Scalable, and Reliable storage services for petabyte-scle storage clusters. S.A weil, A Leung, S.A. Brandt and C. Maltzahn.
[43]. "The Google File System," in proceedings of the nineteeth ACM symposium on operating systems principles.[online] Available http://www.cs.rochester.edu/sosp2003/papers/p125-ghemawat.pdf. S.Ghemawal, H. Gobioff and S-T Leung. s.l. : ACM press, 2003.
[44]. The Hadoop distributed systems and technologies(MSST).[online] available:http://dx.doi.org/10.1109/MSST.2010.5496972. K Shvachko, H Kuang, S. Radia and R Chansler. Washington DC, USA : IEEE Computer Society, 2010.
[45]. Amazon. Amazon DynamoDB.[online]. Available:http:aws.amazon.com/dynamoDB/.
[46]. [online]. Available:http://redis.10/. Redis.
[47]. A searcheable distributed Key-value store[online]Availabe:http://hyperdex.org/. Hyperdox.
[48]. "Cassandra: A decentralised structured storage system". Malik, A. Lakshman and p. s.l. : Sigpos oper.Sys.
[49]. The definitive guide.[online] available:http://proquest.safaribooksonline.com/9781449314682. L. George, H base. 2011.
[50]. Big query.[online]. Available:https//developers.google.com/bigquery/. Query, Big.
[51]. Evolution towards distributed storage in a nutshell-IEEE (International conference on high performance computing and communication (HPCC)),2014 IEEE 6th International Symposium on cyberspac safety and security (CSS) . Pistirica Solin Andrei, Asavgei victor,Geanta Horia, Moldorianu Alin, Neglu Catalin, Mocanu Moriana. Bucharest, Romania : IEEE, 2014.
[52]. Distributed Data management using mapreduce. al, Feng Li et. 46, Singapore : ACM, January 2014, Vol. 3.
[53]. Research on data mining models for the internet of things. Shein Bin, Liu Yuan,Wang Xiaoyi. China : IEEE, 2010, Vol. 3.
[54]. Research on distributed data stream mining in internet of things, International conference on logistics Engineering management and computer science. Xu Liancheng, Xun Jiao. Jinan, China : Antlantis Press, 2014.
[55]. Map-reduce based data processing on IOT-National institute of informatics. Satoh, Ichilo. Tokyo Japan : IEEE, 2014.
[56]. "A framework for evaluating storage system security". Erik Riedel, mahesh kallahalla & Ram swaminathan. s.l. : Helwett Packard Laboratories, 2002.