International Research Journal of Computer Science (IRJCS)
Issue 04, Volume 6 (April 2019)

ISSN: 2393-9842
www.irjcs.com

A VIRTUALIZED SCALABLE AGENT BASED NETWORK ARCHITECTURE (SABSA): USING MAP-REDUCE PROGRAMMING MODEL

BENARD ONG'ERA OSERO
Chuka University Department of computer science,
Chuka, Kenya
bosero@chuka.ac.ke

Dr. ELISHA O. ABADE,
University of Nairobi School of Computing and Informatics,
Nairobi, Kenya
eabade@uonbi.ac.ke.

Dr. STEPHEN N. MBURU,
University of Nairobi School of Computing and Informatics,
Nairobi, Kenya
smburu@uonbi.ac.ke

Manuscript History
Number: **IRJCS/RS/Vol.06/Issue04/APCS10080**
Received: 25, March 2019
Final Correction: 05, April 2019
Final Accepted: 10, April 2019
Published: April 2019
Citation: OSERO, ABADE & MBURU (2019). A Virtualized Scalable Agent Based Network Architecture (SABSA): Using Map-Reduce Programming Model. IRJCS:: International Research Journal of Computer Science, Volume VI, 72-83. **doi://10.26562/IRJCS.2019.APCS10080**
Editor: Dr.A.Arul L.S, Chief Editor, IRJCS, AM Publications, India

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: Store and Forward, Object Storage Devices, Agent, and Map-Reduce.

I. 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 (Randy, 2011). 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 (EMC2, January 2008). (Pedro Jose, 2011) analysed various algorithms in mobile and distributed networks; Mobile objects technology and localisation are the future of emerging networks technologies as shown in their matrix in fig 1 below: Applications that can follow mobile users when they change to a different environment, especially with the change of device and location, are in high demand by pervasive computing. Implementation of application mobility also depends on context-awareness and self-adaptation techniques (Ma, 2006).

ISSN: 2393-9842
www.irjcs.com

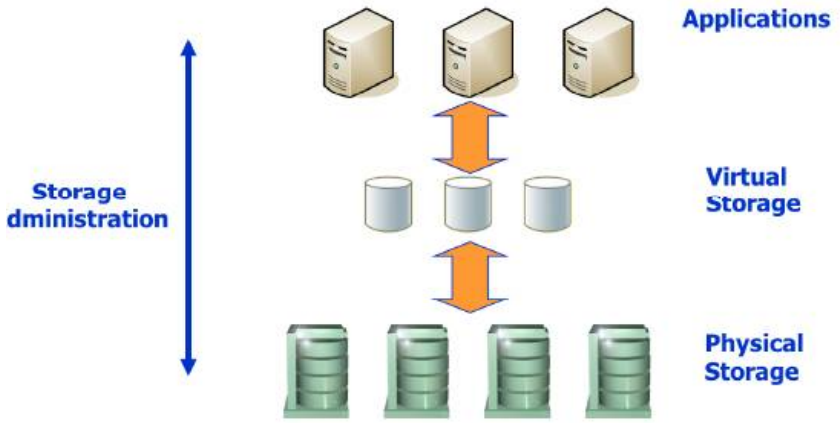

Fig 1.1 San Virtualization (Randy, 2011)

The development of Storage Area Networks (SANs) has passed its experimental phase and SAN has become a mature technology that allows businesses to implement storage pooling and sharing today. SANs are high speed switched networks designed to allow multiple computers to have shared access to many storage devices (Mark, 2000).

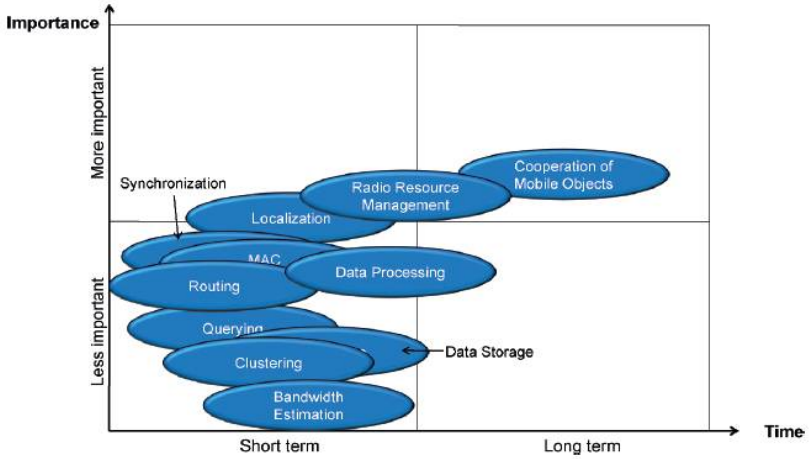

Fig 1.2 A survey matrix of various Distributed Network Algorithms (Pedro Jose, 2011).

1.1 Research Questions

Comparing the performance of the above storage methods and their associated data structure algorithms will help us answer the following research questions:

i. Can Metadata be cached on a distributed network and to what extent can the catching affect or improve latencies, performance, throughput and scalability?
ii. Does splitting of a distributed system into smaller parts improve its speedup and/or performance and how relevant is it in the metadata management in such systems?
iii. What role can a mobile agent play in the management and transfer of meta-data in a distributed virtualized storage environment?
iv. How can map-reduce functions assist meta-data management and in effect if used together with a mobile agent to what extent will they affect performance of the traditional Store and Forward (SAF) and Object Storage Devices (OSD)

1.2 Significance of the Study

The outcome of this research should be beneficial to several categories of entities. Specifically, the outcomes will:

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.

1.3 Problem with current systems and related work

Like the client server problems discussed in (Anon., 2016), The NASD and OSD systems (Osero, 2013) (Osero, 2010) (James, 2006) (Michael Factor, 2005) (Mike Meisner, August 2003) (Feng Wang, 2004) (Lustre, 2016) (Panasas, n.d.) (Permon, 2015) 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 (Ana Avile's-Gonza'lez, July 2012) 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. (Osero, 2013) (Michael Factor, 2005) (James, 2006) employed 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 (Peng Gu, 2010) in metadata pre fetching and catching but the current predictive pre fetching algorithms are for data but not metadata. Although the metadata pre fetching is better than centralized client –server systems, since the algorithm uses a metadata relationship graph to assist pre fetching decision making. The relationship graph is used to dynamically represent the locality strength between predecessors and successors in metadata access streams. The metadata pre fetching algorithm can be improved by use of mobile agents as indicated by (Prakash v. Rajguru, 2011) (Anon., 1998) (Guilherme Soares, 1999) (Anon., 2016) (Atul Mishra, 2012). Only (Atul Mishra, 2012) has attempted to apply agents on distributed network environment but not on distributed objects such as NASD and OSD models. In their conclusion (Atul Mishra, 2012) 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 (Anon., 2016) (Atul Mishra, 2012) 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.

II. WHY MAP REDUCE

The map-reduce model uses the key-value pairs where the records with the same key (i.e. word) are grouped together and eventually fed into the reduce function which then sums the input values and outputs the total number of occurrences in the given document (s) (Aditya B, 2012), thus map-reduce is simple and efficient for computing aggregate. The idea of map-reduce is not different from "filter and then group aggregation" query processing in a DBMS. The Major advantages of the Map-Reduce framework are as follows (Seema Maitrey, 2015):

- Simple and easy to use: No need to specify the physical distribution of the work across nodes but only define the work by use of map and reduce functions.
- Flexibility Map-Reduce: There is loose coupling between data model and schema thus it is possible for the programmer to deal with irregular or unstructured data more easily than they do with DBMS.
- Independence storage: Map-reduce does not rely on the underlying data models layers but can rather work with different storage layers such as Big Table and others.
- Fault tolerant: Map reduces can continue to work even if some failures have been encountered in the system.

III. SABSA CONCEPTUAL MODEL ARCHITECTURE

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 implemented.

3.1 SYSTEM REQUIREMENTS AND CONFIGURATION

The SABSA system Engine was configured to run on an AWS EC2 t2.micro virtual machine instance with: The host machine was a Dell Inspiron Machine, equipped with an Intel(R) Celeron(R) CPU 1.6 GHz CPU, 2GB RAM, a 500 GB hard disk drive, and a GSM wireless network connection. Besides we also created 1 vCPU in the Docker containers. The system was also hosted on the Amazon EC2 Cloud accessible via the following link [http://ec2-35-180-156-58.eu-west-3.compute.amazonaws.com]. The system requires installation of Docker on the virtual machine to run it. Configuration is done using Docker files which specify the VM image being used as well as run instructions for system setup. Instructions include setting environment variables used by the application, installing any packages used as well as exposing ports which need to be accessed from outside of the virtual machine.

3.2 VIRTUAL ENVIRONMENT SETUP AND DESIGN

The system is a three tiered model: The client side, the middleware/operations and storage. This virtual environment is designed as a bridge network that is built on the docker containers as virtual instances and storing the metadata information in the Redis database the system set up for our system was done as follows:

When running on a single host:

1. One needs to install docker. Installation instructions are found (Docker, 2017).
   i. For some versions of Window and Mac one may need to install Docker Toolbox instead.

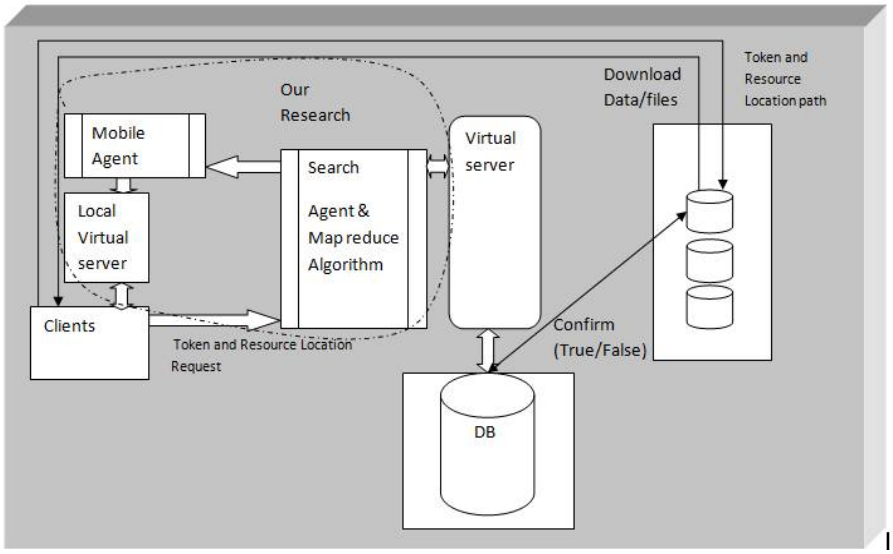

Fig.1.3. The SABSA architecture

2. Ensure docker-compose is installed, and if not do as instructed (Docker, 2017).
3. Open the root directory of the project
4. Open the src folder
5. To run all services run 'docker-compose up'
6. To access the system open http://localhost.

a. On windows one may need to run 'docker-machine ip' to get the ip address of the virtual machine When running on multiple hosts

1. Follow steps 1 to 4 from above
2. Modify the 'docker-compose.yml' file as required for your setup.

a. This mainly involves port mappings and environment variables for the remotes' hostnames and ports.
b. The environment variables used are REDIS_URL, SERVER_HOST, SERVER_PORT, SAN_HOST, and SAN_PORT.

3. Run 'docker-compose up […SERVICE_NAMES]' for the services you will run on your host. Service names include:
   a. san
   b. server
   c. client
   d. redisdb

3.3 DOCKER CONTAINER ARCHTECTURE

The SABSA Engine used Docker, an open source technology launched in 2013, running the Docker Engine. This technology leveraged on the existing computing concepts around containers and specifically LINUX; utilizing primitives such as cgroups and namespaces. Docker is a unique technology as it focuses on the requirements of developers and systems operators to separate application dependencies from infrastructure. Later LINUX partnered with Microsoft to bring the Docker containers and functionality to Windows Server now referred to as Docker Windows containers. Docker and its open source project, Moby has been leveraged by all major data center vendors and cloud providers. The figure 3.4 below shows a general architecture of Docker (Docker, 2019).

3.4 SABSA VIRTUAL NETWORK DESIGN

(Kasireddy, 2016) explains key parts of the docker engine as consisting of three main parts, our SABSA Engine employed all the three parts:

1. A Docker Daemon that runs in the host computer.
2. A Docker Client that then communicates with the Docker Daemon to execute commands.
3. A REST API for interacting with the Docker Daemon remotely.

The daemon is responsible for executing commands sent to the Docker Client—like building, running, and distributing your containers. The Docker Daemon runs on the host machine, but as a user, you never communicate directly with the Daemon. The Docker Client can run on the host machine as well, but it's not required to. It can run on a different machine and communicate with the Docker Daemon that's running on the host machine.

The Docker file is where the instructions were written to build a Docker image follows:

- RUN apt-get y install some-package: to install a software package
- EXPOSE 5010: to expose a port
- ENV ANT_HOME /usr/local/apache-ant to pass an environment variable.

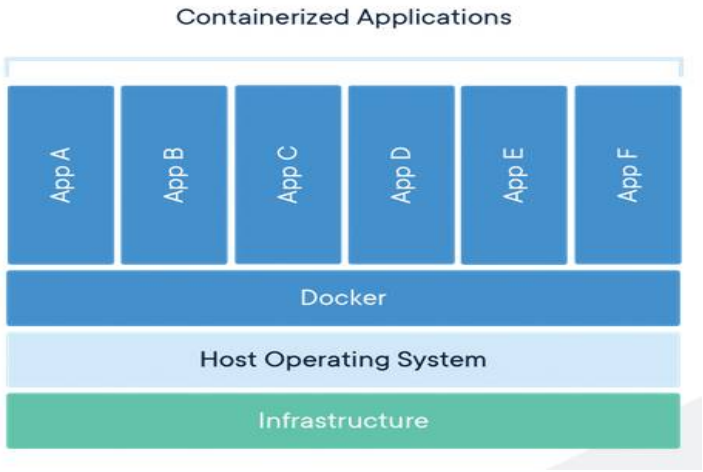

Fig 1.4 Docker containers Archtecture

Once the above file was running then we executed the **docker build** command to build an image. The Docker image was built using a Docker file. Each instruction in the Docker file added a new "layer" to the image, with layers representing a portion of the images file system that either adds to or replaces the layer below it as shown in the diagram below:

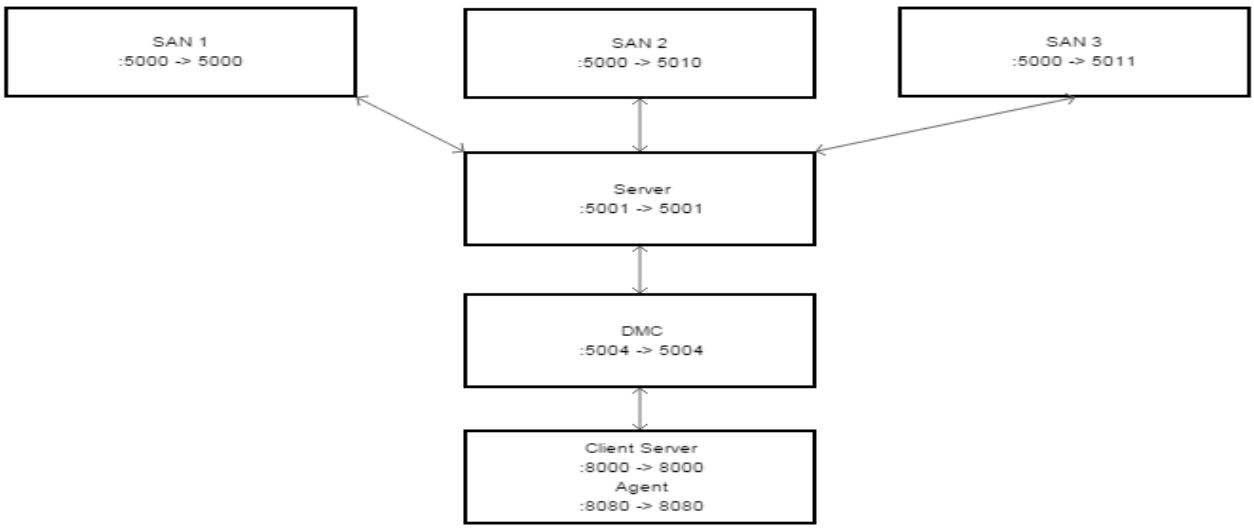

Fig 1.5 Port Configuration in the SABSA engine.

In figure 3.5 above we have three SANs and other code intstances distributed in different containers that were running in the SABSA Engine each of the container instances consisted of a port the port was configured in such a way that it had two sections (one for the container and the other for the container instance):

SAN included three container instances SAN 1, SAN 2 and SAN 3. Each of the three SANs was configured inside a global container running on port 5000, then each of the individual SAN instances was running as follows: SAN 1:port 5000 same as global container port, SAN 2: port 5010 and SAN 3: port 5011. Besides the three container objects in the SAN the othr instances in the containers included Server, DMC, Client-Server ports and Agents ports. The ports were also configured to run as follows: Server Port 5001, DMC port 5004, Client-Server port 8000 and agents were runnin on port 8080. (Kasireddy, 2016) in conclusion observed that a Docker container, wraps an application's software into an invisible box with everything the application needs to run which includes the operating system, application code, runtime, system tools, system libraries, and etc. The Docker containers are therefore built off Docker images. Since images are read-only, Docker adds a read-write file system over the read-only file system of the image to create a container. The figure below summarizes a general a sequence of network configuration steps of our SABSA model. The fig above consists of the following Components: SAN, Mobile Agents, Virtual Server, Client, Domain Controllers (DMC), and Storage. Virtualization and cloud technologies are being embraced in the support of various technologies over the www, we realize that most of the people have various intention some of which turns out to be malicious some of the activities of the attackers over the web may execute include but not limited to the following activities:

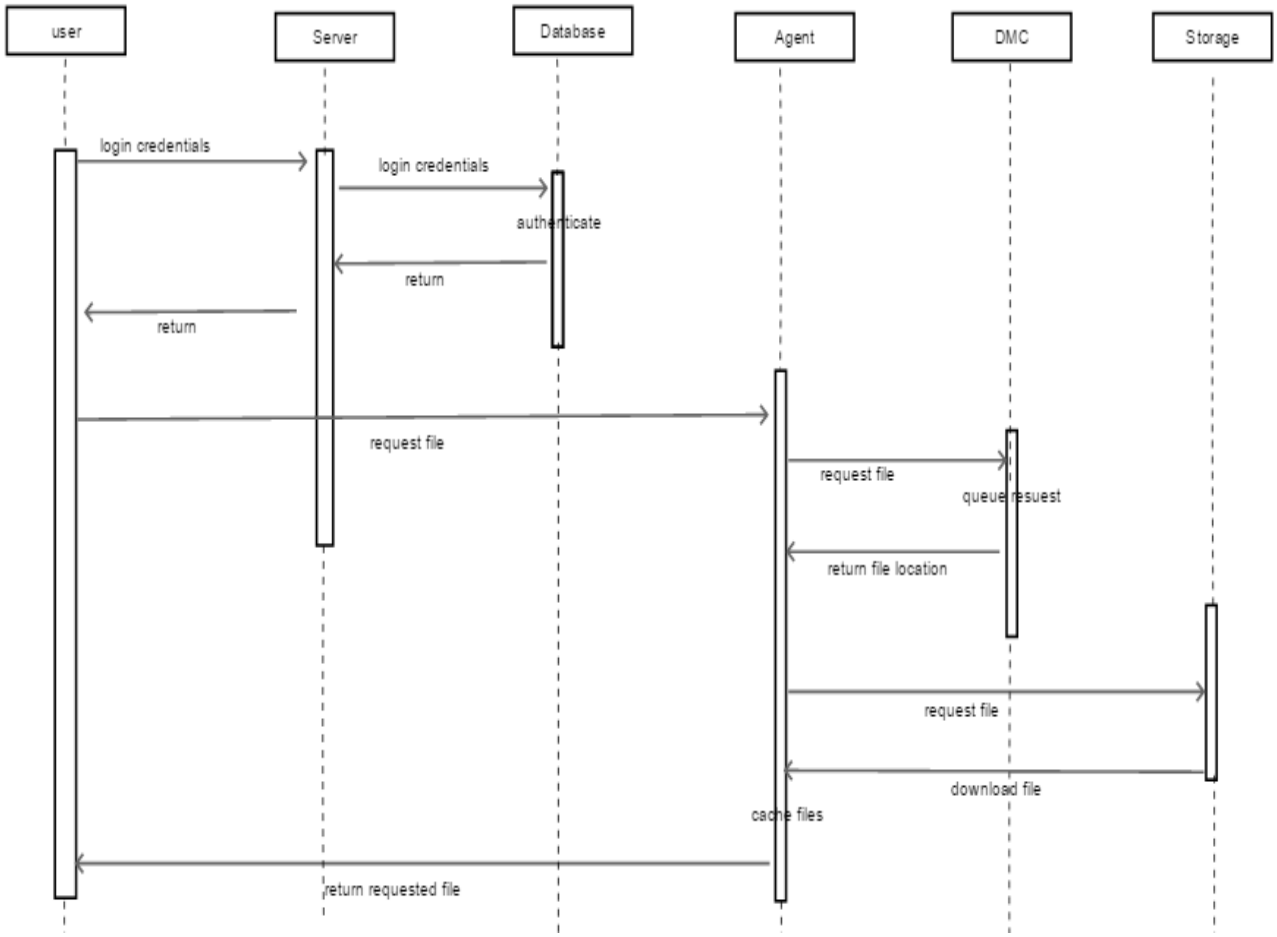

Fig 1.6: A Sequence diagram for Mobile Agent(s) with Domain Controller (De-Centralized DMC Control).

IV. SECURITY DESIGN

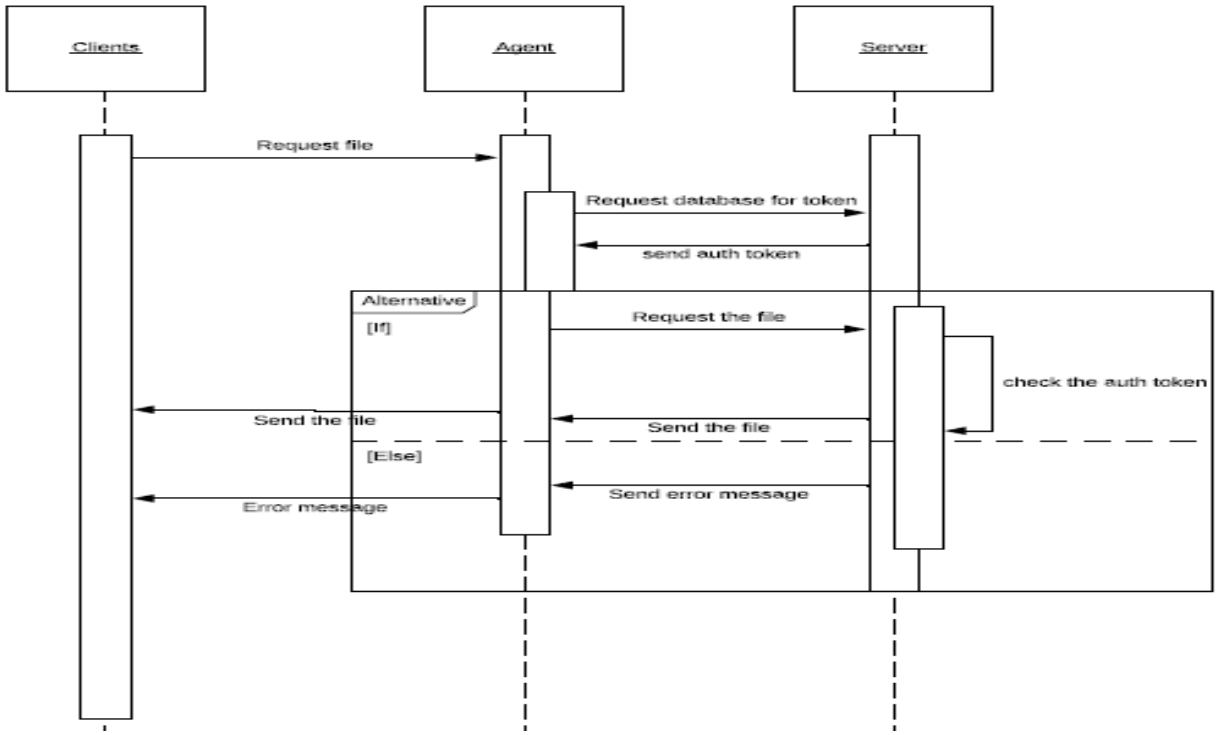

Fig 1.7 Security implementation on the SABSA model

- It is possible for an attacker to eavesdrop the communication between a user's browser and a web server.
- Sensitive information, such as a credit card number, or any other confidential data, could thus be obtained.
- A substantial amount of confidential information is made available via the www. Any unauthorized access to this information should be prevented.

In spite of the security challenges faced by most organizations; key industry players have developed security solutions into their software to ease the burden of the developers implementing weak solutions which might lead to attacks by hackers. (Walsh, 2014) explains in detail how the Docker environment security implimented, it also comes with rich security APIs to impliment various security modules. Besides the Docker environment security Modules we further employed the use of token based passwords encrypted; using a hash function SHA 256 on the client side in order to avoid malicious clients accessing the file, as shown in fig 3.10 and 3.11 below. Also for the Agent based platform the DMC authenticates the agents before they access the file or the file metadata. The diagrams below depicts the security diagrams for the security implementation on the SABSA model: Fig above shows that the SABSA model ensures that not only authenticated clients access the critical system resources but also that the request is valid as per the time stamped token session time; which is actually stored in the DB at the beginning of the session and retrieved at the request for confirmation from the SAN.

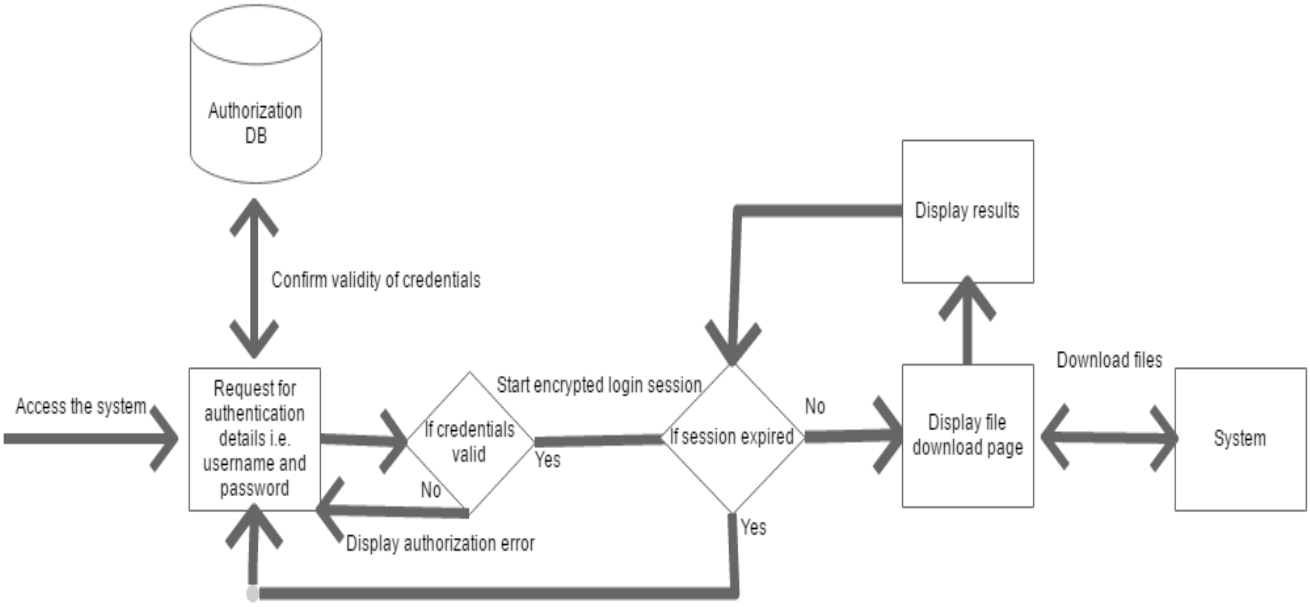

Fig 1.8 The SABSA General Authentication procedure

V. RESULTS

The following experiment was performed on our SABSA model to address the following research questions:

i. Can Metadata be cached on a distributed network and to what extent can the catching affect or improve latencies, performance, throughput and scalability?
ii. Does splitting of a distributed system into smaller parts improve its speedup and/or performance and how relevant is it in the metadata management in such systems?
iii. What role can a mobile agent play in the management and transfer of meta-data in a distributed virtualized storage environment?
iv. How can map-reduce functions assist meta-data management and in effect if used together with a mobile agent to what extent will they affect performance of the traditional Store and Forward (SAF) and Object Storage Devices (OSD).

The experiment was performed on data selected from three different SANS (SAN 1, SAN 2 AND SAN 3):The files in each SAN were also classified as small, medium and large-for this experiment we used the large file parameter in order to measure the effect on Latencies, throughput and scalability of different types of Sorted and Unsorted metadata functions on: SAF, OSD and Agent based distributed network, to achieve these 2 different files from each SAN were randomly selected, then downloaded 100 times and the results obtained from the SABSA Simulator were as follows:

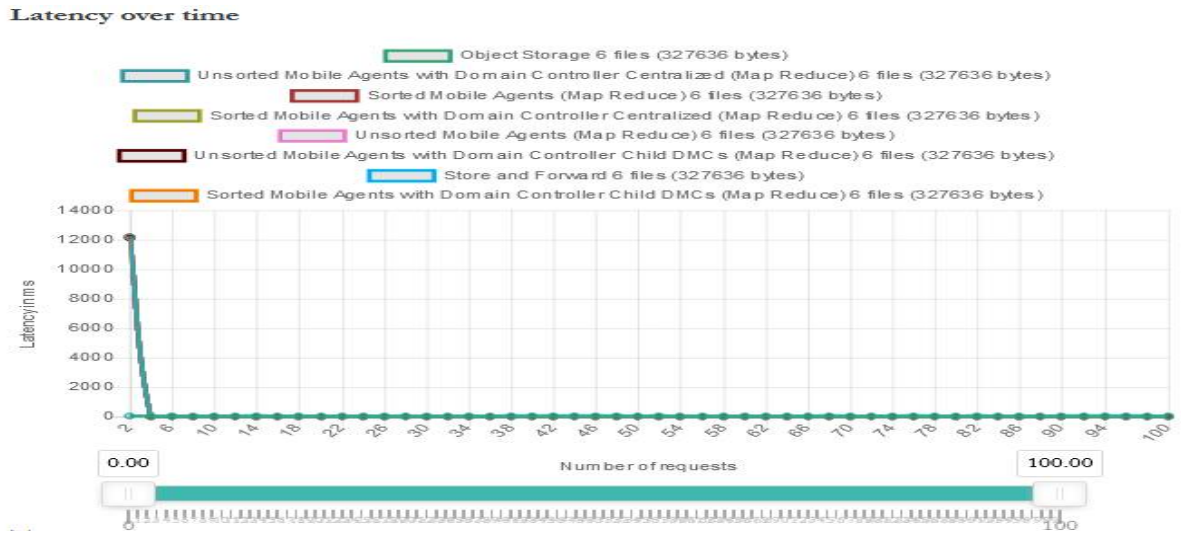

Fig 1.9: Latencies with time of 100 requests for OSD, SAF and Sorted, Unsorted Mobile Agents with Map-reduce distributed over three different SAN containers.

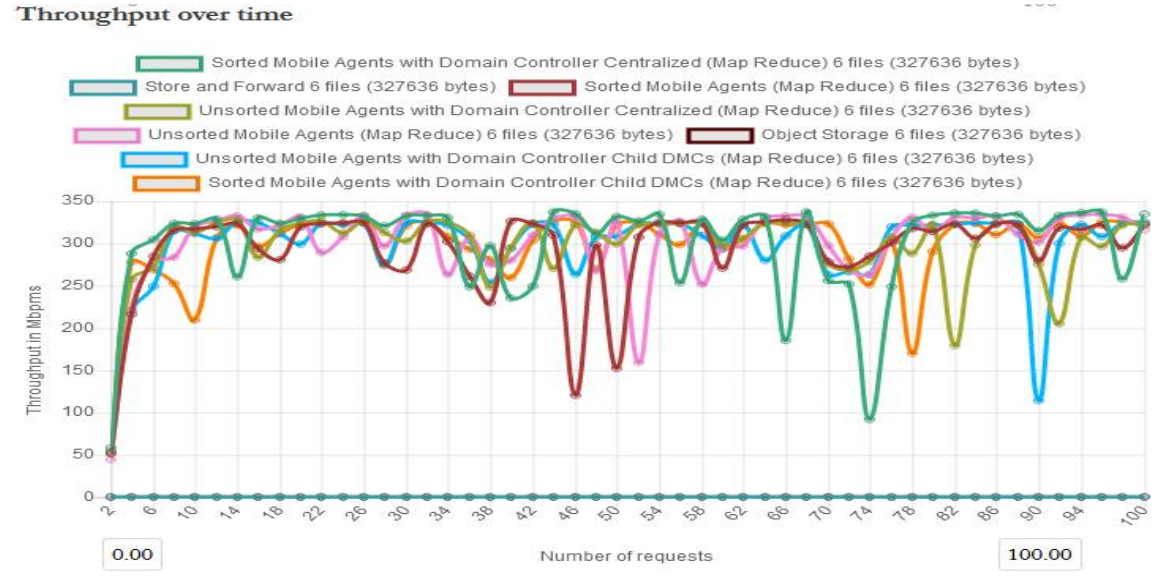

Fig 1.10: Throughput with time of 100 requests for OSD, SAF and Sorted, Unsorted Mobile Agents with Map-reduce distributed over three different SAN containers.

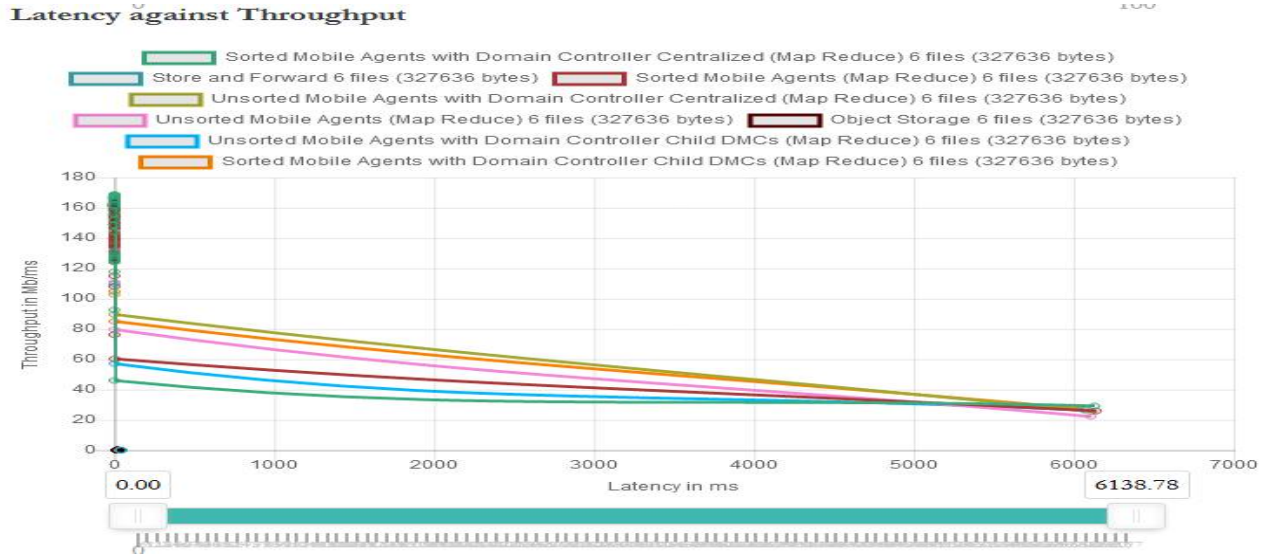

Fig 1.11: Effect of Latencies on Throughput of 100 client requests for OSD, SAF and Sorted, Unsorted Mobile Agents with Map-reduce distributed over three different SAN containers.

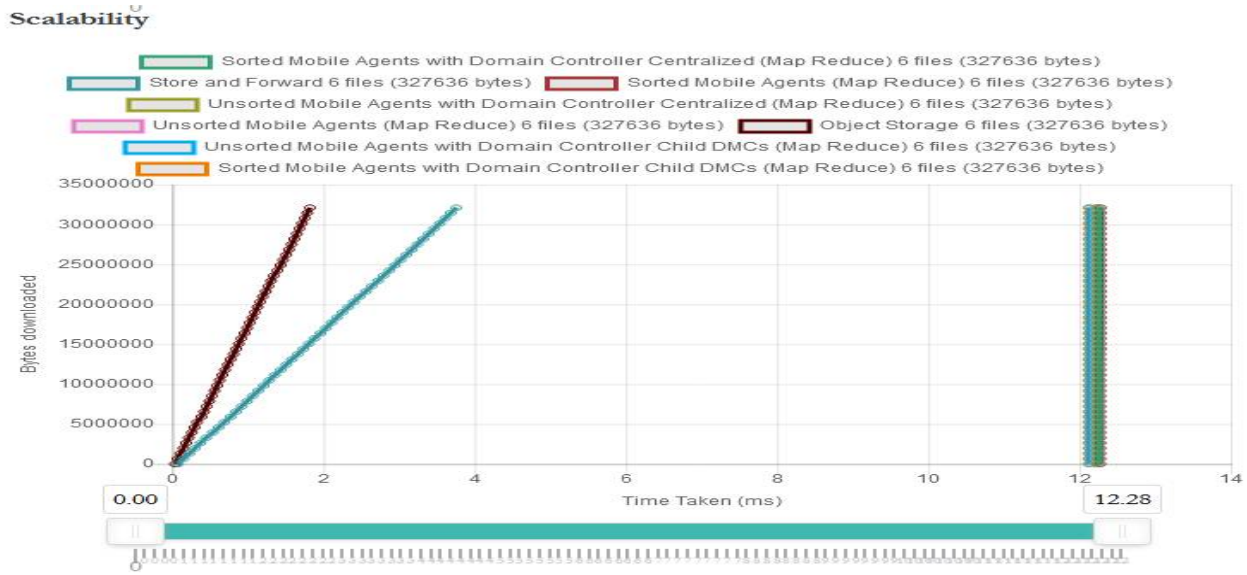

Fig 1.12: Scalability of 100 client requests for OSD, SAF and Sorted, Unsorted Mobile Agents with Map-reduce distributed over three different SAN containers.

5.1. OBSEVATIONS

This section sought to answer the research questions asked by the researcher:

i. Can Metadata be cached on a distributed network and to what extent can the catching affect or improve latencies, performance, throughput and scalability?

All the graphs in fig 1.9,fig 1.10,fig 1.11and fig 1.12 have been able to be run on cached metadata in the DMC and we can observe in fig 1.10 that indeed sorted agents with domain controller using map-reduce functions has the best throughput as the number of requests increase. For latencies object storage seems to be have large latencies at the beginning of downloading the files as indicted in fig 1.9 above, but the latencies decrease at the 6th request from the initial 1200ms to near 0 ms. but as for the Mobile agent based sorted and unsorted metadata the latencies are all kept at near 0ms. Interestingly fig 1.11 show that indeed that at 6000 ms mark, all the object devices operate at high latencies but agent based object devices seem to gain a lot of throughput over time and the best performing are the sorted and unsorted mobile agent devices enhanced with map-reduce. Fig 1.12 shows that object storage and store and forward processes exhibit an exponential growth with time, running at O (N^2), and for all the agent based sorted and unsorted meta-data they exhibit a constant slow growth with time running at O(n log n).

ii. Does splitting of a distributed system into smaller parts improve its speedup and/or performance and how relevant is it in the metadata management in such systems?

This question can best be answered in fig 1.10 and 1.11 where we can observe that indeed in fig 1.10 the best performing function is based on sorted mobile agents with domain controllers using map-reduce, followed by the unsorted mobile agents with a domain controller, it also worth to note that store and forward processes have the lowest throughput at maintained at 0 with increased requests, which is at a wide range followed by object storage which actually rival most of the mobile agent based metadata management schemes. Fig 1.11 shows sorting of metadata does indeed minimize latencies and consequently increasing throughput; which in effect improves performance of the system.

iii. What role can a mobile agent play in the management and transfer of meta-data in a distributed virtualized storage environment?

All the above experiments indicate that there is an improvement in scalability, latencies and throughput where the map-reduce and a mobile agent have been applied; this is indeed a clear testament that sorting metadata by splitting them, creating locality of references and localizing/caching metadata can improve on performance of a system.

iv. How can map-reduce functions assist meta-data management and in effect if used together with a mobile agent to what extent will they affect performance of the traditional Store and Forward (SAF) and Object Storage Devices (OSD).

As shown in Research question ii and iii. Above mobile agents tend to have a good performance when especially where map-reduce was used to sort the metadata, closely followed by the Unsorted metadata and object storage and store and finally forward processes are the worst performing.

VI. CONCLUSIONS

In conclusion from the above observations on the cross-section of the data generated by the SABSA Engine Simulator, the Mobile agents in a distributed network indeed improve on performance, more so in cases where localities were created using map-reduce functions in handling of big data which indeed justifies the assertion by (Pedro Jose, 2011) fig 1.2 above that mobile agents are the future of computing.

VII. FUTURE WORK

1) Extending the concept of Mobile Agent (MA) based virtualization to the field of Artificial Intelligence (AI) and Artificial Neural networks (ANNs); in order to solve the exponential data requirements in the IOT systems. ANNs will provide multiple distributed nodes that will communicate with other peers within a given domain and eventually transfer the processed data the distributed virtual child nodes or domains to the parent Nodes/Domains and eventually to the parent nodes/domain for final storage or processing.
2) To improve on our SABSA Test-Bed simulator to a more advanced simulator with an adaptable API-to allow for testing of applications-;including organizations for testing their storage requirements in regards to :scalability, Latencies and throughput.
3) To study and implement advanced security fencing systems within our agent-based storage architecture.
4) To use Kubernetes containers instead of the Docker Engine in the implementation of SABSA framework and test whether there may be any improvements in performance and storage

REFERENCES

1. Al-Shishitwy, A., 2012. Self-Management for large scale distributed system, Stockholm, Sweden: Royal institute of Technology.
2. Amazon, 2016. Amazon DynamoDB.[online]. Available: http://aws.amazon.com/dynamoDB/accessed 26/6/2016; 8:56 PM, s.l.: s.n.
3. Amazon, 2016. Amazon simple storage system, [online]. Available http://aws.amazon.com/s3/ © 2016, Amazon Web Services; Accessed 26/6/2016, 8:40 PM. s.l., s.n.
4. Amazon, n.d. Amazon DynamoDB.[online]. Available: http://aws.amazon.com/dynamoDB s.l.: s.n.
5. Amazon, n.d. Amazon simple storage system, [online]. Available http://aws.amazon.com/s3/. s.l., s.n.
6. Ana Avile's-Gonza'lez, J. P. g. F., July 2012. Scalable metadata management through OSD+devices. Murcia, Spain, Springer Science+Business Media LLC 2012.
7. Anon., 1998. Mobile objects and mobile agents:The future of Distributed computing. In: Promming and deploying Java mobile agents with aglets Addison wesley 1998 (ISBN:0-20-201-32582-9). Sunnyvale, Carlifonia: General Magic Inc.
8. Anon., 2016. Concordia White paper. https://www.cis.upenn.edu/bcpierce/629/papers/Concordia-Whitepaper/ [Accessed 173 2016].
9. Atul Mishra, A. S., 2012. Application of Mobile Agent in Distributed Network Management;YMCA University of science and technology, Conference on Communication systems and network. Haryana, India, IEEE.
10. B welch, M. U. z. A. G. g. B. J. S. a. B., 2008. Scalable Performance of the Panasas parallel file system-Usenex conference on file and storage technologies pp 17-33. s.l., FAST.
11. BigQuery, 2016. Big query.[online]. Available https://developers.google.com/bigquery/ s.l., s.n.
12. BigQuery, n.d. Big query.[online]. Available: https://developers.google.com/bigquery/. s.l., s.n.
13. C.R, K., 2004. Reseach Methodology:Methods and techniques. Daryaganj,New dheli: New age international publishers.
14. Cengiz Karakoyunlu, e., 2013. Towards a unified object storage foundation for scalable storage systems. Issue 978-1-4799-0898-1/13.
15. Chaturved, V., 2018. Deep Dive into Docker. Available at: https://www.edureka.co/blog/what-is-docker-container [Accessed 26 3 2019].
16. CORP, E., 2016. Content addressed storage systems, http://www.emc.com/products/systems/centera.jsp openfolder=platform ;accessed 26/6/2016, 11:05 PM., s.l.: EMC.
17. CORP, E., n.d. Content addressed storage systems, http://www.emc.com/products/systems/centera.jsp openfolder=platform, s.l.: EMC.
18. Docker, 2017. Docker Docs. [Online] https://docs.docker.com/v17.09/compose/install/ [Accessed 27 3 2019].
19. Docker, 2019. What is a Container.Available at: https://www.docker.com/resources/what-container [Accessed 3 March 2019].
20. EMC2, January 2008. Where information lives:current benefit and future potential technology concepts and business considerations., s.l.: EMC2.

21. Erik Riedel, m. k. &. R. s., 2002. "A framework for evaluating storage system security". s.l., Helwett Packard Laboratories.
22. Feng Wang, S. A. B. E. L. M. a. D. D., 2004. OSBF:A file system for object based storage devices. Adelphi MD, NASA Goddard/IEEE Conference on Mass storage systems and technologies.
23. Feng, January 2014. Distributed Data management using mapreduce. 3(46).
24. FullStack, n.d. Full Stack Python,Redis.https://www.fullstackpython.com/redis.html [Accessed 28 March 2019].
25.G.Mwathi, D., 2018. A model based approach for implimenting Authentication and access control in public WLANS:A CASE OF UNIVERSITIES IN KENYA , Nairobi: UON.
26. Garth A. Gibson, D. F. N. K. A. J. B. F. W. C. H. G. C., 1998. A cost effective high bandwidth storage archtectures. Arctectural support for programming languages and operating systems.
27. Griffith, T., 2018. Opensource.com;redhat. Available at: https://opensource.com/article/18/4/how-build-hello-redis-with-python [Accessed 28 March 2019].
28. Guilherme Soares, L. M. S., 1999. Optimizing Migration of Mobile agents. Coimbra, Portugal, MATA .
29. Hariri, S., 2001. "CATALINA":A smart application control and management". s.l., AMS.
30. Hitachi data systems, April 2016. Storage virtualisation:How to capitalize on its economic benefits. In: s.l.:s.n.
31. Hyperdox, 2016. A searcheable distributed Key-value store[online]Availabe: http://hyperdex.org/; Accessed 26/6/2016, 11:58 PM.. s.l., s.n.
32. Hyperdox, n.d. A searcheable distributed Key-value store[online]Availabe: http://hyperdex.org/.s.l., s.n.
33. James, 2006. Improving small file performance in object based storage. Issue CMU-PDL-06-104.
34.K Shvachko, H. K. S. R. a. R. C., 2010. The Hadoop distributed systems and technologies(MSST).[online] available: http://dx.doi.org/10.1109/MSST.2010.5496972. Washington DC, USA, IEEE Computer Society.
35. Kasireddy, P., 2016. FreecodeCamp:Beginner-Friendly Introduction to Containers, VMs and Docker. https://medium.freecodecamp.org/a-beginner-friendly-introduction-to-containers-vms-and-docker-79a9e3e119b[Accessed 26 March 2019].
36.L. George, H. b., 2011. The definitive guide.http://proquest.safaribooksonline.com/9781449314682. s.l., s.n.
37.librados, 2016. librados API documentation. http://ceph.com/docs/master/api/librados/ s.l.: s.n.
38.librados, n.d. librados API documentation.[Online] Availabe http://ceph.com/docs/master/api/librados/, s.l.: s.n.
39.LI, e. a. G., 2006. Researches on performance optimization of distributed intergrated system based on mobile Agent*. Dalian,China, IEEE.
40. Long, L.-F. C. a. D. E., 1990. Swifuret: A storage archtecture for large objects. Santa Cruz, Tech-Rep.
41. Lustre, 2002. A Scalable High performance distributed file system. http://www.lustre.org/docs/whitepaper.pdf. s.l., Cluster file systems Inc..
42. Lustre, 2016. http://www.lustre.org, s.l.: s.n.
43. Ma, e. J., 2006. Mobile agent enabled application mobility for pervasive computing.
44. Malik, A. L. a. p., 2008. "Cassandra: A decentralised structured storage system". s.l., Sigpos oper.Sys.
45. Malik, A. L. a. p., n.d. "Cassandra: A decentralised structured storage system". s.l., Sigpos oper.Sys.
46. Mark, e., 2000. Storage Virtualisation, What is it all about?. s.l., IBM International technical support.
47. Michael Factor, k. M. D. N. C. R. a. J. S., 2005. Object Storage:The future building block for storage systems. Sardinia, Italy, IEEE 2005.
48. Mike Meisner, G. R. G. a. E. R., August 2003. Object based storage. s.l., IEEE,Communications Magazine.
49. Mingers, 2001. Combining IS Research methods:Towards pluralist methodology. Information systems research 12(3):240-259. s.l., s.n.
50. Osero, 2010. Storage virtualisation and management- Masters Thesis, nairobi: University of Nairobi.
51. Osero, B., 2013. Network Storage Virtualisation and management.[online]www.ijern.com, Chuka, Kenya: International Journal of Education and Research, IJER.
52.P.H Carns, W. l. I. R. R. a. R. T., 2000. A parallel file system for linux clusters, in proceedings of 4th Annual linux showcase and conference. Antlanta, USENIX Association.
53. Panasas, n.d. http:www.panasas.com, s.l.: s.n.
54. Pedro Jose, M. S. A. A., 2011. The Emerging Domain of Co-operating Objects. 1 ed. Newyork: Springer Science.
55. Peng Gu, J. W. Y. Z. H. J. P. S., 2010. A novel weighted-Graph Based grouping Algorithm for metadata prefetching, University of Nebraska-Lincoln: CSE JOURNAL.
56. Permon, S., 2015. Parmabit INC, http://www.parmabit.com,© 2015 Permabit. accessed 26/6/2016, 10:30 PM., s.l.: s.n.
57. Permon, S., n.d. Parmabit INC, http://www.parmabit.com, s.l.: s.n.
58. Petre., P. M. a. M., 2004. The unwritten rules of PhD research-Open skills. http://postgrado.bio.uc.cl/wp-content/uploads/2014/11/Unwritten-Rules-of-PhD-Research.pdf [Accessed Thursday January 2019].

59. Pistirica Solin Andrei, A. v. H. M. A. N. C. M. M., 2014. 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). Bucharest, Romania, IEEE.
60. Prakash v. Rajguru, S. B. D., 2011. Analysis of mobile agents. Volume Volume 2.
61.R.Haskin, F. s. a., 2002. GPFS:"A shared disk file system for large computing clusters," in proceedings of 1st USENEX conference on file and storage technologies. Berkeley,CA USA, USENEX association.
62. Randy, e., 2011. San virtualisation evaluation guide, s.l.: Evaluator group.
63. Redis, 2016. [online]. Available: http://redis.I0/; last accesed 26/6/2016, 11:39 pm.. s.l., s.n.
64. Redis, n.d. [online]. Available: http://redis.10/. s.l., s.n.
65. Riedel, E., 1999. Active Disks Remote Execution for Network Attached Storage. s.l., s.n.
66.S.A weil, A. L. S. B. a. C. M., 2007. "RADOS: A fast Scalable, and Reliable storage services for petabyte-scle storage clusters.. s.l., Petascale Data Storage Workshop SC07.
67.S.A weil, A. L. S. B. a. C. M., n.d. "RADOS: A fast Scalable, and Reliable storage services for petabyte-scle storage clusters.. s.l., s.n.
68.S.A Weil, S. B. E. M. D. E. L. a. C. M., 2006. Ceph: A scalable high performance distributed file system, 7th symposium on operating systems design and implementation. s.l., OSDI.
69.S.Ghemawal, H. G. a. S.-T. L., 2003. "The Google File System," in proceedings of the nineteeth ACM symposium on operating systems principles.http://www.cs.rochester.edu/sosp2003/papers/p125-ghemawat.pdf. s.l., ACM press.
70. Satoh, I., 2014. Map-reduce based data processing on IOT-National institute of informatics. Tokyo Japan, IEEE.
71. Saulsbury, A., 1999. Attacking Latency bottlenecks in distributed shared memory, s.l.: s.n.
72. Sergent, R. g., 2011. Verification and Validation of Simulation Models. Syracuse, NY 13244, U.S.A, IEEE.
73. Shein Bin, L. Y. X., 2010. Research on data mining models for the internet of things. Volume 3.
74. Simsarian, K., 2000. Toward Human Robot Collaboration , s.l.: s.n.
75. Sowmya, N. e. a., 2015. An adaptive Load balancing strategy in cloud computing based on Map-Reduce. Dehradum,India, NCGT.
76. Srinivas Singavarapu, S. H. M. Y., 2001. Self Managing Storage System-Design and evaluation, http://acl.ece.arizona.edu/projects/old/smp/smppaper.pdf,downloaded 25/3/2016; 1:30 PM.. Arizona, s.n.
77. Srinivas Singavarapu, S. H. M. Y., n.d. Self Managing Storage System-Design and evaluation, http://acl.ece.arizona.edu/projects/old/smp/smppaper.pdf,downloaded 25/3/2016; 1:30 PM.. Arizona, s.n.
78. Steven McCanne, m. V. a. V. J., 1997. "low complexity video coding for receiver driven layered multicast". s.l., IEEE journal on selected areas in communications, vol 16.
79. TutorialPoint, n.d. A quick guide to Redis. Available at: https://www.tutorialspoint.com/redis/pdf/redis_quick_guide.pdf [Accessed 28 March 2019].
80. Walsh, D. J., 2014. OpenSource.com:Bringing new Security features to Docker. https://opensource.com/business/14/9/security-for-docker[Accessed 26 3 2019].
81.weber, R. O., July 2004. Information technology-SCSI object-Based storage device commands (OSD). s.l., Ralph O. Weber.
82. Wolfgang Lehner, K.-U. S., 2013. Web Scale data management for the Cloud. 1 ed. New York: Springer.
83. Xu Liancheng, X. J., 2014. Research on distributed data stream mining in internet of things, International conference on logistics Engineering management and computer science. Jinan, China, Antlantis Press.

