International Journal on Emerging Technologies **11**(3): 1147-1156(2020)

ISSN No. (Print): 0975-8364
ISSN No. (Online): 2249-3255

A Virtualized Mobile Agents Based IoT Model with Map-Reduce: Using the Python's Spade Framework

Benard O. Osero[1], Elisha Abade[2] and Stephen Mburu[2]

[1]Lecturer, Department of Computer Science, Chuka, Kenya.
[2]Senior Lecturer, School of Computing and Informatics, Nairobi, Kenya.

(Corresponding author: Benard O. Osero)
(Received 25 March 2020, Revised 20 May 2020, Accepted 23 May 2020)
(Published by Research Trend, Website: www.researchtrend.net)

ABSTRACT: 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. The major challenge with most of the current storage systems is that the don't scale well and also exprience high latencies across the network which may also increase the security risk of the files being migrated. This paper therefore seeks develop applications that can follow mobile users when they change to a different environment, especially with the change of device and location by use of mobile agents. Implementation of application mobility also depends on context-awareness and self-adaptation techniques. This paper has delved into the concept of Virtualization and thus unearthing the deficiencies that impede performance of distibuted Network (Latencies, Scalabity, and throughput) in centralized array based metadata blocks that either employ physical file storage or virtualized file storage including the store and forward (SAF) file systems and Object Storage Devices (OSD) systems and finally propose an IOT archtecture to be able to solve the identified problems.

Keywords: Autonomic computing, distributed computing network storage, Multi-Agent Platforms, Mobile agent, metadata, object storage, Network attached disk.

Abbreviations: SAN, storage area network; NASD, network attached secure disks; SAF, store and forward; OSD, object storage device; DAS, direct attached storage; MAP, multiagent platform.

I. INTRODUCTION

Today, Big Data, IoT and Analytics are driving and making the differences in key performing top organizations. The interplay of these three areas can be instrumental for the future development of research, complex systems and enterprises. IoT will be estimated to rise to billions of devices connected by 2020 [50].

Virtualization is a critical determinant which defines the path distributed storage array and consequently IoT systems should follow in order to succeed. By its nature virtualization manages data right from its source. The storage value has been is changing from drives to the array cluster controllers while enhanced and data protection policies are included in such systems with time [28]. Applications of pervasive nature that can change with the change of environment have been on the increase. To Implement these applications context awareness and self-adaptation techniques should be considered [37].

Storage Area Networks (SAN) is a technology that is no longer being experimented upon but rather mature enough to influence the future of research and technological advancements in the industry. Implementing the SANs alone does not necessarily guarantee solutions of the most basic issues in satisfying the needs of the increased data demands [18]. More than two decades later similar trends in the storage requirements have been kept on the increase thus calling for innovative methods to handle such big data requirements in both data and its processing. The major drawbacks with the current systems such as store and forward (SAF) and Object storage devices (OSD) is that they do not scale very well due to the tight coupling of metadata and associated data, this problem is further exacerbated by the fact that physical files or metadata resources have to be available only in the server or within the Storage Area Network (SAN) at the time of request and thus leading to increased latencies during request of these files. The fact that these files are also transmitted across the network leads to high bandwidth requirements.

This Research paper seeks to explore through literature survey the strengths and weaknesses of the traditional methods of storage in the client-server environments like Store and Forward, Network attached secure disks (NASD) and Object Storage Disks (OSD) and other emerging trends in big data management like map reduce model and other distributed systems approaches, then identify suitable gaps and then propose a suitable Virtualized IOT architecture, whose major strength lies in the use of mobile agents and map-reduce to sort and eventually cache metadata and thus minimizing latencies and increasing throughput. The research will in future blend the NoSQL databases with SPADE to eventually develop an IoT simulator that will eventually be used for performance improvement testing and eventually the actual system implimentation.

This research paper is organized into eight sections; Section I is the introduction, Section II shows the current

storage models, Section III delves into the client-server architecture which forms the basis of all distributed networks and also the various approaches for the client-server implementation such as Distributed and Autonomic Computing approaches, Section IV explains more about Mobile Agents, Map-reduce and the relationship between them, Section V compares various Mobile Agent Platforms (MAPs) in order to identify the best platform for the Mobile Agents on a distributed network, Section VI explains 'SPADE' a mobile agents implementation Framework as identified in Section 5, Section VII gives justification for Map-reduce in a distributed Client-Server environment, section VIII shows the identified gaps in the Literature and the proposed virtualized IOT storage architecture and Section IX is the Conclusion. This research paper majorly dedicates its effort towards identifying suitable algorithms and models required for implementation of a suitable architectural model of distributed storage on a network in order to improve on performance of the previously identified distributed data storage systems and eventually propose a secure IOT implementation owing to the growing demands for unstructured data.

II. CURRENT WORK IN THE CLIENT-SERVER APPLICATIONS

A. Direct Attached Storage(DAS)

[47] referred this type of storage as store and forward (SAF), in which the network disks involved have to keep a copy of another redundant disk in the server. Every time a client requests for a file a copy of the file has to be kept before it is forwarded to the client for downloading [22] further compared Direct Attached Storage (DAS) and Network-Attached Storage Devices (NASD) and demonstrated that by keeping a copy of the disk there was a penalty on performance and scalability he also demonstrated an improved security mechanism by using tokenization on these platforms and concluded that such systems can be improved by use of Object storage management schemes and suggested further work to be carried out on mobile agent and mobile-code migration on a distributed network.

B. Network-Attached Storage Devices (NASD)

Network-attached storage (NAS) happens at the file-level where one or more dedicated servers and disks store data and share it with other clients on a network.

Network-Attached Secure Disks (NASD), is a Networked object based shared storage system discussed by [3] in their classification taxonomy, that modifies the interface for the common direct attached storage devices and thus eliminating the server resources required for the movement of data. Fig. 12 outlines the major components of NASD ARCHTECTURE [12, 22].

C. Object-Based Storage (OSD)

For Object storage the data is broken into small connected units called objects kept in a single repository (pool), instead of being kept as blocks on servers.

Most of the modern cluster file systems consists of many Object Storage Devices (OSD) in information storage consequently, attaining high performance. Because of the unavailability of OSD-based disks in the market, they are implemented by exporting an OSD-based interface, and which locally utilizes regular file system for objects storage. The FPFS metadata cluster implements OSD+ devices in provision of a scalable metadata and high performing system and service [5].

III. ISSUES AFFECTING CLIENT-SERVER APPLICATIONS

DAS are Array based block storage which use a Sequential contiguous access and thus experience more execution time $O(N^2)$ and the need physical file for execution to occur. NASD are Array Object based which use Sequential Access and metadata prefetching with an execution time of O (log n) due to the fact that file is not cached there are high latencies involved in order to get the file downloaded, physical disk needs to be available [3, 12, 22, 23]. OSD on the other hand are object based which use Metadata prefetching, caching and Chunking. OSD has good performance, uses divide and conquer to handle the files whose worst case run time is O (n log n) leading to the Flattening of metadata, although it also does not handle network bandwidth well. OSDs purely use metadata objects to access storage [2, 30], Hyper-dex [10, 19, 11] lustre [21, 24, 26, 8, 5, 14] suggests metadata prefetching and flattening as a solution to the OSD problems this method has the following benefits:

— The Object ID is a locality hint, with closeness indicating relationships, for internal layout and cache management policies.

— Index structures for object metadata may be arranged as tables to offer better locality.

— A set of related files may be identified by a unified object ID range instead of an enumerated list.

Although metadata prefetching solves the issues of latencies it does not provide a solution to the high latencies experienced through metadata requests and transmission. To better solve these problems [15, 2, 20], Rados Gateway [7, 35, 16] have used agent based solutions to improve the problems encountered during the client-server interractions. Self-managed systems use two approaches, Distributed approach and Autonomic Computing approaches:

A. Distributed approach

In this architecture the number of client storage interactions that must be relayed through the file manager is reduced and thus leading file manager's reduced work. Since one channel can be used to transmit the data effectively this reduces bandwidth.

B. Autonomic Computing approaches

Alberola (2010) describes Autonomic computing as the self-managing characteristics of distributed computing resources, adapting to unpredictable changes while hiding intrinsic complexity to operators and users [1]. The popular approaches that have previously been applied under this category are:

—Architectural Approach*:* This approach promotes the assembly of components, autonomic in nature, thus influencing the actions taken by the system in response to changes in the user behavior or environment.

—Control Theoretic Approach: This approach has classically been used in solving control problems in computer systems like throughput regulation, power management and load balancing. It further uses

prediction control mechanism to solve the network delays and unpredictable operating environments to solve challenges beyond classical control.

—Emergence-based Approach: This approach emerges simple behaviors of systems at the micro level with asynchronous exchange of data between subsystems.

— Agent-based Approach: This approach is suited for distributed systems with many complex interactions; forming large societies or organizations used to handle distributed problems [33].

— Legacy Systems: This approach focuses on defining the requirements for a system to be adaptable to existing platforms and therefore it can easily be controlled.

Our research focused on the Agent based approach in solving the client-server issues, together with Map reduce for metadata sorting in order to create sorted metadata domains.

IV. MOBILE-AGENT BASED DISTRIBUTED NETWORKS

Distributed Mobile Agents use in distributed networks are an emerging technology that is gaining momentum in the field of distributed and autonomic computing to develop applications for mobile, pervasive, and 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. A common problem when one wants to benefit from mobile agent technology to develop distributed applications is the decision about which platform to use fortunately in the latest years several commercial implementations of mobile agent systems have been presented in the market, in which for the clients to be easily identfied domain metadata needs to be first sortedinto specific IP cluster domains and eventually mapped to the clients via the Domain controller (DMC).

A. Reasons for Mobile Agents

The study of mobile agents has a wide range of applications in our day to day applications which is not only promoted by technological advancements demands but also how they can be of help in creating distributed network environments. The following are the motivating factors for the use of mobile agents [4]:

— Load reduction. In Fig. 1. Computations are taken or moved to the data and not the data being taken or moved to the computations.

— They overcome network latency, can be localized and directly executed via the controller.

— They are used in encapsulation of protocols; Mobile agents, are able to migrate to remote hosts in order to establish a channels of communication based on proprietary protocols.

— The agent execution is asynchronous and autonomous in nature. They can act on their own without undue influence of the process that created them as shown.

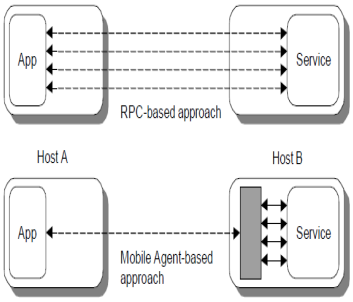

Fig. 1. Mobile Agents Reduce Network Load [4].

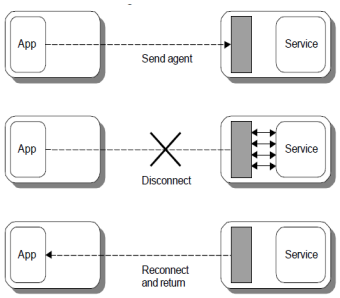

Fig. 2. Disconnected Operation with Mobile Agents.

— They are able to adapt dynamically. They can optimally be distributed over the network to other nodes to solve particular problems.

— They are heterogeneous in nature.

— They are fault-tolerant and robust. Since they provide a centralized code server, from where all the other code is stored. Although the server code ensures a solid structure it does not necessarily permit an efficient migration of the agents through the platform. A single centralized point of request can easily become a point of failure in the system. Fetching the classes from a single node can equally be a very inefficient approach and does not meet the needs for high-performance computing. One major characteristic of the mobile agent systems is capability of flexibly and efficiently transporting both the data and code to where they are mostly needed.

B. Mobile agent-based Map Reduce system

Map Reduce is a computing platform with certain kinds of distributable problems using a cluster consisting of a large number of computers, the original map-reduce consists of three phases: Map-phase, worker phase and Reduce phase. New models using agents have been suggested where the Mapper agent is a container agent corresponding to the master node in the Map Reduce pattern, it supports multicast coordination and contains at least one worker agent inside it.

Fig. 3, developed by Satoh (2014) describes a platform for dynamically organizing multiple mobile agents for computing, it depicts a comprehensive model for map-reduce platform employing mobile agents, advancing on the model previously proposed by [31], also previously demonstrated that agent size has a direct implication on cost.

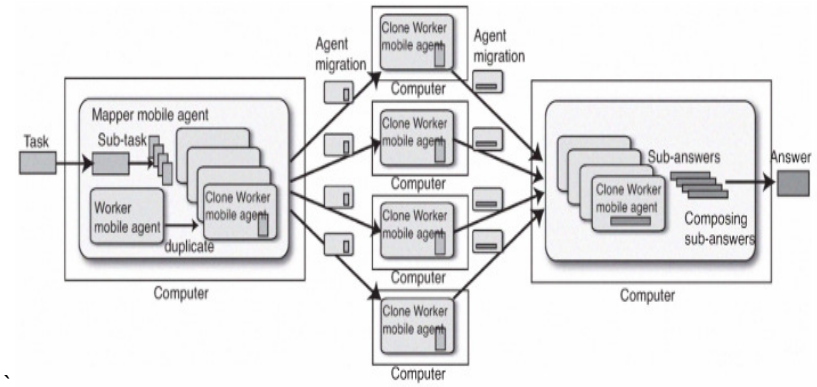

Fig. 3. Mobile agent-based Map-Reduce system.

V. A PERFORMANCE EVALUATION OF MULTI-AGENT PLATFORMS

In order to implement the mobile agent system in distributed network a suitable model has to be sought to cost effectively design and test the system. Multi-agent Platforms (MAPs) provide tools to help in the improvement, development and implementation of Multi Agent Systems (MASs). Following the large number of existing MAPs in the market, choosing a suitable tool to develop a MAS becomes a challenging for MAS developers, Alberola, (2010) [1] gives an analysis of the current MAP deficiencies through an in-depth research where he analyzed the current MAPs and the features they provide as shown in the Table 1 below. These results have been a key driver in the choice of multi agent platforms.

According to Alberola, (2010), there are many existing criteria for selecting MAP like; the tutorials, degree of support, availability of different operating systems, documentation support for development, discussion forums, and etc. According to their study platforms that are FIPA compliant and Open Source OS are the most preferred: Java based Zeus and Python based Spade are the best satisfying all the identified MAP selection criteria.

Table 1: Features Provided by Various Multi-agent Platforms [2].

Platform	Language	O.S.	FIPA	Sec	Org.	Comm
3APL	Java	✓	✓	-	-	RMI
AAP	April	✓	✓	-	-	ICM
ABLE	Java		✓		-	RMI
ADK	Java	-	-	✓	-	
AgentDock	Java	-	✓			RMI
AgentScape	Java/Python	✓	-	-	-	RMI
Aglets	Java	✓	-	✓	-	RMI
Ajanta	Java	✓	-	✓		RMI
Ara		✓	-	✓	-	RMI
CAPA	Java		✓	-	-	
CapNet	C#	-	✓	✓	-	Several
Concordia	Java	-	-	✓	-	RMI
Cougaur	Java	✓	-	✓	-	RMI/Corba/http
CrossBow	Java				-	Proxy
Cybele	Java	✓	-		-	
Dagents		✓	-	✓	-	RPC
Decaf	Java	✓	-		-	
Genie	Java	✓	✓	-	-	RMI
Grasshopper	Java	-	✓	✓	-	RMI
Cypsy	Java	✓	-	✓	-	RMI
Hive	Java	✓	-	✓	-	RMI
Jade	Java	✓	✓	-	-	RMI/Corba/http
Jack	Java	-	-	-	✓	tcp/ip
Jackal	Java		-	-	✓	tcp/ip
Jason	Java	✓	✓		-	Jade
Mage	Java	✓	✓		-	RMI
Madkit	Java	✓	-	-	✓	Sockets
Sage	Java	✓	✓	✓	-	RMI
Samoa	Java	✓	-	✓	-	RMI
Soma	Java	✓	-	✓	-	Corba
Spade	Python	✓	✓	✓	✓	Jabber
Spyse	Python	✓	✓		-	
Voyager	Java	-	-	✓	-	RMI/Corba
Zeus	Java	✓	✓	✓	✓	

VI. SPADE AGENT FRAMEWORK FOR IOT IMPLEMENTATION

SPADE platform research has been carried out by [1] on Multi Agent Platforms (MAPs) where he Identified the strengths of various platforms, the strength of SPADE lay in the use of FIPA and open source and it has also inbuilt enabled security features. The other strengths of SPADE are described in [9, 47].
The model is composed of a connection, a message dispatcher, and a set of different behaviors. Every agent requires Jabber ID (JID) and a password for the establishment of a connection with the XMPP server. The JID is composed of a username, an @, and a server domain which will be the name that identifies an agent in the platform, e.g.myagent@myhomeprovider.com. This model will be suitable in the implementation of the actual mobile agent model.

A. Common Spade Agent Services
[65] describes some of the common agent service as a component to provision user machines or application servers across the whole enterprise an architecture that has been advanced in the design of our IOT based model. These services provide remote deployment capabilities, secure connectivity, and shared machine resources. The Services they offer include the subcomponents shown in Fig. 4.

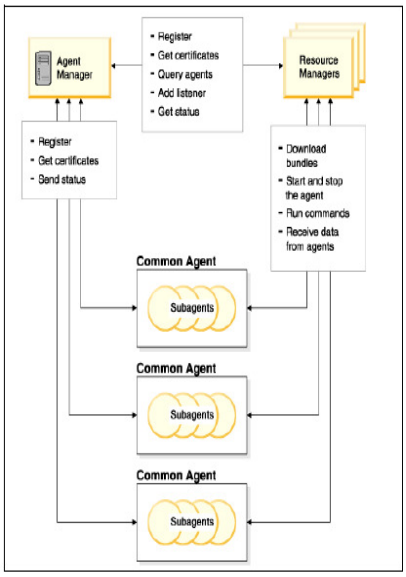

Fig. 4. Common Agent Services [33].

1. Agent manager: It ensures a secure connection between managed endpoints and maintaining the database information.
2. Common agent: It *acts* as common container for all of the subagents to share resources during the management of a system.
3. Resource Manager: Together with the subagents they are used for software distribution and software inventory scanning.

B. Spade with FIPA Framework
The Spade agent model conforms to the standard FIPA specifications which makes it compatible with other network and software platforms [37] described the FIPA framework as consisting of the following key components as illustrated in the FIPA reference model below.

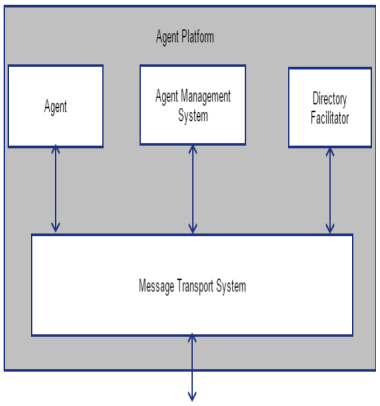

Fig. 5. FIPA archtecture.

FIPA Agent Communication procedures define the Agent Communication Language (ACL) messages which describe how; message exchange interaction protocols, speech act theory-based communicative acts and content language representations.
The FIPA ACL contains message specific communications controls within the FIPA framework. The objectives of standardizing the FIPA ACL message provide:
— Compatibility through a standard set of ACL message structure.
— Well-defined process to maintain the defined set.

C. Spade Agent messaging relationships
An agent is thought of having the following relationships as defined by [36] depicted as a UML in Fig. 6.

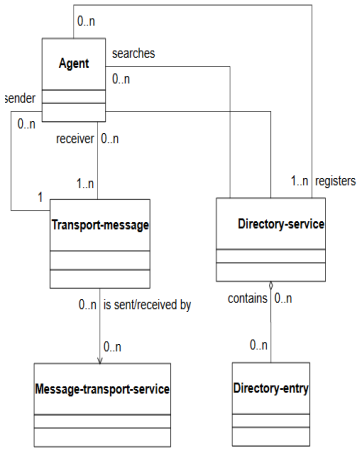

Fig. 6. UML- Basic agent relationships.

Xu *et al.,* [36] further describes the transport relationship from one agent to another, with Transport-message being the conveyed object from one agent to another. It further contains the transport-descriptor for the sender of the message and receiver(s), along with a payload containing the message, as shown in Fig. 7.

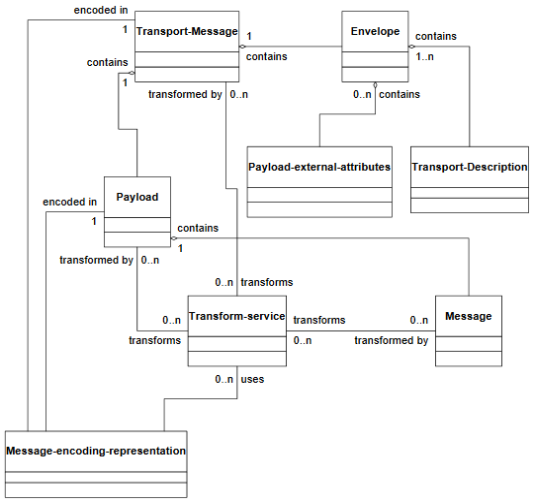

Fig. 7. UML Transport relationships.

The relationships described in the above models are an important precursor in the design and implementation of an effective IOT design.

VII. THE NEED FOR MAP REDUCE IN A DISTRIBUTED NETWORK

It is estimated that there will be approximately 125 billion Internet of Things (IoT) devices connected to the Internet by 2030, which are expected to generate large amounts of data. This will challenge data processing capability, infrastructure scalability and privacy [49].

The map-reduce model uses the key-value pairs where the records with similar key (i.e. word) are grouped together where finally they are fed into the reducing function which then aggregates the input values and generates an output of the aggregate occurrences in a given document(s) [25], therefore, map-reduce is a simple and efficient for computing aggregate. The idea of map-reduce is not different from "filter and then group aggregation" and its major advantages are as follows [17]:

— Simple and easy to use: Only define the work by use of map and reduce functions.

— Flexibility Map-Reduce: There is loose coupling between data model and schema; irregular or unstructured data can easily be handled than it is with the DBMS.

— Independence storage: Map-reduce does not rely on the underlying data models layers but are compatible with different storage layers like the Big Table and others.

— Fault-tolerant: Map reduce can continue to work even if some failures have been encountered in the system.

— Highly scalable: Map-Reduce is advantageous because of its high scalability. Yahoo has in its report that the Hadoop gear was able to scale out 4,000 nodes.

Performed An experiment showed that map-reduce scales better on both increased nodes and data [25].

Dataset: File Size used = 100 Mb

Experiment for increasing size of dataset and nodes

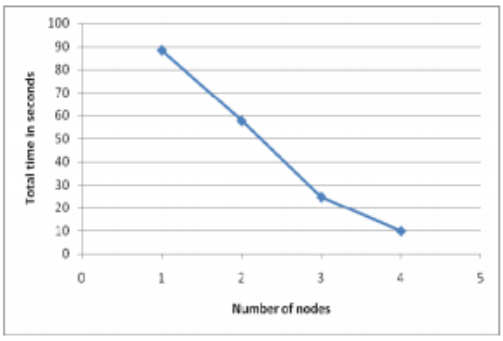

Fig. 8. Execution time versus the number of nodes [25].

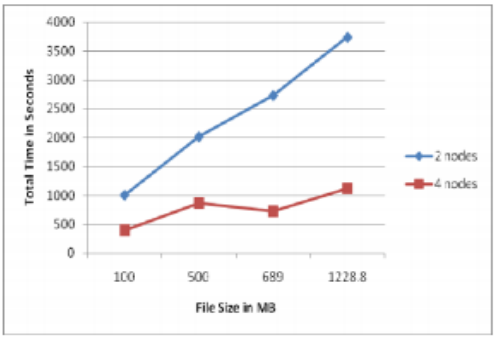

Fig. 9. Execution time, varying dataset and Nodes [25].

Fig. 8 above indicates as that the number of nodes increases the time of execution decreases in the Hadoop cluster. It is also clear in Fig. 9 that as the number of node increase from 2 to 4 the time taken to download the same amount of data decreases significantly; a clear indication of how scalable map reduce is with increase in load capacity. In applying Map-reduce Model to the cloud [45], Fig. 10 and 11 compares the response time and throughput of Map-reduce and Round-Robin scheduling algorithm and he observed that there was an improvement of Map Reduce as compared to Round Robin in regards to Response time and Throughput as shown below:

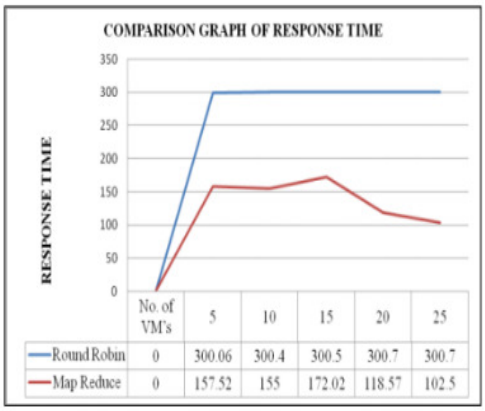

Fig. 10. Reponse time of Map Reduce over Round Robin [45].

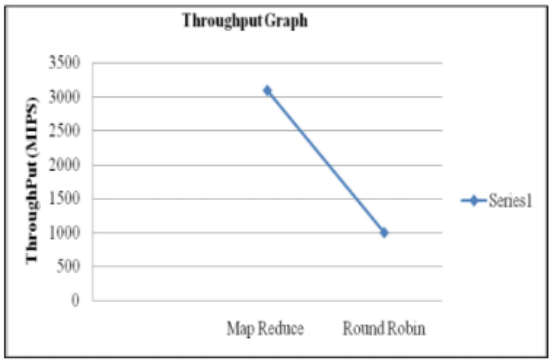

Fig. 11. Shows improvement of Map-Reduce over Round Robin in terms of throughput [45].

A. Performance Comparison of Open MP, MPI, and Map Reduce in Practical Problems

Open MP: Open MP is an (API) that facilitates easy development of parallel programs in shared memory where threads can share the same memory address space and thereby enabling communication between threads to be very efficient [38].

MPI: Is a message passing library on a distributed computing environment. Programmers take charge of partitioning workload and mapping tasks about which tasks are to be computed by each process. Table 2 and 3 below shows practical problems execution time [38]. Execution Times for the all-pairs-shortest path problem.

Table 2: Execution Times for the all-pairs-shortest path problem.

Node size	Framework			
	MapReduce	Cluster	Single Machine MPI	Open MP
10	2m 26s	0.32 s	0.34 s	0.1 s
100	16m 52s	0.44 s	0.41 s	0.25 s
1000	4h 4m 39 s	4m 48 s	24.14 s	8.03 s

Table 2 above compares three parallel system execution frameworks the results above indicate poor performance of map-reduce on computational-intensive and iterative computation problem.

There are improved response times as the number of Nodes increase.

Table 3: The execution time for the join problem.

Problem	Framework		
	MapReduce	MPI	Open MP
The Join problem	24m 15s	135h34m	93h14m

Table 3 shows that the execution time varies depending on the execution context like network bandwidth and resource management for operating systems. The same experiments have been conducted three times for each setting. The MapReduce-based program was the best one among the three models for data-intensive processing of big volume of data [38]. Therefore this justifies the need for Mapreduce in dealing with distributed unstructured data/metadata sets.

VIII. GAPS IDENTIFIED IN THE RELATED WORK

There have been a number of gaps that have been observed in the previous researches some of which this research aims to accomplish including the following:

— There has been an increase in the data processing demands and this requires faster systems that can scale well over short periods of time, most of the systems so far covered in the literature are either traditional in nature like SAF (DAS) systems which are array based or they are object based like NASD and OSD but they experience *high latencies and decreased throughput* as witnessed in [10, 11, 13, 19, 22, 23].

— In all the models so far discussed in the literature none has attempted to compare all the other previous models: SAF, NASD, OSD and their associated metadata with the agent based and map reduce platform in order to establish the weaknesses or strengths of these models in terms of performance and scalability.

— So far there has been no specific model or framework for agent development that has been implemented on the existing mobile agent frameworks with map reduce; although [31, 32, 39-42] only provide models for agent development and no particular framework or architectural model for agents has been implemented so far in these systems.

— Low bandwidth and unmanaged Latencies have a major effect on the performance a distributed cluster or network, data prefetching methods have been implemented through predictive prefetching algorithms, but little progress has so far been made on metadata management schemes [48]; Although, [5] provide a solution to bandwidth issues which occur when the client and server interact, this solution only improves on bandwidth and not latencies.

— So far there has been little effort to consider the effects of creating locality of reference through *metadata sorting and subsequently identifying its effects on performance* improvement on a distributed cluster. In their solution [10,14] extensively used unsorted blocks of metadata for mapping the metadata to the client, which is not efficient.

— There is no major distributed network platform that has so far been developed for testing the performance of a distributed network more specifically using mobile agents together with the map reduce algorithm that has been independently been used for big data and data mining solutions [46] and the Internet of things IOT [32]. There have been major attempts to use Mobile agents on a distributed network platform together with map reduce by [31, 32, 39-43] but they didn't explore the performance implication of both the agents and map reduce.

— So far there is no Custom made Distributed storage performance testing Simulator developed to address file level performance testing although, most of the simulators existing are only meant for the lower layers of the OSI model (Physical, Data link and Network layers) as witnessed in [43, 44]. This makes it very hard to test file performance and scalability on this physical networks and custom made simulation tools.

A. The General Virtualized Secure Agent Based IOT 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 proposed to address the issues of latencies and throughput on big data requests on Internet of Things.

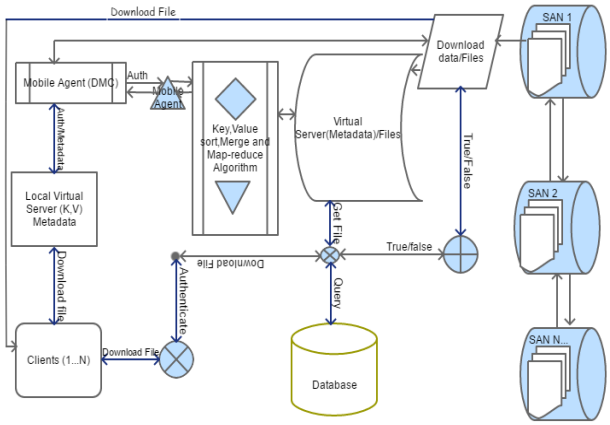

Fig. 12. A Conceptual Architecture for intelligent objects using agents and Map-reduce.

Fig. 12 above shows an architectural model of the agent based design using map-reduce it is a three tiered model with the client as the front end the virtual serve as the middle tier and storage SAN as the backend, the functionalities of this model are as follows:

1. Storage Area network (SAN)-It is responsible for the storage of the physical files it is implemented as a storage container that has a global IP address to identify the container; included is also the port number and individual internal IP address to identify each internal individual container.
2. Virtual Server(VS)-It contains the logical implementation of the switching of networks to enable the clients access the metadata. It is also responsible for the authentication of the clients by providing a tokenization mechanism whose capabilities are stored in the database and later mapped onto the storage to allow clients download files.
3. Client-It is an important aspect of this distributed architecture. It is responsible for requesting for the files and then allowing the clients to view the files through the console or preferred browser interface.
4. Map-reduce Functions-Responsible for sorting and reducing metadata functions which can then be transported to client side for further processing.
5. Mobile Agent-It is responsible for migrating sorted metadata values from the virtual resource server to the client side.
6. Local Client-Functions hand in hand with the domain controller, which manages the local switching of clients and keeps a registry of the requested and served metadata requests for each client, it also caches the requests for future access.

The model uses a search mechanism to the existing metadata resource storage pool enhanced by the map reduce algorithm that sorts the metadata blocks according to the client IP address domains before mapping them to a mobile agent and eventually migrated to a Domain Controller (DMC).

The mobile agent then fetches the sorted metadata (using map-reduce function) pool and migrates them to one of the selected local servers where they are executed henceforth, this would be terminated if this particular local server terminates normally or it is terminated by the parent server in case the local server used the resources that were not allocated to it or issued instructions beyond its allocated mandate or a critical unrecoverable event happened.

The clients within a particular domain are then given the resource paths 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 has the potential of enforcing their local security mechanisms to be able to protect the clients within a particular domain.

IX. CONCLUSION

We have explored using literature survey the strengths and weaknesses of client-server environments and also the strengths of map reduce model, we were able to identify distributed systems approaches such as Distributed approach and Autonomic Computing approaches.

As can be seen in Fig. 9-11 and also Table 2 and 3 it is clear that map reduce has the capability of improving both throughput and system scalability and therefore justifying the reason why it has been earmarked to be used our IoT architecture design.

This research is also based on the autonomic approach by use of mobile agents which also combines the aspect of map reduce to come up with the hybrid architectural design aimed at eventually improving performance of a distributed network through delocalization of metadata by creating independent and secured metadata clusters through caching at the domain controller(DMC).Unlike the SAF and OSD models, the IOT model in this paper will have the capability of creating independent clusters which will be cached in the DMC with their identified domains and therefore minimizing the fetch execute cycles which will not only improve the security of the system but also minimize latencies and increase throughput.

Having identified subtle gaps with other existing storage models this research was able to make a contribution in formulating an IOT architectural design that will eventually be implemented as a simulator for performance improvement testing of the traditional methods (SAF, NASD and OSD) of storage as compared to mobile agent based OSDs in a distributed storage network.

X. FUTURE SCOPE

Since IOT is an emerging trend, our research will focus in Implementation and testing, using the resulting simulator, of the above IOT architecture in the Python environment and then testing of the performance improvement in the Mobile agent based OSDs as compared to the SAF and OSD and eventually the actual IoT system implementation.

ACKNOWLEDGEMENTS

The authors would like to thank the School of Computing and informatics, University of Nairobi, and Chuka University Department of Computer Science for ALL the resources and the Conducive environment they have provided to enable us to carry out our research.

Conflict of Interest. This research paper has no conflict of interest since no similar research is being carried out nor being funded by any other institution.

REFERENCES

[1]. Alberola, J. M. (2010). A performance evaluation of three Multiagent Platforms. *Artificial Intelligence Review*, *34*(2), 145–176.

[2]. Amazon (2019). *10-Minute Tutorials*. Available at: https://aws.amazon.com/getting-started/tutorials/.

[3]. Andrei, P. S. (2014). Evolution towards Distributed Storage in a Nutshell,1267–1274.

[4]. Anon (2016). *Concordia White paper.* Available at: https://www.cis.upenn.edu/bcpierce/629/papers/Concordia-Whitepaper/ (Accessed: 17 March 2016).

[5]. Avilés-González, A., Piernas, J. and González-Férez, P. (2014). 'Scalable metadata management through OSD+ devices. *International Journal of Parallel Programming*, *42*(1), 4–29.

[6]. Caidi, M. (2008). 'The Google File System Sanjay', *Journal de Chirurgie*, 145(3), 98–299. doi: 10.1016/S0021-7697(08)73776-1.

[7]. Ceph (2016). *Welcome to Ceph*. Available at: http://docs.ceph.com/docs/master/# (Accessed: 30 April 2019).

[8]. CORP (2016). *Content addressed storage systems*, *EMC*. Available at: http:www.emc.com/products/systems/centera.jsp?openfolder=platform (Accessed: 26 June 2016).

[9]. Escriv, M., C, J. P. and Bada, G. A. (2014). 'A Jabber-based Multi-Agent System Platform ∗', (January 2006). doi: 10.1145/1160633.1160866.

[10]. Factor, M. (2006). 'Object Storage: The Future Building Block for Storage Systems A Position Paper', 119–123. doi: 10.1109/lgdi.2005.1612479.

[11]. Feng, D. (2004). 'Enlarge Bandwidth of Multimedia Server with Network Attached Storage System 3 The Redirection of Data Transfer', 489–492.

[12]. Gibson, G. A. (2001). A cost-effective, high-bandwidth storage architecture', *High Performance Mass Storage and Parallel I/O: Technologies and Applications*, (2014). 431–444. doi: 10.1109/9780470544839.ch28.

[13]. Hendricks, J. (2006). 'Improving small file performance in object-based storage', (May).

[14]. James (2006). 'Improving small file performance in object based storage.', *CMU-PDL-06-104*.

[15]. Karakoyunlu, C. (2013). 'Toward a Unified Object Storage Foundation for Scalable Storage Systems'.

[16]. Li, G. (2006). 'Researches on Performance Optimization of Distributed Integrated System Based on Mobile Agent*', 4038–4041.

[17]. Maitrey, S. (2015). 'Handling Big Data Efficiently by using Map Reduce Technique'. doi: 10.1109/CICT.2015.140.

[18]. Mark (2000). *Storage Virtualisation, What is it all about?*

[19]. Mesnier, M. (2003). '01222722', (August), 84–90.

[20]. 'MSST-Cabrera' (1991).

[21]. Oracle, S. (2011). 'Lustre Software Release 2. x Operation Manual'.

[22]. Osero, B. O. (2010). *Storage virtualisation and management*. University of Nairobi.

[23]. Osero, B. O. (2013). 'Network Storage Virtualisation and Management Benard Ong ' Era Osero Lecturer Network Attached Devices , Storage virtualization, Security. *International Journal of Education and Research*, *1*(12), 1–10.

[24]. Panasas, I. (2016). 'Panasas', *Wikipedia*. Available at: https://en.wikipedia.org/wiki/Panasas.

[25]. Pedro Jos´e Marr ´on, Stamatis Karnouskos, D. M. A. O. and the C. consortium (2011) *No Title*.

[26]. Permabit (2015). 'Permabit', *Wikipedia*. Available at: https://en.wikipedia.org/wiki/Permabit.

[27]. Rajguru, P. (2011). 'Available Online at www.jgrcs.info ANALYSIS OF MOBILE AGENT', *Journal of Global Research in Computer Science*, 2(11), pp. 6–10. Available at: www.jgrcs.info.

[28]. Randy, Fellows, A. R. and Kerns, R. (2012). 'SAN Virtualization Evaluation Guide', p. 2.

[29]. Riedel, E. and Nagle, D. (1999). 'Active Disks - Remote Execution for Network-Attached Storage Thesis Committee', *Science*. Available at: https://pdfs.semanticscholar.org/74ac/0dd0a14ea27f016b170a1254c14fe8c73b37.pdf.

[30]. Rodríguez-enríquez, L. R. C. (2015). A general perspective of Big Data : applications , tools', *The Journal of Supercomputing*. Springer US. doi: 10.1007/s11227-015-1501-1.

[31]. Satoh, I. (2011). Mobile Agent Middleware for Dependable Distributed Systems.

[32]. Satoh, I. (2014). MapReduce-based Data Processing on IoT', (iThings). doi: 10.1109/iThings.2014.32.

[33]. Tate, J. (2017). Introduction to Storage Area.

[34]. Tekniska, K., Ögskolan, H. and Simsarian, K. T. (2000). VETENSKAP OCH KONST Dissertation, March 2000 Computational Vision and Active Perception

Laboratory (CVAP).
[35]. Weil, S. A., Brandt, S. A. and Miller, E. L. (2006). CRUSH : Controlled, Scalable, Decentralized Placement of Replicated Data.
[36]. Xu, H. and Shatz, S. M. (2001). A Design Model for Intelligent Mobile Agent Software Systems', pp. 1–23. Available at: file:///C:/Users/ltturche/Downloads/32bfe51224d170bc42.pdf.
[37]. Yazdi, H. T., Fard, A. M. and Akbarzadeh, T, M. R. (2008). 'Cooperative criminal face recognition in distributed web environment', *AICCSA 08 - 6th IEEE/ACS International Conference on Computer Systems and Applications*, 524–529. doi: 10.1109/AICCSA.2008.4493582.
[38]. Kang, S. J., Lee, S. Y. and Lee, K. M. (2015). 'Performance Comparison of OpenMP, MPI, and MapReduce in Practical Problems', *Advances in Multimedia*, 2015, pp. 1–9. doi: 10.1155/2015/575687.
[39]. Satoh, I. (2003). 'Building reusable mobile agents for network management', *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, *33*(3), pp. 350–357. doi: 10.1109/TSMCC.2003.818944.
[40]. Satoh, I. (2004). Dynamic Federation of Partitioned Applications', 2–6.
[41]. Satoh, I. (2016). 'Agent-based MapReduce Processing in IoT', 1(Icaart), 250–257. doi: 10.5220/0005802102500257.
[42]. Satoh, I. and Society, I. C. (2003). 'A Testing Framework for Mobile Computing Software',
[43]. Sarkar, N. I., Member, S. and Halim, S. A. (2011). 'A Review of Simulation of Telecommunication Networks : Simulators, Classification, Comparison, Methodologies , and Recommendations'.
[44]. Kabir, M. H. (2014). 'Detail Comparison of Network Simulators', (November). doi: 10.13140/RG.2.1.3040.9128.
[45]. Sowmya, N., Aparna, M. and Tijare, P. (2015). 'An Adaptive Load Balancing Strategy in Cloud Computing based on Map Reduce', (September), 4–5.
[46]. Li, G. (2006). Researches on Performance Optimization of Distributed Integrated System Based on Mobile Agent. 4038–4041.
[47]. Palanca, J. (2018). 'SPADE Documentation'.
[48]. Tutorialpoint (no date) *REDIS - QUICK GUIDE REDIS - ENVIRONMENT REDIS - DATA TYPES*.
[49]. Alsboui, T., Qin, Y. and Hill, R. (2020). Enabling distributed intelligence for the Internet of Things with IOTA and mobile agents'. Springer Vienna.
[50]. Chang, V., Méndez, V. and Muthu, M. (2020). Emerging applications of internet of things, big data, security, and complexity : special issue on collaboration opportunity for IoTBDS and COMPLEXIS', Computing. Springer, *102*(6), 1301–1304.

How to cite this article: Osero, B. O., Abade, E. and Mburu, S. (2020). A Virtualized Mobile Agents Based IoT Model with Map-Reduce: Using the Python's Spade Framework. *International Journal on Emerging Technologies*, *11*(3): 1147–1156.