Modeling a Fault Tolerant Control Mechanism for Cloud e-marketplaces using Raft Consensus Protocol (RCP)

Ige A.O., Akingbesote A.O., Oriola O., and Aranuwa F.O.

Computer Science Department, Adekunle Ajasin University, Akungba – Akoko, Ondo State, Nigeria

ABSTRACT

The evolution of marketplaces started from the traditional marketplace, the internet marketplace, the web service marketplace, the grid marketplace, before moving to cloud e-marketplace. The need to have rapid access to various service by different customers brought about cloud e-marketplaces. The goal of the cloud e-marketplace is to attract the biggest possible number of buyers while ensuring a reduced waiting time for customers and maximized profit for cloud service providers. Challenges like security, performance and fault tolerance are of great concern in the cloud market. While discussion on the issue of security and performance are ongoing, that of fault tolerance is yet to be fully addressed. Although some researchers have proposed the use of multiple servers in achieving the main idea of the cloud e-marketplace, different kinds of faults still affects the performance of cloud e-market. Balancing of providers’ cost and customers’ waiting time is still a major concern. Various techniques have been proposed in solving these problems. However, these techniques only work in a static environment where these servers can be faulty which may lead to long waiting time. We propose the use of Raft consensus protocol as our fault tolerant approach. We use the dynamic environment as against the already static approached already discussed in the literature. In the dynamic environment, two fault tolerant centers that are capable of surviving failure caused by server overload or congestion are used. These are primary and the reservoir centers. The Raft Consensus Protocol is used in both centers to coordinate the servers and make sure that each of the servers exist either as the leader, a candidate, or a as a follower. A waiting time counter algorithm is developed that directs customers request to the primary center when waiting time t<N, and to the reservoir center when waiting time t≥N. We set up our consumer arrival time, the service time is recorded and our N is set to 5 sec. Furthermore, the result of this research when compared to existing system using various performance metrics showed that the developed mechanism allowed optimal performance in the servers used for cloud e-marketplaces’ service delivery, thereby causing reduced waiting time and increased profit.

Keywords:
marketplaces, cloud e-market, raft consensus protocol, waiting time, profit.

*Correspondence to Author:
Ige A.O.
Computer Science Department, Adekunle Ajasin University, Akungba – Akoko, Ondo State, Nigeria

How to cite this article:
Introduction

A marketplace is a common place that accelerates and makes easy the trade of resources between producers and customers [1]. E-marketplaces can be referred to as the virtual environment for buying and selling of services [2]. The e-marketplace has a variety of names; including web or virtual marketplace, market space, market maker, electronic intermediary, exchange and e-hub, all with diverse definitions [3]. The major difference between e-marketplaces and the traditional marketplace is the use of communication technology in e-market. Early researches on e-marketplaces such as [4] and [5] found that e-marketplaces play a significant role in coordinating inter-organizational activities.

The adoption of e-market is rapidly growing in popularity and taking a critical role in organization’s success across all industries; manufacturing and service, small and medium [6]. With the advancement of the modern human society, basic essential services need to be commonly provided such that everyone can easily obtain access to them, this brought about cloud services. The rapid advances in cloud technologies in the past decade have enabled a wide variety of cloud services [7]. The idea of cloud market was brought from that of utility services, such as water, electricity, gas, and telephone services which are deemed essential for fulfilling daily life routines.

These utility services are accessed so frequently that they need to be available whenever customers make request for them. In order to accommodate at best the increasing demand for these utilities, cloud service providers must adopt effective policies of resource allocation [8]. This will then enable customers to pay service providers based on their usage of these utility services [9]. E-marketing is fast gaining ground as an accepted business paradigm. The adoption of cloud in e-market refers to not only “the applications delivered as services over the internet” but also “the infrastructures and systems in the datacenters” [10]. In cloud computing, marketplaces offer to customers an independent access to cloud services [1]. Cloud computing reshaped the way services are delivered, this is apparent from the definition in [11]. This definition made us to know that cloud computing is “a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services). These resources can be rapidly provisioned and released with minimal management effort or service provider interaction. This implies that cloud offers distinguishable capabilities such as on-demand self-service, broad network access, resource pooling, rapid elasticity, and measured service [11].

Prolonged waiting time will cause a service provider to lose customers to a faster service provider. One noticeable thing is that as the number of customers increase, the waiting time also increases. Addition of server to solve this problem brings additional cost to the provider. Balancing the trade-off between customer waiting time and providers’ cost is a great challenge. Therefore optimization of these servers in terms of servers’ fault tolerant capability to balance the trade-off is the focus of this research. This research proposes a fault tolerant control mechanism for the management of cloud e-marketplace using raft consensus protocol.

Server optimization is extremely important in cloud service deployment. This is because profit maximization and waiting time reduction depends mainly on it. Most research works on performance in cloud e-marketplaces focused on performance analysis and maximization of profit. See for example [12], [13], [14] and [15]. This is done by increasing the number of servers used for cloud service delivery. One important thing that has great effect on cloud performance and that is yet to be fully discussed in the literature is that of cloud fault tolerance capability. That is, how the cloud can
tolerate fault for effective performance. For example, when the used servers are overloaded, this can sometimes lead to omission failure, byzantine failure, and other performance related failures. This may lead to prolonged customers’ waiting time and sometimes loss of data. This will invariably affect the providers in terms of costs. Therefore, effective management of these servers is highly needed.

This research will attempt to approach these challenges by designing a good fault tolerant control mechanism using raft consensus protocol in a typical cloud e-marketplace. This research is closely related the work of [16] and [17]. What differentiate this work are:

i. The inclusion of fault tolerant mechanism in a dynamic environment

ii. The introduction of raft protocol in the context of cloud e-market community.

This will allow a collection of cloud e-market infrastructures to work as a coherent group that can survive minor or major failure. This to the best of the researcher’s knowledge is yet to appear in the literature.

The rest of this paper is organized as follows: Section II: Literature Review, Section III: Materials and Methods, Section IV: Results and Discussions, Section V: Conclusion and Recommendation and Section VI: References.

LITERATURE REVIEW

Electronic marketplace or e-marketplace as an online market environment over the internet where both sellers and buyers act as market players to exchange goods and services [18]. The authors in [19] defined a marketplace as an area of exchange in which many buyers and vendors meet in order to conduct business transactions. The goal of the e-marketplace is to attract the biggest possible number of buyers and suppliers, which will become members of that e-marketplace [20].

Cloud e-marketplace has three important phases: The cloud implementation, performance and optimal provisioning of servers [21]; which will minimize cost and customer’s waiting time. A lot of researchers have worked on these phases. For example [2], [22], [8], [9], [23], [12], [13], [24]. Cloud computing is generally associated with three main service models: Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS) [11]. Although these service models operate in different ways, service provisioning follows almost the same blueprint. The author in [25] emphasized that a major challenge faced by cloud service providers is how to monitor if the servers used in deploying their services are being utilized with the changing load on the system.

With the development of internet, buyers and sellers use the internet as a quick and timely communication channel, and switch to conduct business online [26]. Thus, more and more business companies favor to do business on e-Market. There are three main functions of a marketplace according [18] matching of buyers and sellers, facilitate the exchange of information, goods and services, providing an infrastructure that enables the efficient functioning of the market. In [27], the author categorized E-marketplaces into different types according to their ownership or governance. These types are: (i) Private Marketplace: Owned by an individual company which focus on connecting buyers and sellers to the marketplace, (ii) Public Marketplace: Described as a free marketplace or Business-to-Business (B2B), it is often owned by third party organization, (iii) Consortia Marketplace: Created as a result of competitive companies from same industry and having same set of products and services, (iv) Community Marketplace: Owned by multiple participants with varied backgrounds.

Also, e-marketplace was classified in [28] based on the way in which they are operated; Independent e-marketplace, Buyer oriented e-marketplaces, Supplier-oriented e-Marketplace, Vertical and Horizontal e-marketplaces.
The author in [18], developed a reference model for e-marketplaces that defines three phases: The information phase, agreement phase and settlement phase which was an improvement on the reference model developed for e-marketplaces in [29].

Research shows that e-market has grown from web market to grid and cloud market. Researchers have worked in the context of web and grid. See for example [30], [31], [32], [33], [34]. Our work focuses on cloud market and a detailed literature is carried out in the area of resource scheduling in cloud e-market and fault tolerance.

TRADITIONAL MARKETPLACE:
A traditional marketplace is a location where buyers and sellers come together to perform transactions. The buyer and seller meet in person to talk about the product, price and in some cases delivery arrangements [35]. Traditional market, considered as a key retail space for affordable produce and low-income families, is presently in danger of decline and even disappearance.

![Figure 1: Evolution of Cloud e-marketplaces](image)

It is being affected by real estate pressures, retail globalization trends and the “urban renaissance” in cities. It is argued that the reasons for the decline of the traditional retail market in the world have to be contextualized within the particular trajectory of recent neoliberal urban political economy rather than any supposedly “natural” trends in retail geography and consumer behavior.

Research works in [36] and [37] highlighted the importance and diversity of traditional marketplace. The traditional e-marketplace is a web portal where buyers and suppliers come together to explore new business opportunities [20]. Traditional marketplace can be said to be the foundation on which all other marketplaces were built. The author in [38] listed some benefits of traditional marketplaces and explained that one can easily reach the local target audience, and also stated that marketing (advertising products and services to potential customers) is easier in traditional marketplaces. According to [20], another advantage of the traditional marketplace is that suppliers will be paid for the delivered goods, even if the supplier does not know if the buyer is trustworthy or not. The authors also stated that the challenge with traditional marketplaces is that it supports a single business model which is only effective in dealing with simple exchanges, therefore leading to the concept of internet marketing.

INTERNET E-MARKETPLACES
The authors in [18] defined an electronic marketplace or e-marketplace as an online market environment over the Internet where both sellers and buyers act as market players to exchange goods and services. The author in
listed in addition to some benefits of the traditional market; richness, cost effectiveness and extra value provisioning as some advantages of Internet E-marketplaces. Internet marketplaces reshaped the way in which business transactions are carried out. The authors in [21] highlighted the emergence of Service Oriented Architecture (SOA) which internet e-marketplaces couldn’t cope with as the reason for the introduction of Web Service Marketplaces.

WEB SERVICE MARKETPLACES
The authors in [40] listed three benefits of service marketplace as; a third-party service, once implemented, can be reused across all applications. Second, since cloud providers vet every service listed on their marketplace, applications can use the services offered in a marketplace without having to vet them on their own. Third, the marketplace will democratize the ability to add to the cloud’s functionality and thereby lead to a greater rate of innovation than when all the burdens lie on the cloud provider.

GRID E-MARKETPLACES
The advent of grid e-market is the inability of the web service e-marketplaces to perform at its best when used for high computational power. The author in [21] explained Grid e-market as a market where computational power is purchased by consumers through the use of middleware or a resource allocation broker.

CLOUD E-MARKETPLACES
The term Cloud Computing (CC) was first proposed by Professor Ramnath Chellappa in 1997 in a lecture in which he explained it as a novel computing paradigm where the computing boundaries will be found by economic rationale instead of technical restrictions. The basic theory of cloud computing is that IT resources are made available, within an environment that enables them to be used, via a communications network as a service [41]. There are two classifications of IT resources; it can be either raw computation resources, or certain computation resources that are programmed to perform a function [42]. The new cloud environment enables managers to work, select, configure, and deploy those resources.

The adoption of cloud in e-market refers to not only “the applications delivered as services over the Internet” but also “the infrastructures and systems in the datacenters” [10]. A lot of businesses are utilizing cloud services. Gartner [43] said worldwide public cloud service was projected to grow 18 percent in 2017 to total $246.8 billion, up from $209.2 billion in 2016. Gartner also said the highest growth will come from cloud system infrastructure service which was projected to grow 36.8 percent in 2017 to reach $34.6. He also forecast that the global public cloud service market is expected to reach almost $247 billion by 2018 and grow to $383 billion by 2020.

Figure 2: Global market of public cloud service
Https://escipub.com/american-journal-of-computer-sciences-and-applications/
The benefits of cloud services as described in [44] are scalability, cost effectiveness, immediate availability and performance. According to [7], developing effective methods for evaluating cloud service performance is a very important research problem that has attracted extensive attention from both academia and industry.

Researchers have taken different types of approaches to tackling this challenging problem from various perspectives. Various approaches for effective cloud service delivery exist in the literature, see for example [45], [46], [27], [15]. The author in [46] concluded that an efficient resource allocation method should meet some criteria such as cost reduction, power reduction, energy reduction, Quality of Service (QoS) enhancement, and utilization of resources. The work of [45] considered the cloud e-marketplace as an ecosystem that host heterogeneous cloud services from different cloud providers and supports collaboration. The author developed a cloud service selection framework that has a feature that aids numerous customers’ requirements.

Cloud service providers are faced with various users with various demands. The cloud vendor desires to meet the various demands to maximize profits corresponding to which diversified pricing policies are considered necessary. Meanwhile, a variety of request durations creates a dynamic computing resource allocation problem [12].

The work of [15] focused on scheduling customer requests for SaaS providers with the explicit aim of cost minimization with dynamic demands handling by considering the customers’ requests for an enterprise software services from a SaaS provider by agreeing to the pre-defined Service level Agreement (SLA) clauses and submitting their Quality of Service (QoS) parameters. This research allowed customers to dynamically change their requirements and usage of cloud services. The cloud service provider’s objective is to schedule a request such that its profit is maximized while the customers’ (QoS) requirements are assured.

Three cost driven algorithms were proposed in [15] to allow service providers to maximize profit. The author focused on profit maximization on the side of service providers minimizing violations of pre-defined Service Level Agreements and re-using servers. The author measured the effectiveness of the algorithms based on some parameters: Arrival rate variation, Impact of QoS parameters, and others. However, the author only focused on Quality of Service for customers and did not consider optimizing the performance of the Servers to reduce customer’s waiting time.

Du in [12] focused on allocating resources in a way that is sensible and cost effective by considering two cloud deployment models, the author called one deployment model the hybrid cloud which faces contract adopters who choose to rent exclusive Virtual Machine servers to handle the IT workload of regular hours but also buy public computing resources to handle peak hours’ IT workload, and called another deployment model a public cloud which faces walk-in adopters who choose to only buy pooled resources when they need but willing to pay high prices.

The author assumed that all physical machines are identical and parallel. Similarly, the author also assumed that the duration of each request for individual arrivals are independent and identically distributed non-negative random variables and are independent of the arrival process. The probability distributions of the pattern of arrivals are assumed to follow Poisson distributions with different arrival rates. The service distributions are assumed to follow exponential distributions. The public cloud follows “first come first served” in most of the case. However, when the hybrid adopters have heavy traffic that could not be handled by the exclusive Virtual Servers, those requests wait in the private Virtual Machine with a probability and also departs with a probability and jump into the line of public cloud with a priority. A
delay cost per unit time per request is generated due to this congestion.

The result in this research showed that the developed model could help cloud providers decide which deployment model to be chosen. The cloud vendor could briefly allocate pooled resources on each virtual machine on each machine optimally. Also, the model could tell if the cloud vendor should expand the capacity (increase the number of servers on the server farm) to meet the demand. However, the developed model focuses on pricing and increasing numbers of servers without discussing how to manage the performance of the already available servers.

The authors in [13], modeled a cloud server farm as a queuing system which indicates that the inter-arrival time of requests is exponentially distributed, while task service times are independent and identically distributed random variables that follow a general distribution with mean value of $\mu$. The author considered the developed model to contain $m$ number of servers which renders service in First come First serve basis. The author modeled the arrival process as a markovian process by exploiting [47], but stated that his work can be classified as “Semi-Markovian”. The research concluded that due to the nature of the cloud environment, the author assumed the general service time for requests as well as large number of servers, which makes the developed model in this work more flexible in terms of scalability and diversity of service time. Numerical analysis and simulation carried out to validate the model showed that the approximate method used provides results with high degree of accuracy for the mean number of tasks in the system, and also indicates that a cloud centre that accommodates heterogeneous services may impose longer waiting time for its clients compared to its homogeneous equivalent with the same traffic intensity. However, it does not discuss successful management of server farms.

Yin in [48], developed a fault tolerant control system which is suitable for online implementation and consists of two parts, the adaptive residual generator and a gradient based optimizer. The effectiveness of the developed model was demonstrated through the TE benchmark model, and the result shows that the system adapts to faulty situations such that the system continues to satisfy its goal, and the process performances are automatically optimized.

Bala and Chana in [49] discussed the challenges of fault tolerance, the techniques and its implementation in cloud computing. The authors proposed HAproxy architecture by introducing a backup server in order to ensure availability, but didn’t discuss how the performance of these servers can be optimized.

The authors in [45], developed a fuzzy oriented framework for cloud selection which according to the authors, is capable of handling complex user requirements, enabling flexibility to accommodate subjective quality of service inputs, improved presentation format for search results to aid decision making.

The authors in [16], developed a prescriptive model that changes the numbers of servers dynamically based on customer’s waiting time. The authors employed the N-Policy as seen in [50] which enables the servers to be turned on when the system is busy and turned off when it is idle. [16] developed a model that consist of a fixed number of servers ($M_1, M_2, M_3$) and a limited set of backup servers called reservoir servers. The model works by transferring customers to reserved centres when the waiting time of the customer reaches a point denoted by “N”. The authors used queuing theory as the proof of concept.

The result obtained when the model in [16]; which has variable servers was compared to models with fixed servers reveals a better performance in terms of server utilization and optimal server provisioning and also, the cost benefit accrued using this model is greater than the cost, therefore concluding that the
developed model is profitable. However, fault tolerance in the reserved centres was never discussed; a situation whereby failure occurs in the reservoir centre and the customer’s waiting time reaches the point N, then the whole system will be ineffective.

Early research on fault tolerance, [51] defined fault tolerance as the ability of a system to continue its intended operation, possibly at a reduced level, rather than failing completely, when some part of the system fails.Fault tolerance can be divided into two subareas, the hardware and software fault tolerance [52]. Most fault tolerant systems are either based on replication or redundancy.

Various literatures exist on approaches to fault tolerance for example [53][54][55]. The authors in [52] explained the approaches and stated that; in redundancy technique, multiple identical instances of the same system or part are provided, and the failed instance is switched to one of the remaining instances in case of a failure. But in replication technique, multiple identical instances of the same system or part are provided, and tasks or requests are made to perform in all of instances in parallel, allowing the selection of correct result from the instances.

The idea of fault tolerance using consensus protocol is not new. The work of [56] developed an affine based consensus protocol for grid computing which is an enhanced model for self-validated numerical analysis in which the quantities of interest are represented as affine combinations of certain primitive variables. But in the context of cloud marketplaces, fault tolerance using consensus protocol is yet to be discussed.

The Raft algorithm which is an improvement on the paxos algorithm in [57] was developed by Ongaro in [17]. Although the most widely used consensus protocol is the Paxos algorithm and it has been the subject of various researches as seen in [58], [59], [60], [61], [62], [63], [64], [65], [66], [19], to mention a few. Marandi in [63] emphasized that Paxos is one of the dominant protocols in building fault-tolerant systems; however, Paxos algorithm is extremely difficult to understand [61], [62], [58]. Two major drawbacks were emphasized in [67], that Paxos algorithm is exceptionally intricate to understand and also that Paxos algorithm does not provide a good foundation for building practical implementations because the Paxos architecture is not suitable for building practical systems. An example of this is Google’s chubby [68], where the developers concluded that there are significant gaps between the description of the Paxos algorithm and the needs of a real-world system, thereby causing the final developed system to be based on an unproven protocol, hence the reason for the adoption of Raft consensus protocol in this research above other consensus protocols like we have in [57], [69] and [70].

Raft consensus protocol according to [17], is also more suitable than other consensus algorithms for real-world implementations, it performs well enough for practical deployments, and it addresses all aspects of building a complete system.

The importance of consensus as stated in [67] are that:

i. They ensure safety,

ii. They are fully functional (available) as long as any majority of the servers are operational and can communicate with each other and with customers,

iii. They do not depend on timing to ensure consistency,

iv. A minority of slow servers need not impact overall system performance.

CONSENSUS IN CLOUD

The work of [71] explicitly stated Agreement, Validity and Termination as the major conditions required to be met before consensus can occur. The idea of consensus pertains majorly to researches on distributed systems, where a lot of research works focused on adopting consensus in solving byzantine failures. The work of () concluded that since
most research works on cloud market resource allocation proposed the use of multiple servers in cloud service delivery.

The idea of failure detection service and flexible failure handling framework was discussed in [53], but this was done in the context of grid. The work considered the detection of both task crashes and user-defined exceptions which solved the limitations of the GRAM protocol as discussed in the early work Foster and Kesselman in [72].

The work of [73] concluded that since most research works on cloud market resource allocation proposed the use of multiple servers in cloud service delivery, that existing approaches in distributed systems can be applied to achieve enhanced performance in cloud e-marketplaces.

The idea of some cloud architectures being related to distributed systems was also mentioned in [74], where federated internetworking of administratively distributed cloud was discussed and benefits of such distributed cloud architectures include (i) improving the ability of cloud service providers to meet the required QoS levels, (ii) adapting to failures such as natural disasters, etc.

**MATERIALS & METHODS**

The basic idea of the RCP is server election. We made use of Cloudsim plus as the simulation environment for this work and modeled two (2) centres with four (4) servers each. Each server in our model can exist either as the leader, a follower or a candidate. Leader takes place, and the server elected in the primary centre as the leader; either server S1, S2, S3 or S4 will coordinate the operations of other servers in the primary centre, thereby making it fault tolerant. But should there be any problem in the primary centre or whenever a customer’s request waits for Time ≥ N, it indicates there is a problem in the primary centre, then the waiting time counter automatically transfers customer’s request to the reservoir centre which also utilizes the raft consensus protocol. i.e. election will take place in the reservoir centre and either Servers S5, S6, S7 or S8 will coordinate the operations in the reservoir centre. The leader will be responsible for accepting customer’s request. Customer requests will flow only in one direction: from leader to other servers. Each centre is designed to communicate using a form of signal, and the basic consensus algorithm requires only two types of signals between servers. RequestVote signals, which will be initiated by candidates during elections to request for votes, and AppendEntries signals will be initiated by leaders to provide a form of heartbeat to notify the candidate and follower of a leader’s presence, thereby preventing an election from taking place.

![Figure 3: Architecture of the developed model](image-url)
LEADER ELECTION

The work of [17] explains in detail how election takes place in the raft consensus protocol. This will be adopted in describing how leader election will take place in the model to be developed.

The leader elected in both centers will periodically send a heartbeat to other servers to maintain authority. A leader election will be triggered when a server times out after waiting for a heartbeat from the leader. To begin an election in the reservoir room, a server in the follower state increases its state to the candidate state and votes for itself. It then sends the RequestVote signal to other servers in the Reserve Centre.

Figure 4: Leader Election in the developed model Source: Ongaro 2014, [17]

RULES OF OPERATION

Requestvote Signals

Arguments:
Term: candidate’s term
CandidateNumber: candidate requesting vote
term: current term, for candidate to update itself
voteGranted: true means candidate received vote

1. Reply false if term < currentTerm
2. If votedFor is null or candidateNumber, and candidate’s log is at least as up to date as receiver’s log, grant vote.

APPENDENTRIES Signals

Used by leaders to replicate log and also used as heartbeat

Arguments:
Term: leader’s term
leaderId: so follower can redirect customers
prevLogIndex: index of log entry immediately preceding new ones
prevLogTerm: term of PrevLogIndex entry
entries[]): log entries to store (empty for heartbeat; may send more than one for efficiency)
leaderCommit: leader’s commitIndex

Domino effect:
1. Reply false if term < currentTerm
2. Reply false if log doesn’t contain an entry at PrevLogIndex whose term matches

PrevLogTerm

3. If an existing entry conflicts with a new one (same index but different terms), delete the existing entry and all that follows it.
4. Append any new entry not already in the log
5. If leaderCommit < commitIndex, set commitIndex = min(leaderCommit, index of last new entry)
MATHEMATICAL MODEL

In order to find the best mathematical model to support this research, various literatures were searched, but the most appropriate mathematical concept to support this research is based on [75] and [76].

The model developed in this work has the following assumptions:

i. Customers seek to be served as soon as a request is made
ii. Customers are served on a FIFO basis
iii. Customers’ request arrival rate is random
iv. Service is random i.e., as long as a customer is being served by a server, the customer’s request must be completed during an interval of $j$. $j$ will be considered as the exponential holding time.

The probability that a customer make use of the reserved centre during an interval $j$ is given as

$$ P_n(t) = aj + o(j) \quad \{1\} $$

where $a$ is the average arrival rate.

Assume that average processing rate of all servers in the centres are equal, and if they were to serve the customer independently it would be $L$. Assume that the service each performs on the customer causes each to have an average processing rate of $2L$.

Let $P_n(t)$ be the probability that at time $t$, there are $n$ customers in the system,

Let $r$ be the probability that only one server in both centres can be elected as the leader and can service a customer's request at a particular time $h$.

Then:

$$ P_n(T + h) = P_{n-1}(T)ah + P_n(T)(1 - 2rLh)(1 - ah) + P_{n+1}(T)(2rLh) + o(h) \quad \{2\} $$

Thus,

$$ \frac{dP_n(T)}{dt} = aP_{n-1}(T) - (2rL + a)P_n(T) + 2rLP_{n+1}(T) \quad \{3\} $$

If $n = 0$,

$$ P_0(T + h) = P_0(T)(1 - ah) + P_1(T)(2rLh) + o(h) \quad \{4\} $$
\[
\frac{dP_0(T)}{dT} = -aP_0(T) + 2rLp_1(T) \quad \{5\}
\]

Statistical equilibrium is assumed so that \(dP_n(T)/dT = 0, (n = 0,1,2, \ldots)\) and also, \(\lim_{T \to \infty} P_k(T) = p_k\) has a unique value for all \(k = 0,1,2, \ldots\)

Then,
\[
p_1 = \frac{a}{2rL} p_0, \quad \{6\}
\]
\[
p_2 = (\frac{a}{2rL})^2 p_0, \quad \{7\}
\]
\[
p_k = (\frac{a}{2rL})^k p_0 \quad \{8\}
\]

From \(\sum_{i=0}^{\infty} p_i = 1\),
\[
p_0 [1 + \frac{a}{2rL} + (\frac{a}{2rL})^2 + \cdots] = 1 \text{; provided } a < 2rL \quad \{9\}
\]

Then \(p_k = \frac{a^{k(2rL-a)}}{(2rL)^{k+1}} \quad \{10\}\)

\(k = (0,1,2, \ldots)\) will give the probability that the servers are in a particular state (follower, candidate or leader) upon arrival of a customer’s request into any of the centres.

**RESULTS AND DISCUSSION**

Performing benchmarking experiments in repeatable, dependable, and scalable environments using real-world cloud environments is not possible, hence the need to simulate using a viable simulation tools [74].

Simulation of cloud environment is extremely challenging because cloud exhibit varying demands, supply pattern and resources [74]. Various distributed system simulators for example [77] and [78] cannot offer the environment needed for cloud simulation, hence the reason for simulating this research with Cloudsim plus.

Both the existing system and the developed system showed no significant difference when they were tested with low throughput.

**Table 1: Throughput Reading of Developed System**

<table>
<thead>
<tr>
<th>MIPS</th>
<th>1ST CENTRE</th>
<th>2ND CENTRE</th>
<th>3RD CENTRE</th>
<th>4TH CENTRE</th>
<th>5TH CENTRE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>4.2 P</td>
<td>1.1 P</td>
<td>4.2 P</td>
<td>4.2 P</td>
<td>4.31 P</td>
</tr>
<tr>
<td>2000</td>
<td>8.32 P</td>
<td>2.1 P</td>
<td>2.1 P</td>
<td>4.21 P</td>
<td>8.21 P</td>
</tr>
<tr>
<td>3000</td>
<td>12.2 P</td>
<td>12.2 P</td>
<td>3.1 P</td>
<td>6.21 R</td>
<td>12.2 P</td>
</tr>
<tr>
<td>4000</td>
<td>16.31 R</td>
<td>4.21 P</td>
<td>4.21 P</td>
<td>16.1 R</td>
<td>16.2 R</td>
</tr>
<tr>
<td>5000</td>
<td>20.1 R</td>
<td>20.1 R</td>
<td>10.1 R</td>
<td>10.1 R</td>
<td>20.2 R</td>
</tr>
</tbody>
</table>

[Graph showing 1ST Reading vs MIPS]
Ige A.O. et al., AJCSA, 2019; 3:14

2ND READING

3RD READING

4TH READING

5TH READING

The result obtained from the developed model as shown in the graph shows positive correlation between the throughputs measured in Million Instruction Per Second (MIPS) and the time it takes to execute the throughput. Result shows that the developed system was able to handle 1000 MIPS at an average time of 3.602, 2000 MIPS at 4.988 seconds, 3000 MIPS at 9.182 seconds, 4000 MIPS at 11.406 seconds and 5000 MIPS at 16.12 seconds.

An existing system that does not utilize the raft consensus protocol in control management was also tested using the same metric (Throughput), at the same condition and also at the same number of times (5). The result is shown below.

### RESULTS FOR NON RAFT PROTOCOL UTILIZING MODEL

Table 2: Throughput Data for an existing system

<table>
<thead>
<tr>
<th>MIPS</th>
<th>1ST READING</th>
<th>2ND READING</th>
<th>3RD READING</th>
<th>4TH READING</th>
<th>5TH READING</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>12.1 P</td>
<td>12.1 R</td>
<td>14.7 R</td>
<td>15.6 R</td>
<td>17.5 R</td>
</tr>
<tr>
<td>2000</td>
<td>22.7 R</td>
<td>26.7 P</td>
<td>25.6 P</td>
<td>26.7 R</td>
<td>25.3 P</td>
</tr>
<tr>
<td>3000</td>
<td>0 -</td>
<td>0 -</td>
<td>0 -</td>
<td>0 -</td>
<td>0 -</td>
</tr>
<tr>
<td>4000</td>
<td>0 -</td>
<td>0 -</td>
<td>0 -</td>
<td>0 -</td>
<td>0 -</td>
</tr>
<tr>
<td>5000</td>
<td>0 -</td>
<td>0 -</td>
<td>0 -</td>
<td>0 -</td>
<td>0 -</td>
</tr>
</tbody>
</table>
A total of five (5) readings were taken to compare the throughput at 1000, 2000, 3000, 4000 and 5000 MIPS for our developed model and the existing system.

The first reading was able to execute customer requests at 1000MIPS, and also at 2000MIPS, but crashed at 3000MIPS. Also, the trend shows that the response time increases with increased throughput i.e. the higher the throughput, the higher the time it takes to process requests.

Centre utilization was also considered as a metric for evaluating the performance of the developed model with the existing one. From Table 4.1, the developed model made use of the Primary Center seventeen (17) times while it made use of the reservoir centre six (8) times, with no failure recorded in the whole 25 processes.

From Table 2, the existing model was able to successfully execute ten (10) processes between 1000 MIPS and 2000 MIPS, but failed to process 3000MIPS, 4000 MIPS and 5000 MIPS.
CONCLUSION AND RECOMMENDATION

The adoption of Raft Consensus Protocol (RCP) in achieving fault tolerance in cloud e-marketplaces has shown foremost upgrade with respect to existing systems. Even though the developed system showed no major difference in response time when tested with low throughput values (100MIPS to 500 MIPS), significant differences in the response time were recorded when the system was tested with high throughput values (1000 MIPS to 5000 MIPS).

Experimental conditions were not altered in comparing the developed system to an existing system, but in order to achieve accurate results, Raft Consensus Protocol (RCP) was bypassed to mimic an existing system and also the same metrics were used for evaluating the existing system. The result we obtained showed that the developed system can successfully execute customers’ requests at up to 5000 MIPS, while the existing system could only execute customers’ requests at a maximum of 2000 MIPS.

The trend shows a positive correlation between increasing throughput and increasing waiting time, because as the throughput value increased, the response time also increased.

Server Utilization comparison also showed that the developed system made use of the primary centre 68% of time and made use of the reservoir centre 32% of time in 25 trials across 1000-5000 MIPS. The existing system was able to make use of the primary centre 16% of time, made use of the reservoir centre 24% of the time while it could not process any request due to failure for 60% of the time.

REFERENCES

6. H. M. Mutlu and A. Sürer, “Effects of market, e-marketing, and technology orientations on


