ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 1, Issue 6, June 2012

A Light-weight Data Replication for Cloud Data Centers Environment

Mohamed-K HUSSEIN, Mohamed-H MOUSA

Abstract— Unlike traditional high performance computing environment, such as supercomputers, the cloud computing is a collection of interconnected and virtualized computing resources that are managed to be one or more unified computing resources. The Cloud environment constitutes a heterogeneous and a highly dynamic environment. Failures on the data centers storage nodes are normal rather than exceptional. As a result, the cloud environment requires some capability for an adaptive data replication management in order to cope with the inherent characteristic of the Cloud environment. In this paper, we propose a data replication strategy to adaptively select the data files which require replication in order to improve the availability of the system. Further, the proposed strategy decides dynamically the number of replicas as well as the effective data nodes for replication. Experimental results show that the proposed strategy behaves effectively to improve the availability of the Cloud system under study.

Keywords— **System Availability, Replication, Adaptive, Cloud Computing**

I. INTRODUCTION

Cloud computing is a large-scale parallel and distributed computing system. It consists of a collection of inter-connected and virtualized computing resources that are managed to be one or more unified computing resources. Further, the provided abstract, virtual resources, such as networks, servers, storage, applications and data, can be delivered as a service rather than a product. Services are delivered on demand to the end-users over high-speed Internet as three types of computing architecture, namely Software as a Service (SAAS), Platforms as a Service (PAAS) and Infrastructure as a service (IAAS). The main goal is to provide users with more flexible services in a transparent manner, cheaper, scalable, highly available and powerful computing resources [1]. The Software as a Service (SaaS) architecture provides software applications hosted and managed by a service provider for the end-user replacing locally-run applications with web services applications. In the Infrastructure as a Service (IaaS), Service includes provision of hardware and software for processing, data storage, networks and any required infrastructure for deployment of operating systems and applications which would normally be needed in a data center managed by the user. In the Platform as a Service (PaaS), Service includes programming languages and tools and an application delivery platform hosted by the service provider to support development and delivery of end-user applications [1, 2].

In general, the Cloud Computing provides the software and hardware infrastructure as services using large-scale data centers. As a result, Cloud computing moved away the computation and data storage from the end user and onto servers located in data centers, thereby relieving users of the burdens of application provisioning and management. Software can then be thought of as purely a service that is delivered and consumed over the Internet, offering users the flexibility to choose applications on-demand and allowing providers to scale out their capacity accordingly. However, it is challenging to provide high availability and efficient access to the cloud data centers because of the large scale and dynamic nature of the Cloud. Replication is the process of providing different replicas of the same service at different nodes. Replication is a used technique in different clouds, such as GFS (Google file system) and HDFS (Hadoop Distributed File System) [3, 4]. In the cloud, data replication is achieved through data resource pool and the number of data replicas is statically set based on history and experience. Further, it is not necessary to create replica for all data files, especially for those non-popular data files. Therefore, it is necessary to adaptively replicate the popular data files, determine the number of data replicas and the data nodes where to place the new replicas according to the current cloud environments conditions. In this paper, we propose an adaptive replication strategy in a cloud environment that adaptively copes with the following issues:

- Which data should be replicated and when to replicate in a cloud systems to improve the data files and the overall system availability. Further, the selection process must take into account the users requirements on waiting time reduction and data access speeding up.
- How many suitable new replicas should be created in the cloud to meet a reasonable system availability requirement? With the number of new replicas increasing, the system maintenance cost significantly increases, and too many replicas may not increase availability, but bring unnecessary overhead cost instead.
- Where the new replicas should be placed to enhance the users' tasks response time and bandwidth consumption requirements. By keeping all replicas active, the replicas may improve system task successful execution rate and bandwidth consumption if the replicas and requests are reasonably distributed. However, appropriate replica placement in a large-scale, dynamically scalable and totally virtualized data centers is much more complicated.

The proposed adaptive replication strategy is originally motivated by the fact that the recently most accessed data files will be accessed again in the near future according to the collected prediction statistics of the files access pattern

[5]. A replication factor is calculated based on a data block and the availability of each existing replica passes a predetermined threshold, the replication operation will be triggered. A new replica will be created on a new block which achieves a better new replication factor. The number of new replicas will be determined adaptively based on enhancing the availability of each file heuristically.

The remainder of this paper is organized as follows. Section II presents the related work on data storage and data replication of cloud computing systems. Section III presents a formalization of a cloud system model. Section IV describes the dynamic data replication strategy, including the replication decision, the number of replicas, and the replica placement. Section V addresses the simulation environment, parameter setup and performance evaluation of the proposed dynamic data replication strategy. Finally, conclusions and future work are given in Section VI.

II. RELATED WORK

This section presents two broad categories of related work. The first category discusses cloud data storage, and the second category presents the related work to the cloud data replication.

A. Cloud Data Storage

Cloud computing technology moved computation and data storage away from the end user and onto servers located in data centers, thereby relieving users of the burdens of application provisioning and management. As a result, software can then be thought of as purely a service that is delivered and consumed over the Internet, offering users the flexibility to choose applications on-demand and allowing providers to scale out their capacity accordingly. Many large institutions have set up data centers and cloud computing platforms, such as Google, Amazon, IBM. Compared with traditional large scale storage systems, the clouds which are sensitive to workloads and user behaviors focus on providing and publishing storage service on Internet [6-8]. The key components of the cloud are distributed file systems, such as The Google File System GFS, the Hadoop distributed file system HDFS. In the GFS [3], there are three components, multiple clients, a single master server, and multiple chunk servers. Files are stripped into one or many fixed size chunks, and these chunks are stored in the data centers, which are managed by the chunk servers. Chunks are stored in plain Linux files which are replicated on multiple nodes to provide high-availability and improve performance. The master server maintains all the metadata of the file system, including the namespace, the access control information, the mapping from files to chunks, and the current locations of chunks. Clients interact with the master for metadata operations, but all data bearing communication goes directly to the chunk servers. Secondary name servers provide backup for the master node.

In a multi-cluster system, each cluster is a complete GFS cluster and with its own master, and each master maintains the metadata of its own file system. Different masters can share the metadata by the namespace, which describes how the log data is partitioned across multiple clusters [9]. Compared with a single cluster, in a multi-cluster system, the performance of the cloud system and the size of the cloud data storage can be improved significantly. The mechanism of HDFS is similar to that of GFS, but it is light-weighted and open-source [4]. HDFS also follows a master/slave architecture which consists of a single master server that manages the distributed file system namespace and regulates access to files by clients called the *Name node*. In addition, there are multiple data nodes, one per node in the cluster, which manages the disk storage attached to the nodes and assigned to Hadoop. The Name node determines the mapping of blocks to data nodes.

B. Cloud Data Replication

Replication technology is one of the useful techniques in distributed systems for improving availability and reliability. In Cloud computing, replication is used for reducing user waiting time, increasing data availability and minimizing cloud system bandwidth consumption by offering the user multiple replicas of a specific service on different nodes. For example, if one node fails, a replica of the failed service will be possibly created on a different node in order to process the requests [8]. Data replication can be classified into two categories: static replication [3, 4, 9] and dynamic replication algorithms [6, 9]. In a static replication, the number of replicas and their locations are predetermined. On the other hand, dynamic replication dynamically creates and deletes replicas according to changing environment load conditions. There has been an interesting number of works for data replication in the Cloud computing. For example, in [3], a static distributed cloud data replication algorithm is proposed. In the GFS, a single master considers three factors when making decisions on data chunk replications: 1) to place the new replicas on chunk servers with below-average disk space utilization; 2) to limit the number of “recent” creations on each chunk server; 3) to spread replicas of a chunk across racks. A data chuck is replicated when the number of replicas falls below a limit specified by the users. Similarly, in [4], an application can specify the number of replicas for each file, and the block size and replication factor are configurable per file. In [9], a p-median static centralized data replication algorithm is proposed. The p-median model finds p replica placements sites that minimize the request-weighted total distance between the requesting sites and the replication sites holding the copies assigned. In [6], a dynamic distributed cloud data replication algorithm CDRM is proposed. The CDRM is designed on the HDFS platform, the data replica placement is based on the capacity and location according to workload changing and node capacity, and the lower bound of the number of replicas is dynamically determined according to the availability requirement. In [9], six different dynamic data replication algorithms, Caching-PP, Cascading-PP, Fast

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 1, Issue 6, June 2012

Spread-PP, Cascading-Enhanced, and Fast Spread-Enhanced are proposed. In [10], a dynamic centralized data replication algorithm MinDmr is proposed. MinDmr treats hot and cold data differently and uses a weighting factor for the replication. MinDmr is developed into four prediction-based replica schemes. Similarly, in [4], an replication algorithm is proposed which selects a popular file for replication and calculates a suitable number of copies and grid sites for replication. The differences between the mentioned replication algorithms and our proposed strategy lie in the following aspects. 1) A heuristic is proposed based on a formal model that describes the relationship between the data files availability and the number of replicas. 2) The popular data is identified according to the history of the user access to the data. When the popularity of a data file passes a dynamic threshold, the replication operation will be triggered. 3) Replicas are placed among data nodes in a balanced way.

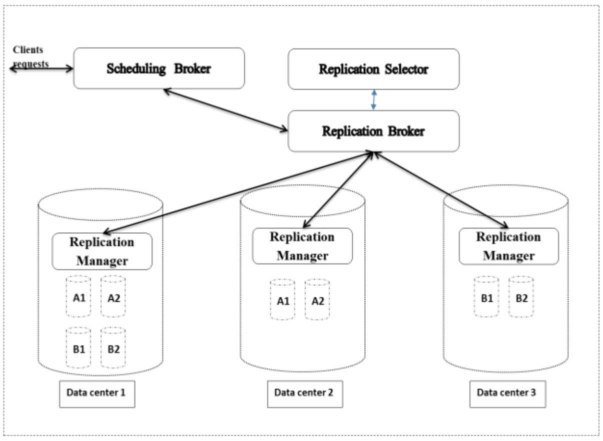

Fig. 1. The Cloud Data Server Architecture.

III. PROBLEM FORMALIZATION

A cloud data service system typically consists of the scheduling broker, replica selector, replica broker and data centers [5, 7, 8, 10-14], as shown in Fig. 1. The scheduling broker is the central managing broker. The replica managers hold the general information about the replica locations in data centers. The specific features of cloud data servers can be described as follows.

Let $U = \{u_1, u_2, \dots, u_m\}$ be m users at the Cloud, $TS = \{TS_1, TS_2, \dots, TS_m\}$ be a set of tasks of the user set U, and $TS_j = \{ts_{j1}, ts_{j2}, \dots, ts_{jm_j}\}$ be a subset of tasks of the *j*th user u_j, where m_j is the number of subtasks, and ts_k is the *k*th task submitted to the scheduling broker through a user interface and independent of other users. The replica broker schedules them to the appropriate cloud data server sites. If u_0 has two tasks, then $TS_0 = \{ts_{01}, ts_{02}\}$, and $m_0 = 2$. A task ts_k is characterized by a 4-tuple $ts_k = (tid_k, tr_k, td_k, tfn_k)$, where tid_k, tr_k, td_k and tfn_k are the task identification, task generation rate, task deadline time and the number of required files of task ts_k, respectively. For simplicity, we assume that the tasks are non-preemptable and non-interruptible [7, 11, 15], which mean that a task cannot be broken into smaller subtasks and it has to be executed as a whole using a single processor on the given resources. In addition, as soon as a task starts its execution on a processor, it cannot be interrupted and it occupies the processor until its execution completes successfully or a failure occurs.

Let $DC = \{dnd_1, dnd_2, \dots, dnd_n\}$ be a data center composed of n data nodes, which are running virtual machines on physical machines. A data node dnd_k is characterized by a 5-tuple $dnd_k = (dnd_k, dr_k, dst_k, df_k, dbw_k)$, where dnd_k, dr_k, dst_k, df_k **and** dbw_k are the data node identification, request arrival rate, average service time, failure probability and network bandwidth of data node dnd_k, respectively.

In order to guarantee the service performance of the data center *DC*, the task generation rate tr_k of user set *U*, the request arrival rate dr_k and failure probability df_k of DC should meet (1).

$$\sum_{j=0}^{\#subtasks} tr_j \le \sum_{i=0}^{n} dr_i \times (1 - df_i)$$
(1)

Where tr_j the task generation rate of task j is, dr_i is the request arrival rate of task *j* on the node *i*, df_k is the failure probability of task *j*.

Let $F = \{f_1, f_2, \dots, f_l\}$ be a data file set of a data center *DC*. $B = \{B_1, B_2, \dots, B_l\}$ be a set of blocks in the data center *DC*, **and** $B_i = \{b_{i1}, b_{i1}, \dots, b_{in1}\}$ be the *i*-th subset of blocks belonging to the *i*-th data file f_i, which is stripped into n_i fixed blocks according to its length. A block b_k is characterized by a 5-tuple $b_k = (bid_k ; bp_k ; bs_k ; bn_k ; bt_k)$, where bid_k, bp_k, bs_k, bn_k **and** bt_k are the block identification, number of requests, block size, the number of replicas and the last access time of block **bk**, respectively.

When user u_j requests a block b_k from a data node dnd_i with bandwidth performance guarantee, bandwidth $\frac{bs_k}{dst_i}$ should be assigned to this session. The total bandwidth used to support different requests from use set U should be less than dbw_i, as shown by (2).

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 1, Issue 6, June 2012

$$\sum_{i=0}^{s_i} \frac{bs_k}{dst_i} \leq dbw_i$$
(2)

Where s_i is the maximum number of network sessions of data node dnd_i that can serve concurrently, bs_k is the block size of block b_k, dst_i is the average service time of data node dnd_i, dbw_i is the network bandwidth of data node dnd_k .Block availability is the ability of a data block to provide proper service under given constraints. The block availability of a block b_k is denoted as BA_k. $P(BA_k)$ is the probability of block b_k in an available state. $\bar{P}(BA_k)$ is the probability of block b_k in an unavailable state, and $\bar{P}(BA_k) = 1 - P(BA_k)$. The number of replicas of block b_k is bn_k. It is obvious that block b_k is considered unavailable only if all the replicas of block b_k are not available. So the availability and unavailability of block b_k are calculated as 3 and 4.

$$P(BA_k) = 1 - (1 - P(ba_k))^{bn_k}$$
(3)

$$\bar{P}(BA_k) = (1 - P(ba_k))^{bn_k}$$
(4)

File availability is the ability of a data file to provide proper service under given constraints. The file availability of a data file f_i is denoted as FA_i. $P(FA_i)$ is the probability of data file f_i an available state. $P(FA_i)$ is the probability of data file f_i in an unavailable state, and $\bar{P}(FA_i) = 1 - P(FA_i)$.

If the data file f_i is stripped into n_i fixed blocks denoted by $B_i = \{b_{i1}, b_{i2}; \dots ; b_{in1}\}$, which are distributed on different data nodes. $N_i = \{bn_{i1}, bn_{i2}, \dots, bn_{ini}\}$ is the set of the numbers of replicas of the blocks of B_i. The availability and unavailability of data file f_i is given as follows:

$$P(FA_i) = \left(1 - (1 - P(ba_i))^{bn_i}\right)^{n_i}$$
(5)

If the data file f_i is stripped into n_i blocks, there are n_i replicas of each block in data file f_i, and all blocks at the same site will have the same available probability as all blocks are stored in data nodes with the same configuration in cloud data centers, the available probability of each replica is $p(ba_i)$ in data file f_i.

IV. Dynamic Data Replication Strategy

The proposed adaptive data replication has three important phases: 1) which data file should be replicated and when to replicate in the cloud system to meet users' requirements such as waiting time reduction and data access speeding up; 2) how many suitable new replicas should be created in the cloud system to meet a given availability requirement; 3) where the new replicas should be placed to meet the system task successful execution rate and bandwidth consumption requirements. The first step is to decide which data replicate and the replication timing. Given the fact that a more recently accessed data file might be accessed again in the near future according to the current status of data access pattern, a popular data file is determined by analyzing the access to the data from users. When the popularity of a data file passes a dynamic threshold, the replication operation will be triggered. Let pd_k be a popularity degree of a block b_k. pd_k is defined as the future access frequency based on the number of access demand, $an_k(t)$ at a time t, the popularity degree pd_k of a block b_k can be calculated using Holt's Linear and Exponential Smoothing (HLES).

Holt's Linear and Exponential Smoothing (HLES) is a computationally cheap time series prediction technique. HLES is selected for its capability of smoothing and providing short-term predictions for the measured requests arrival rates and service demand rates. Hence, HLES enables the proposed framework to monitor the arrival rates and service rates and to provide a short-term prediction for the future arrival rates and service rates with low computation time. Using these predictions, we can predict the utilization on each server host using equation (2), and predict the future response time of the web service using equation (1).

HLES smoothes the time series and provides a short-term forecast based on the trend which exists over the time series [16]. Suppose $an_k(t)$ is a time series value at time *t*. The linear forecast for the *m* steps ahead is as follows:

$$pd_k = an_k(t + m) = L_t + b_t \times m$$
(6)

Where L_t and b_t are exponentially smoothed estimates of the level and linear trend of the series at time *t*: $L_t = \alpha\, an_k(t) + (1 - \alpha)(L_{t-1} + b_{t-1})$

Where α is a smoothing parameter, $0 < \alpha < 1$, $b_t = \beta\,(L_t - L_{t-1}) + (1 - \beta)b_{t-1}$

Where β is a trend coefficient, $0 < \beta < 1$.A large value of α adds more weight to recent values rather than previous values in order to predict the next value. A large value of β adds more weight to the changes in the level than the previous trend. The replica factor is defined as the average of the ratio of the popularity degree and the average availability of replicas on the different data nodes of all blocks l of the data file f_i. It is used to determine whether the data file f_i should be replicated, denoted as

$$RF_i = \frac{1}{l}\sum_{k=1}^{l}\frac{\sum_{k=1}^{bn_i} pd_k \hat{P}(BA_k)}{bn_i} \quad (7)$$

Where $pd_k, \hat{P}(BA_k), l$ and bn_i are the popularity degree, the failure probability of a block b_k, number of blocks and number of replicas of data file f_i, respectively? In each time interval T, the replication operation of the data file f_i will be triggered if the replication factor RF_i is less than a specified threshold. The details of the proposed adaptive strategy are shown in Fig. 2.

```
• Initialize available and unavailable probability of
  each replica of block $b_k$, $p(ba_k) and \hat{p}(ba_k)$.
• for each data file $f_i$ at all data centers $DC$ do
    o Calculate the popularity degree $pd_k$ of a
      block $b_k$ of data file $f_i$
    o Calculate replica factor $RF_i$ of data file $f_i$
      .
    o If $RF_i$ is less than a threshold λ, trigger
      the replication for the file $f_i$
• end for
• for each triggered replication for data file $f_i$ do
    o for each block $b_k$ in the file $f_i$
        ▪ Calculate the new $RF_i$ by adding
          a replication on the each data
          center $dc_k$.
        ▪ Apply the replication which gives
          the highest new $RF_i$.
    o end for
• end for
• find the file $f_i$ which has the least $\sum_{k=1}^{l} pd_k$.
• for each replica in the file
    o delete the replica which gives the new,
      without the replica, $RF_i$ bigger than a
      threshold λ
• end for
```

Fig. 2. The Proposed Adaptive Replication Strategy.

IV. Simulation and Performance Evaluation

This section evaluates the effectiveness of the proposed adaptive replication strategy. The CloudSim framework is a Java based simulation platform for the Cloud environment, it supports modeling and simulation of large scale cloud computing data centers, including users and resources [17-19]. In the CloudSim simulation, 64 data centers are created with the corresponding topology shown in Fig. 1. The service providers are represented by 1000 virtual machines, and the processing elements (PEs) number of each virtual machine is within the range of 2 to 4. A hundred different data files are placed in the cloud storage environment, with each size in the range of [0.1, 10] GB. Each file is stored in fixed size (bs = 0:2 GB) storage unit called block. Blocks of the same data file are scattered across different virtual machines. 10000 tasks are submitted to the service providers using the Poisson distribution. Each task requires 1 or 2 data files randomly. Initially, the number of replicas of each data file is 1 and placed randomly. For simplicity, it is assumed that the basis element of data storage is block and the element of replication is one total data file. Fig 3 and Fig 4 are shown in Appendix. As shown in Fig. 3, with time elapses, the number of replicas is increasing within a very short period of time. Then, the number of replicas is maintained at a relatively stable level, which is determined by the adjustable parameter α in the HLES technique. We conclude that the greater the adjustable parameter α and the increasing block request bp_k of a certain file, the more replicas are needed to improve the file availability.

The response time for a data file is the interval between the submission time of the task and return time of the result. The average response time of a system is the mean value of the response time for all data request tasks of the users, which can be obtained by the following equation.

$$rt_{avg} = \frac{\sum_{j=1}^{m}\sum_{k=1}^{m_j}\left(ts_{jk}(rt) - ts_{jk}(st)\right)}{\sum_{j=1}^{m} m_j} \quad (8)$$

Where $ts_{jk}(st)$ and $ts_{jk}(rt)$ are the submission time and the return time of the result of task k of the user j, respectively, and m_j is the number of the tasks of user j. As shown in Fig. 4, with the number of tasks increasing and α = 0.7, the response time increases dramatically. The less the block availability, the longer the response time will be. It is clear that the proposed adaptive replication strategy enhances the response time and maintains the response time at a stable level within a short period of time.

V. Conclusions and Future Work

This paper proposes an adaptive replication strategy in the cloud environment. The strategy investigates the availability and efficient access of each file in the data center, and studies how to improve the reliability of the data files based on prediction of the user access to the blocks of each file. The proposed adaptive replication strategy redeploys dynamically large-scale different files replicas on different data nodes with minimal cost using heuristic search for each replication. The proposed adaptive strategy is based on a formal description of the problem. The strategy identifies the files which are popular file for replication based on analyzing the recent history of the data access to the files using HLES time series. Once a replication factor based on the popularity of the files is less than a specific threshold, the replication signal will be triggered. Hence, the adaptive strategy identifies the best replication location based on a heuristic search for the best replication factor of each file. Experimental evaluation demonstrates the efficiency of the proposed adaptive replication strategy in the cloud environment.

Future research work will focus on building a database system for 3D models. In fact, high quality 3D models are archived in huge files. These files are traditionally stored in distributed databases which suffer from answering visualization queries and traffic overloading on data centers. We aim to provide a framework for speeding up data access, and further increasing data availability for such databases on a cloud environment. Further, we will study using Genetic algorithms to find the best replication in less time. In addition, the replication strategy will be deployed and tested on a real cloud computing platform. Future work is also planned to provide the adaptive data replication strategy as a part of cloud computing services to satisfy the characteristics of cloud computing.

REFERENCES

[1] Buyya, R., et al., Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. Future Gener. Comput. Syst., 2009. **25**(6): p. 599-616.

[2] Armbrust, M., et al., A view of cloud computing. Commun. ACM, 2010. **53**(4): p. 50-58.

[3] Ghemawat, S., H. Gobioff, and S.-T. Leung, The Google file system. SIGOPS Oper. Syst. Rev., 2003. **37**(5): p. 29-43.

[4] Shvachko, K., et al., The Hadoop Distributed File System, in Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST). 2010, IEEE Computer Society. p. 1-10.

[5] Chang, R.-S. and H.-P. Chang, A dynamic data replication strategy using access-weights in data grids. J. Supercomput., 2008. **45**(3): p. 277-295.

[6] Wei, Q., et al., CDRM: A cost-effective dynamic replication management scheme for cloud storage cluster., in 2010 IEEE International on Cluster Computing. 2010. p. 188 - 196

[7] Bonvin, N., T.G. Papaioannou, and K. Aberer, A self-organized, fault-tolerant and scalable replication scheme for cloud storage, in Proceedings of the 1st ACM symposium on Cloud computing. 2010, ACM: Indianapolis, Indiana, USA. p. 205-216.

[8] Nguyen, T., A. Cutway, and W. Shi, Differentiated replication strategy in data centers, in Proceedings of the 2010 IFIP international conference on Network and parallel computing. 2010, Springer-Verlag: Zhengzhou, China. p. 277-288.

[9] Dogan, A., A study on performance of dynamic file replication algorithms for real-time file access in Data Grids. Future Gener. Comput. Syst., 2009. **25**(8): p. 829-839.

[10] Wang, S.-S., K.-Q. Yan, and S.-C. Wang, Achieving efficient agreement within a dual-failure cloud-computing environment. Expert Syst. Appl., 2011. **38**(1): p. 906-915.

[11] McKusick, M.K. and S. Quinlan, GFS: Evolution on Fast-forward. Queue, 2009. **7**(7): p. 10-20.

[12] Lei, M., S.V. Vrbsky, and X. Hong, An on-line replication strategy to increase availability in Data Grids. Future Gener. Comput. Syst., 2008. **24**(2): p. 85-98.

[13] Jung, D., et al., An effective job replication technique based on reliability and performance in mobile grids, in Proceedings of the 5th international conference on Advances in Grid and Pervasive Computing. 2010, Springer-Verlag: Hualien, Taiwan. p. 47-58.

[14] Yuan, D., et al., A data placement strategy in scientific cloud workflows. Future Generation Computer Systems, 2010. **26**(8): p. 1200-1214.

[15] Litke, A., et al., A Task Replication and Fair Resource Management Scheme for Fault Tolerant Grids Advances in Grid Computing - EGC 2005, P. Sloot, et al., Editors. 2005, Springer Berlin / Heidelberg. p. 482-486.

[16] Makridakis, S.G., S.C. Wheelwright, and R.J. Hyndman, eds. Forecasting: Methods and Applications, 3rd Edition. 1998.

[17] Calheiros, R.N., et al., CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw. Pract. Exper., 2011. **41**(1): p. 23-50.

[18] Wickremasinghe, B., R.N. Calheiros, and R. Buyya, CloudAnalyst: A CloudSim-Based Visual Modeller for Analysing Cloud Computing Environments and Applications, in Proceedings of the 2010 24th IEEE International Conference on Advanced Information Networking and Applications. 2010, IEEE Computer Society. p. 446-452.

[19] Xu, B., et al., Job scheduling algorithm based on Berger model in cloud environment. Adv. Eng. Softw., 2011. **42**(7): p. 419-425.

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 1, Issue 6, June 2012

APPENDIX

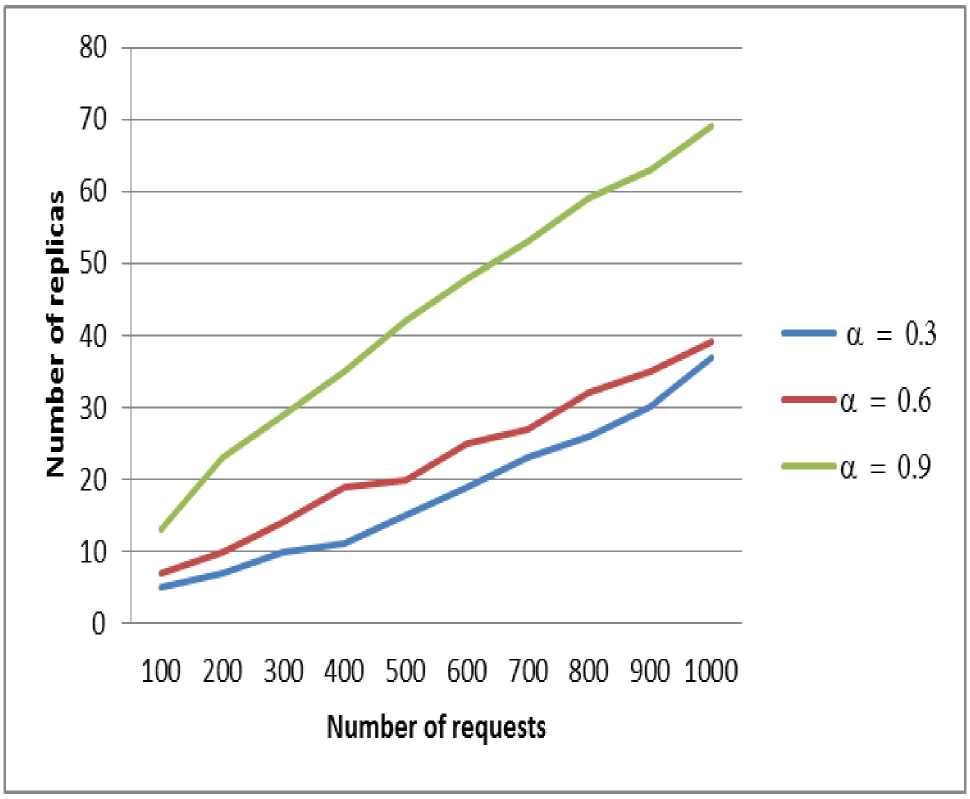

Fig. 3. Number of Replicas with Increasing Parameter α of HLES and Increasing Requests.

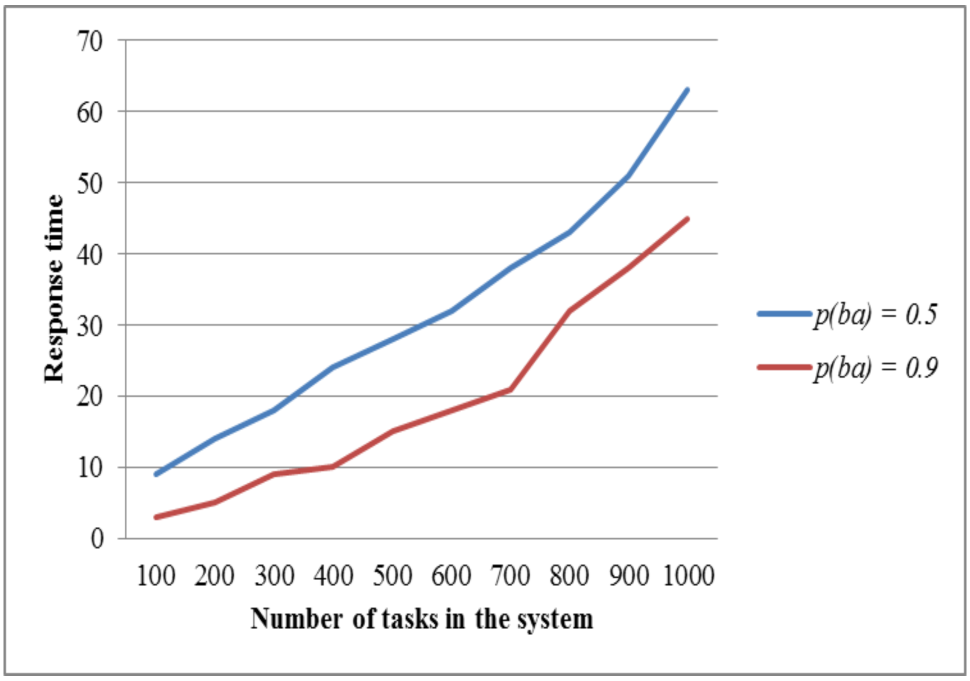

Fig. 4. The Response Time versus the Number of Tasks Using Different Probabilities.