Al-Aqsa University Journal (Natural Sciences Series),Vol.20, Pages 1- 14,Jan.2016
ISSN 2070-3155

Fussy logic replica selection algorithm for cloud storage system

Dr. Mohammed A Radi *

الملخص

خوارزمية منطق ضبابي لإختيار النسخة في نظام التخزين السحابي

في السنوات الاخيرة حاز نظام التخزين السحابي على اهتمام ملحوظ كاتجاه جديد في إدارة البيانات. تكرار البيانات يستخدم لتعزيز المرونة الوثوقية، وزمن الإستجابة. تقنيات إختيار النسخة تستخدم لاختيار النسخة الأنسب للرد علي طلبات المستخدمين وتقليل زمن الإستجابة للمستخدمين. في هذا البحث تم اقتراح خوارزمية منطق ضبابي لاختيار النسخة في نظام التخزين السحابي وذلك لاختيار أفضل مركز بيانات لتحسين أداء النظام. تم تنفذ الخوارزمية المقترحة وتقيمها باستخدامcloud−Analyst simulator بالمقارنة مع الخوارزميات الأخرى، فالخوارزمية المقترحة تقلل زمن الاستجابة وتظهر تحسن ملحوظ في أداء النظام.

Abstract

In recent years, cloud storage systems have received considerable attention as a new trend in data management. Data replication is utilized to enhance scalability, reliability, and response time. Replica selection techniques are used to choose the most suitable replica to respond to user requests and minimize user response time. In this paper, a fussy logic replica selection algorithm for cloud storage system is proposed to determine the best data center and to improve system performance. The proposed algorithm is implemented and evaluated using a CloudAnalyst simulator. Compared with other algorithms in literature, the proposed fussy logic algorithm, as shown in simulation results, reduced overall response time and exhibited remarkable improvement in system performance.

Keywords: cloud storage, data replication, replica selection, fussy algorithm.

* **computer science department, Alaqsa university- Gaza**

1. Introduction:

Cloud computing is becoming increasingly popular as a new data management trend (Elzeiny, Elfetouh et al. 2013). Within clouds, data replication has been introduced and used widely to store copies of data sets (files or subsets of a file) across different data centers (DCs). In cloud data storage, data replication is used to ensure timely, reliable, and efficient data access thereby improving cloud storage system response time (Wu, Ping et al. 2010). Data replication specifically limits access latency by allowing users to access nearby replicas and enhances availability by providing access to data even when some replicas are unavailable. Finally, data replication improves cloud system performance by distributing user requests among different DCs. A cloud storage system with multiple replicas requires efficient management mechanisms for replica placement, replica selection, and replica consistency. A suitable replica selection is one of the most critical issues related to replica management in a cloud data storage system. A good replica selection has to choose the most suitable replica to respond to cloud user requests because a cloud storage system has multiple replicas.

Traditional non-adaptive replica selection algorithms such as random and round robin do not always guarantee the best performance. The closest server algorithm and greedy algorithm are not always the best because each algorithm only considers one factor in choosing the replica server, i.e., distance for the closest server algorithm and processing capacity for greedy algorithm. In reality, factors such as distance, server speed, available bandwidth, and server response time also have major effects on suitable replica selection (Oliveira and Fernandez 2013; Zhang, Wang et al. 2014).

This paper discusses method for selecting suitable data replica to respond to user access requests, which requires an efficient management mechanism to determine the suitable replica. In this paper, we propose a new replica selection algorithm based on fussy logic to choose the best DC. This algorithm considers the following parameters: (1) the network delay between the user and server and (2)the last processing time at the server. Performance evaluation was conducted using CloudAnalyst simulator architecture(Wickremasinghe, Calheiros et al. 2010). The proposed algorithm is compared with four different replica selection algorithms, including round robin, random, closets replica, and greedy. The metric used

for the evaluation is the response times. Simulation results showed that fussy algorithm reduced the response time better than algorithms in literature.

The rest of the paper is structured as follows. Section 2 presents some related works. Section 3 introduces the replica management framework. Section 4 presents our proposal, i.e., the Fussy replica selection algorithm. Section 5 shows the evaluation framework and experiment setting. Section 6 contains the overview of the results and discussion. Finally, Section 7 concludes the paper.

2. Related works:

Replica selection is one of traditional problems in a distributed system. One of the oldest replica selection algorithm is the round robin (RR) algorithm (Xu and Huang 2009). In RR, the user request is forwarded to replica servers following a cyclic order. This algorithm performs well in a homogeneous environment, i.e., the servers have similar capacities, are located in the same place, share the same network, and the requests generate a similar workload. However, in the case of heterogeneous environment and the replicas are distributed at different places, the RR algorithm gives bad results. The random replica selection algorithm is another classical example (Motwani and Raghavan 2010). In random replica selection algorithm, the replica server will always be chosen at a random manner among replica servers. Generally, predicting the performance of random algorithms is impossible because of variations in the results. A greedy algorithm called least loaded redirects the request to the server with the lowest load(Dahlin 2000). Least loaded routing algorithm is an algorithm that attempts to distribute the requisition to servers with the largest idle capacity, i.e., the least loaded server. The least loaded server can be determined by monitoring the current server state via a protocol [14].

Several replica selection algorithms have been proposed recently for cloud environments. Bai (Bai, Jin et al. 2013) proposed a replica selection method for cloud data storage based on load process capability(LPC). The metrics of LPC consists of three components: CPU process capability, network transmission capability, and I/O capability of disks. The replica selection method chooses the node with the highest LPC to respond to a user request and then the user accesses the file from the node with the highest

LPC. The performance of the proposed replica selection method is not revealed fully in the simulation. Bang Zhang (Zhang, Wang et al. 2014) proposed a data replica selection scheme for cloud storage called plant growth simulation algorithm (PGSA). PGSA is an intelligent optimization algorithm that uses a plant phototropism mechanism. PGSA takes into account network status, performance, load of storage node, and historical information to improve average access time and replica utilization. In (Bingxiang and Kui 2010), an adaptive replica selection strategy was presented to oncurrent data transfers. In (Cheng, Wang et al. 2008), based on the min–max balancing workload method, a dynamic replica selection strategy was proposed to upgrade the efficiency of execution in data grid environments. In (Almuttairi, Wankar et al. 2010), a new replica selection strategy was presented. The proposed strategy used the concept of association rules of data mining approach to the most stable network segments, and could adapt its replica selection criteria dynamically to satisfy user access requirements and reduce system response time. In(Sun, Sun et al. 2007), an ant-algorithm-based replica selection scheme was proposed. The calculation formula of the probability of a replica being selected was devised, and candidate replicas predicted based on historical replica accessed records. The formula could help balance the access loads among replicas dynamically.

Fussy logic is an extension of Boolean logic where fussy logic replaces Boolean truth values with the degree of truth. The degree of truth is often employed to capture imprecise modes of reasoning. The degree of truth plays an essential role in the human ability to make decisions in an environment of uncertainty and imprecision. Fussy inference system (FIS) derives answers from a knowledge base by using a fussy inference engine(McNeill 1994). The inference engine provides methodologies for reasoning around the information in the knowledge base and formulating the results. Some researchers have been using fussy logic efficiently in replica management. Thiago Queiroz (Oliveira and Fernandez 2013) proposed a new adaptive algorithm based on fussy logic to choose the best replica server. This algorithm considers the following parameters: (1) size of the service queue for each replica server; (2) time required to answer a request from a given URL on the replica server; and (3) response time of replica server [round-trip time (RTT)]. Toukir Imam in (Imam and Rahman 2011)

presented a replica replacement algorithm based on fussy logic by building a fussy rule base called Fussy12 algorithm and implementing it on GridSim simulator.

3. Replica management framework

In this section, we will provide a brief description of the general architecture of cloud data storage. Cloud storage is composed of numerous storage devices connected or combined through networks, various distributed file systems, and a few middleware storage services to establish a secure and reliable cloud storage facility. In this study, we consider a scenario in which several DCs are distant from one another in different regions (R) or even on dissimilar continents. Each DC encapsulates a set of computing or storage hosts that are either heterogeneous or homogeneous based on their hardware configurations. The user base (UB) models a group of users and generates a representative cloudlet. This cloudlet specifies a set of user requests containing an application ID, the name of the UB as the originator to which responses can be returned, request execution commands (in terms of size), and the target logical file name. Any DC can serve the cloudlet only if the DC contains a replica of the target file.

We integrate the CloudAnalyst simulator architecture(Wickremasinghe, Calheiros et al. 2010) with three main algorithm components, namely, replica catalog, replica selection, and replica selection algorithm to manage replicas efficiently and provide the required functionalities. The replica catalog contains information on the locations of files and associated replicas, as well as the metadata associated with these files. Replica selection handles inquiries from replica catalog and runs the replica selection algorithm. As indicated in Figure 1, the Internet requests the target DC controller from the replica selection as soon as the UB generates an Internet cloudlet. Replica selection inquires with the replica catalog on the DCs holding the target file. The replica catalog then returns a list of such DCs. Replica selection applies a replica selection algorithm to this list to return the target DC controller to the Internet. The Internet then sends the requests to the selected DC controller based on this information. The selected DC controller processes the requests using a virtual machine (VM) load balancer before sending the response to the Internet.

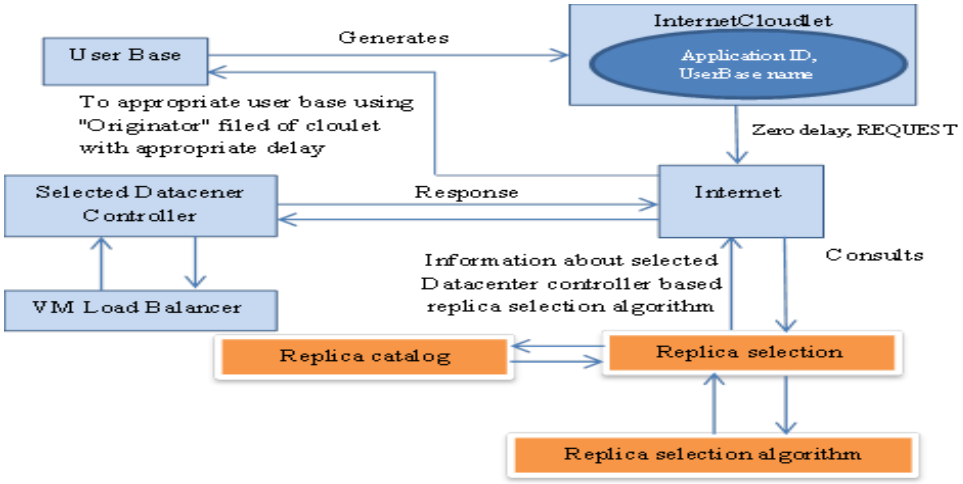

Figure 1: Integrated framework to route user requests in CloudAnalyst

4. Fussy replica selection algorithm:

This work proposes an adaptive algorithm in selecting the appropriate data replica to respond to user access requests based on fussy logic (McNeill 1994). Fussy logic algorithm considers delay and processing time as two input parameters. The delay is the current network delay between the user and DC, whereas processing time is the last recorded processing time for DC. The proposed algorithm follows the FIS mechanism (Oliveira and Fernandez 2013), which has three phases, including fuzzification, rule evaluation, and defuzzification.

4.1 Fuzzification:

In this step, mapping of input parameters for fussy sets, which are generally numerical and accurate, is realized using the membership functions. The input membership functions are triangular, and have the same characteristics within the function range. The delay variable has five linguistic values associated, including VLow, Low, Medium, High, and VHigh. Figure 2 shows their membership functions and respective ranges. The processing time variable also has five membership functions associated, including VSmall, Small, medium, High, and Vhigh. Figure 3 indicates the membership function. The output consists of three membership functions using triangular functions (VGood, Good, Bad), as shown in Figure 4.

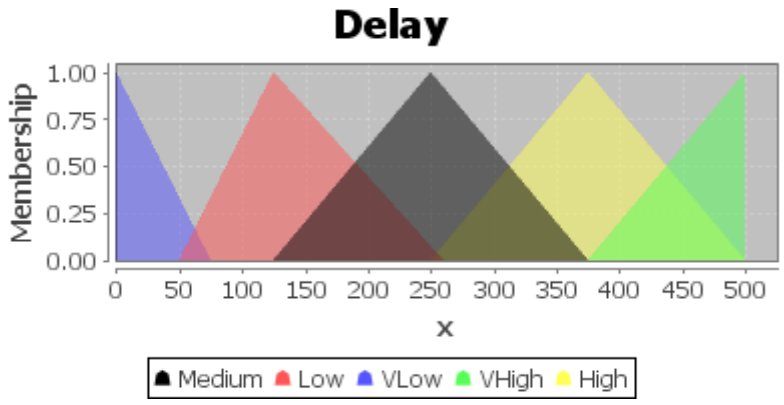

Figure 2: Delay membership functions

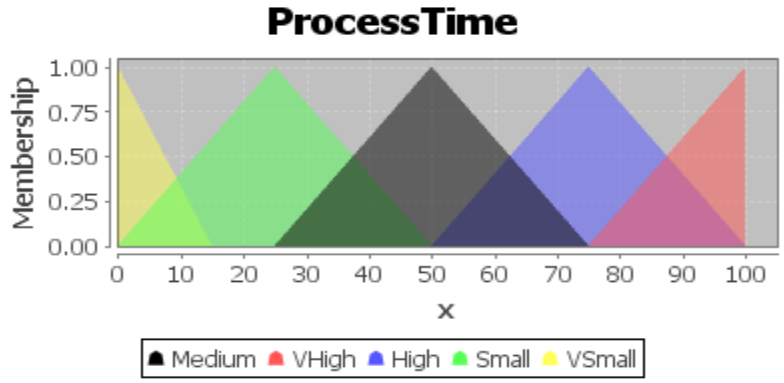

Figure 3: Process time membership functions

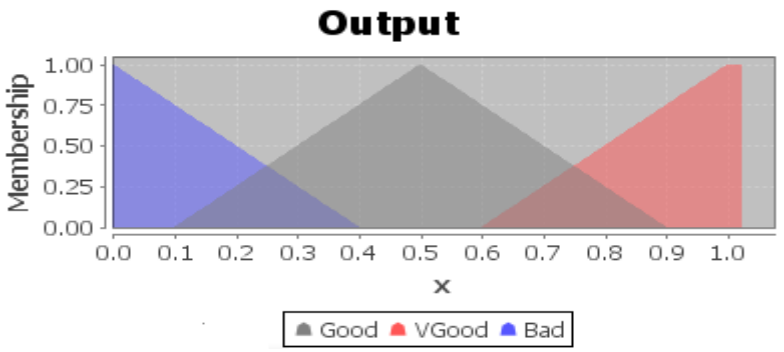

Figure 4: Output membership functions

4.2 Rule evaluation:

Defining the fussy controller inference rule is not a simple task because it can produce inconsistent results(Oliveira and Fernandez 2013). In FISs, the number of rules has direct effect on its time complexity. We produced 25 rules for our proposed system. The proposed rules are mentioned in Figure5.

```
RULE 1  : IF Delay IS VLow AND ProcessTime IS VSmall THEN Output IS Excellent;
RULE 2  : IF Delay IS VLow AND ProcessTime IS Small THEN Output IS Excellent;
RULE 3  : IF Delay IS VLow AND ProcessTime IS Medium THEN Output IS Excellent;
RULE 4  : IF Delay IS VLow AND ProcessTime IS High THEN Output IS VGood;
RULE 5  : IF Delay IS VLow AND ProcessTime IS VHigh THEN Output IS Good;
RULE 6  : IF Delay IS Low AND ProcessTime IS VSmall THEN Output IS Excellent;
RULE 7  : IF Delay IS Low AND ProcessTime IS Small THEN Output IS Excellent;
RULE 8  : IF Delay IS Low AND ProcessTime IS Medium THEN Output IS VGood;
RULE 9  : IF Delay IS Low AND ProcessTime IS High THEN Output IS Good;
RULE 10 : IF Delay IS Low AND ProcessTime IS VHigh THEN Output IS Good;
RULE 11 : IF Delay IS Medium AND ProcessTime IS VSmall THEN Output IS VGood;
RULE 12 : IF Delay IS Medium AND ProcessTime IS Small THEN Output IS VGood;
RULE 13 : IF Delay IS Medium AND ProcessTime IS Medium THEN Output IS Good;
RULE 14 : IF Delay IS Medium AND ProcessTime IS High THEN Output IS Good;
RULE 15 : IF Delay IS Medium AND ProcessTime IS VHigh THEN Output IS Bad;
RULE 16 : IF Delay IS High AND ProcessTime IS VSmall THEN Output IS Good;
RULE 17 : IF Delay IS High AND ProcessTime IS Small THEN Output IS Bad;
RULE 18 : IF Delay IS High AND ProcessTime IS Medium THEN Output IS Bad;
RULE 19 : IF Delay IS High AND ProcessTime IS High THEN Output IS Bad;
RULE 20 : IF Delay IS High AND ProcessTime IS VHigh THEN Output IS Bad;
RULE 21 : IF Delay IS VHigh AND ProcessTime IS VSmall THEN Output IS Good;
RULE 22 : IF Delay IS VHigh AND ProcessTime IS Small THEN Output IS Bad;
RULE 23 : IF Delay IS VHigh AND ProcessTime IS Medium THEN Output IS Bad;
RULE 24 : IF Delay IS VHigh AND ProcessTime IS High THEN Output IS Bad;
RULE 25 : IF Delay IS VHigh AND ProcessTime IS VHigh THEN Output IS Bad;
```

Figure 5: Inference rules used

4.3 Defuzzification:

Defuzzification is defined in the literature as the process of producing a quantifiable result in fussy logic, given fussy sets, and corresponding membership degrees. The process converts a number of variables into a fussy result using a number of rules and the result is described in terms of membership in fussy sets. The defuzzification process is integrated in the proposed algorithm.

4.4 Proposed algorithm:

Algorithm 1 shows the controller operation in the decision-making process. The algorithm obtains the user request as input and gives the target DC as output. In line 3, the algorithm obtains the DC list for a given replica. Then, from lines 4 to 11, the algorithm calculates the fussy value for each DC using delay and processing times as input parameters. The parameter values are then mapped to fussy sets according to their previously defined membership functions. After this step, the inference rules are applied, resulting in a fussy set output. Finally, defuzzifaction returns a quantitative value using the center area method. The algorithm returns the target DC with the highest fussy value.

Algorithm 1: Fussy replica selection algorithm

1 $Fvalue_{max}$ → Null
2 TargetDC → Null
3 For all DC have the replicas
4 $Delay_i$ → the current network delay between the user and the data center.
5 Processing $time_i$ →: data center last Recorded Processing Time
6 Fvalue → calculate the fussy value($Delay_i$, Processing $time_i$)
7 if (Fvalue > $Fvalue_{max}$) or ($Fvalue_{max}$ == Null)
8 $Fvalue_{max}$ → Fvalue
9 TargetDC → DC
10 End
11 End
12 return TargetDC

5. Performance evaluation:

The proposed algorithm is evaluated using the CloudAnalyst simulator(Wickremasinghe, Calheiros et al. 2010).For our FIS, we used an open-source Java library called jFussyLogic (jFussyLogic), which offers a fully functional and complete implementation of FIS. We ran the simulator under configurations for user based, DC, and other simulation parameters, which are presented in Tables 1, 2, and 3 respectively.

Table 1: UB configuration

UB Name	Region	Requests per users per hr	Peak hrs (GMT)	Avg. users	Avg. off-peak users
UB1	0	60	3:00—9:00	47000	8000
UB2	1	60	3:00—9:00	60000	11000
UB3	2	60	3:00—9:00	35000	6500
UB4	3	60	3:00—9:00	80000	12500
UB5	4	60	3:00—9:00	11500	1200
UB6	5	60	3:00—9:00	150000	305000

Table 2: DC configuration

Name	Region	#VMs	Cost/VM ($/Hr)	Data transfer cost ($/Gb)	Number of processors	Speed (MIPS)
DC1	0	5	0.1	0.1	4	10000
DC2	2	5	0.1	0.1	4	12000
DC3	4	5	0.1	0.1	4	15000

Table 3: Other simulation parameters

Parameter	Value
User grouping factor in UB	1000
Request grouping factor	10
Executable instruction length/request	500
Load balancing policy	Round-Robin
Replica selection algorithm	Closest data center
Simulation duration	24 hr
VM image size	10000
VM memory	512 Mb
VM bandwidth	1000
DC architecture	X86
DC processor/machine	4
DC OS	Linux

The characteristics of the Internet, which includes Internet latency and bandwidth, are configured in Tables 4 and 5, respectively.

Table 4: Internet latency

Region/ Region	0	1	2	3	4	6
0	25.0	100.0	150.0	250.0	250.0	100.0
1	100.0	25.0	250.0	500.0	350.0	200.0
2	150.0	250.0	25.0	150.0	150.0	200.0
3	250.0	500.0	150.0	25.0	500.0	500.0
4	250.0	350.0	150.0	500.0	25.0	500.0
5	100.0	200.0	200.0	500.0	500.0	25.0

Table 5: Internet bandwidth

Region/ Region	0	1	2	3	4	6
0	2000.0	1000.0	1000.0	1000.0	1000.0	1000.0
1	1000.0	800.0	1000.0	1000.0	1000.0	1000.0
2	1000.0	1000.0	2500.0	1000.0	1000.0	1000.0
3	1000.0	1000.0	1000.0	1500.0	1000.0	1000.0
4	1000.0	1000.0	1000.0	1000.0	500.0	1000.0
5	1000.0	1000.0	1000.0	1000.0	1000.0	2000.0

6. Results and discussion:

Overall response time is the main metric for evaluating the proposed algorithm, and this metric is compared with that of other replica selection algorithms. Overall response time is defined as the time between user request generation and response derivation. Figure 6 show the results for the five algorithms (round robin, random, closest, greedy, and fussy). For each algorithm, we obtained the minimum, average, and maximum overall response time. As shown in Figure 6, all algorithms recorded nearly the same minimum response time, while fussy algorithm recorded the best minimum response time. In terms of average response time, the proposed fussy algorithm obtained the best average response time with a clear difference compared with the other algorithms, whereas greedy algorithm recorded the worst average response time. The random algorithm recorded

the best maximum response time with narrow difference with that of the fussy algorithm.
In general, we noted that the proposed fussy logic algorithm yielded good results as a replica selection algorithm in a cloud data storage environment. The good average response time of the proposed fussy algorithm enables the deployment of the proposed algorithm in real cloud data storage environment.

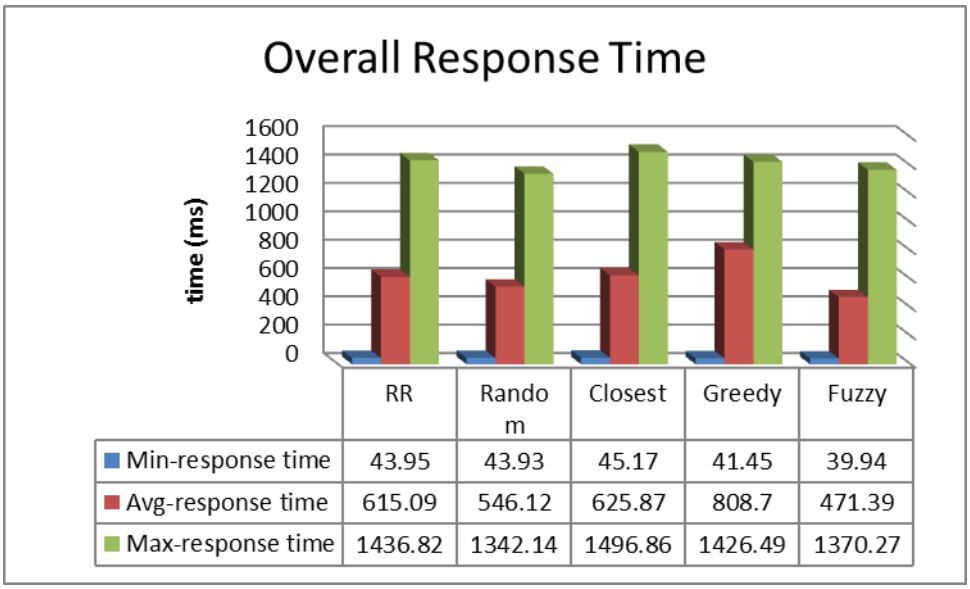

Figure 6: Overall Response Time

7. Conclusions:

In this paper, we proposed a fussy logic replica selection algorithm that selects the suitable data replica to respond to user access requests in a cloud storage system to improve system performance. The proposed algorithm was implemented and evaluated using a CloudAnalyst simulator. The simulation results showed that the proposed fussy logic algorithm reduced the overall response time better than the other algorithms in literature. The proposed algorithm contributed to a remarkable improvement in the system performance. For future works, other variables can be considered as input parameters for FIS and other metrics algorithm can be analyzed.

8. References:

Almuttairi, R. M., R. Wankar, et al. (2010). New replica selection technique for binding replica sites in Data Grids. Energy, Power and Control (EPC-IQ), 2010 1st International Conference on, IEEE.

Bai, X., H. Jin, et al. (2013). RTRM: A Response Time-Based Replica Management Strategy for Cloud Storage System. Grid and Pervasive Computing, Springer: 124-133.

Bingxiang, G. and Y. Kui (2010). A Global Dynamic Scheduling with Replica Selection Algorithm Using GridFTP. Challenges in Environmental Science and Computer Engineering (CESCE), 2010 International Conference on, IEEE.

Cheng, K.-Y., H.-H. Wang, et al. (2008). Dynamic file replica location and selection strategy in data grids. Ubi-Media Computing, 2008 First IEEE International Conference on, IEEE.

Dahlin, M. (2000). "Interpreting stale load information." Parallel and Distributed Systems, IEEE Transactions on 11(10): 1033-1047.

Elzeiny, A., A. A. Elfetouh, et al. (2013). "Cloud Storage: A Survey." International Journal of Emerging Trends & Technology in Computer Science (IJETTCS) 2(4): 342-349.

Imam, T. and R. M. Rahman (2011). Implementation and Performance Analysis of Fussy Replica Replacement Algorithm in Data Grid. Computer and Information Science 2011, Springer: 95-110.

jFussyLogic. "Open source fussy logic library and fcl language implementation.", 2015, from http://jfussylogic.sourceforge.net/html/index.html.

McNeill, F. M. a. E. T. (1994). Fussy Logic A Practical Approach. United Kingdom, Academic Press Limited.

Motwani, R. and P. Raghavan (2010). Randomized algorithms, Chapman & Hall/CRC.

Oliveira, T. and M. Fernandez (2013). Fussy Redirection Algorithm for Content Delivery Network (CDN). ICN 2013, The Twelfth International Conference on Networks.

Sun, M., J.-Z. Sun, et al. (2007). "Research of replica selection scheme based on ant algorithm in data grid." Jisuanji Gongcheng yu Yingyong(Computer Engineering and Applications) 43(1): 145-147.

Wickremasinghe, B., R. N. Calheiros, et al. (2010). Cloudanalyst: A cloudsim-based visual modeller for analysing cloud computing environments and applications. Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on, IEEE.

Wu, J., L. Ping, et al. (2010). Cloud storage as the infrastructure of cloud computing. Intelligent Computing and Cognitive Informatics (ICICCI), 2010 International Conference on, IEEE.

Xu, Z. and R. Huang (2009). "Performance study of load balancing algorithms in distributed web server systems." CS213 Parallel and Distributed Processing Project Report 1.

Zhang, B., X. Wang, et al. (2014). A PGSA Based Data Replica Selection Scheme for Accessing Cloud Storage System. Advanced Computer Architecture, Springer: 140-151.