BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **15**, No 6
Special Issue on Logistics, Informatics and Service Science

Sofia • 2015 Print ISSN: 1311-9702; Online ISSN: 1314-4081
DOI: 10.1515/cait-2015-0072

Distributed Consistency Method and Two-Phase Locking in Cloud Storage over Multiple Data Centers

Fan Lin, Jianbin Xiahou, Qihua Huang

Software School, Xiamen University, Xiamen, China
Emails: iamafan@xmu.edu.cn jbxiahou@xmu.edu.cn 23020111153052@stu.xmu.edu.cn

Abstract: *In replication management under multi-datacenter cloud storage environments, the problem of a replica being inconsistent across multiple datacenters must be solved. Focusing on data management strategies under multiple datacenters, this paper presents a design of a HDFS-based two Cascading Master-Slave Storage System (2CMSSS), which simplifies the research work of the replication management strategies under multi-datacenter cloud storage environments. On this basis a two-phase locking maintenance strategy for replica consistency is presented to validate and fix the problem of replica inconsistency across multiple datacenters. According to the experiment simulation, this strategy can ensure strong data consistency and maintain low latency for users to access data.*

Keywords: *Multiple datacenters; replication management; Two Cascading Master-Slave Storage System (2CMSSS); AHP-backward cloud; two-phase locking.*

1. Introduction

In distributed storage systems based on cloud computing, data replication management technologies are the key to realizing reliability and security [1]. However, they are often limited to a single datacenter model in the current cloud storage systems while neglecting the presence of multiple datacenters across the regions, such as service selection of a datacenter, replication layout across datacenters, as well as maintenance of the replication consistency [2, 3]. Therefore, after exploring the Hadoop Distributed File System (HDFS) [4], storage distribution strategies across multiple datacenters, as well as the traditional policies of distributed data consistency, this paper has put forward a multi-data cloud storage framework based on a Two Cascading Master-Slave Storage System (2CMSSS), which simplifies the research work of the replication management strategies under multi-datacenter cloud storage environments. In addition, to maintain the consistency of a replica in a cloud storage over multiple datacenters, a two-phase locking mechanism is designed on the 2CMSSS framework to realize improved consistency maintenance strategy across the datacenters.

2. 2CMSSS framework

In cloud storage over multiple datacenters, the storage nodes are connected to datacenters, thus constituting a whole hierarchical mesh storage system [5, 6]. Hence, in cloud storage over multiple datacenters, the replication management strategies must be divided into two different levels of inter-data and intra-datacenters [7]. From the perspective of its development trend and load balancing of the cloud storage over multiple datacenters, for disaster recovery and improved access rate to the data, it has become essential to design a model of a multi-datacenter cloud storage system model.

Aiming to establish two levels of the replication management in cloud storage over multiple datacenters, this paper designs the HDFS-oriented 2CMSSS cloud storage model architecture, which has simplified replication management strategies under multi-datacenter cloud storage environments and has set the foundation for replication management strategies for multiple datacenters in subsequent researches [8-10]. A two-phase locking strategy for maintaining the replication consistency is designed on this basis.

2.1. Storage model

The design of a 2CMSSS storage system model has not only met the management and storage needs of ultra-large-scale data clusters relying on autonomous datacenters and favourable characteristics of HDFS, but has also provided a particular object of study for the replication management strategies in a cloud storage over multiple datacenters and has laid the foundation for subsequent researches of replication management strategies [11].

The multi-datacenter cloud storage system is composed of a plurality of distributed datacenters [12]. The set of these distributed datacenters is denoted as $\mathrm{MDS} = \bigcup_{i=1,2,\dots,|\mathrm{MDS}|}\{\mathrm{dc}_i\}$, where dc_i datacenter is numbered *i*. Let

(1) $$2\mathrm{CMSSS} = \{\mathrm{MDS}, \mathrm{MM}\},$$

presents the 2CMSSS multi-datacenter storage system designed in this paper, where

(2) $$\mathrm{MM} = \begin{pmatrix} \mathrm{mmID}, \mathrm{mmIP}, \mathrm{StorageInfo}, \mathrm{IoInfo}, \mathrm{CpuInfo}, \\ \mathrm{MemInfo}, \mathrm{GFNS}, \mathrm{CmInfo}, \mathrm{otherInfo} \end{pmatrix},$$

represents the Main Master of 2CMSSS. It is the master server of the entire multi-datacenter cloud storage system, which is equivalent to the NameNode in HDFS; where mmID and mmIP represent the ID number and IP address of the main master, respectively

(3) $$\mathrm{StorageInfo} = (\mathrm{StorageTotal}, \mathrm{StorageUsed}, \mathrm{StorageUsage}),$$

(4) $$\mathrm{IoInfo} = (\mathrm{IoTotal}, \mathrm{IoUsed}, \mathrm{IoUsage}),$$

(5) $$\mathrm{CpuInfo} = (\mathrm{CpuTotal}, \mathrm{CpuUsed}, \mathrm{CpuUsage}),$$

(6) $$\mathrm{MemInfo} = (\mathrm{MemTotal}, \mathrm{MemUsed}, \mathrm{MemUsage}),$$

(7) $$\mathrm{CmInfo} = \cup_{i=1,2,\ldots,|\mathrm{MDS}|}\{\mathrm{CM}_i\}.$$

The storage space, IO throughput, CPU, memory, and client master information of the main master in the 2CMSSS storage system are shown respectively. GFNS (Global File Name Space) denotes a file or data object namespace of the entire multi-datacenter cloud storage system. It is expressed as $\mathrm{GFNS} = \cup_{i=1,2,\ldots,|\mathrm{MDS}|}\{\mathrm{dcFNS}_i\}$, where dcFNS_i is the namespace of the files or data objects in DC_i and its property will be given later, while otherInfo is used to reserve information bits or store other information by the main master.

Based on HDFS's client-server architecture and the autonomy of the datacenter, let the set $S_i = \cup_{i=1,2,\ldots,|\mathrm{dc}_i|}\{S_{ij}\}$ denote the storage node-set in the datacenter DC_i and let DC_i be denoted as:

(8) $$\mathrm{DC}_i = \{S_i \mathrm{CM}_i\},$$

where DC_i is a local HDFS system whose system structure, data reading/writing mechanism and heartbeat mechanism are the same as HDFS, while the replication strategy is improved in combination with the relevant work

(9) $$S_{ij} = (i, j, \mathrm{StorageInfo}, \mathrm{IoInfo}, \mathrm{CpuInfo}, \mathrm{MemInfo}, \mathrm{Blocks}),$$

indicates the j-th storage node in data center DC_i, and $\mathrm{Blocks} = \cup_{i=1,2,\ldots}\{\mathrm{block}_m\}$ represent the data blocks stored in the node.

(10) $$\mathrm{DC}_i = (i, \mathrm{cmID}, \mathrm{cmIP}, \mathrm{StorageInfo}, \mathrm{IoInfo}, \mathrm{CpuInfo}, \mathrm{MemInfo}, \mathrm{metaInfo}, \mathrm{otherInfo}),$$

where CM_i represents the client master in the i-th datacenter DC_i and NameNode in DC_i. cmID and cmIP are the ID number and IP address of CM_i; StorageInfo, IoInfo, CpuInfo and MemInfo are shown respectively in Equations (3)-(6), and have similar meanings, while otherInfo is used to reserve information bits or store other information by CM_i and

(11) $$\mathrm{metaInfo} = (\mathrm{BandWidth}, \mathrm{StorageInfo}, \mathrm{FailureTime}, \mathrm{dcFNS}_i, \mathrm{SiInfo}, \mathrm{otherInfo}),$$

represents metadata information stored in CM_i and it is metadata information in the whole DC_i, BandWidth represents the network bandwidth of DC_i. Due to the dynamic nature of the inter-datacenter network environment, the network bandwidth in the general sense has no practical significance. In this paper the network latency value of the datacenter DC_i is used to present the bandwidth of DC_i. As shown in Equation (3), StorageInfo denotes the storage space information in DC_i, and its value is calculated by adding the corresponding total storage space and the total amount of storage space of each storage node $S_i, i = 1, 2, \ldots, |\mathrm{dc}_i|$, to DC_i. FailureTime denotes the mean free error time.

dcFNS_i = (Files, FileNum, TotalAccess, TotalAnswered) shows the namespace of files or data objects, where $\mathrm{Files} = \cup_{i=1,2,\ldots,|\mathrm{FileNum}|}\{\mathrm{File}_i\}$ denote files or database objects in DC_i; FileNum shows the number of files or data objects in DC_i; TotalAccess and TotalAnswered represent the number of visits and the number of responses to requests of DC_i, respectively. Next formula shows the namespace of files or data objects:

(12) $$\mathrm{dcFNS}_i = (\mathrm{Files}, \mathrm{FileNum}, \mathrm{TotalAccess}, \mathrm{TotalAnswered}),$$

where $\mathrm{Files} = \bigcup_{i=1,2,\dots,|\mathrm{FileNum}|}\{\mathrm{File}_i\}$ denotes the files or database objects in DC_i; FileNum shows the number of files or data objects in DC_i; TotalAccess and TotalAnswered represent the number of visits and the number of responses to requests of DC_i, respectively.

(13) $$\mathrm{File}_i = (\mathrm{FileGUID}_i, \mathrm{BlockSize}, \mathrm{FileBlocks}_i, \\ \mathrm{FileReplicaNum}_i, \mathrm{FileMetaInfo}_i),$$

represents the i-th file, where $\mathrm{FileGUID}_i$ is the only GUID value in the 2CMSSS storage system; BlockSize is the block size of the file (with a default size of 64 M); FileReplicaNum indicates the number of replicas in the file's data blocks (in this 2CMSSS storage system model, the replicas are managed in the unit of files, which will be illustrated in the following section): FileMetaInfo indicates some meta-information in the file (such as file owner, user group, file size, read and write access, the number of visits, and the number of responses to requests, etc.), $\mathrm{FileBlocks}_i$ is a list of the file's data blocks, and denoted as $\mathrm{FileBlocks}_i = \bigcup_{i=1,2,\dots}\{\mathrm{block}_n\}$, and $\mathrm{SiInfo} = \bigcup_{j=1,2,\dots,|\mathrm{dc}_i|}\{s_i\}$ is the metadata information set for each storage node in DC_i.

In the 2CMSSS system model designed herein, MM and each $\mathrm{CM_i}$ constitute the master-slave relationship; each CM_i will report its storage status in the datacenter to MM by heartbeat messages and accept MM's management. They have the same relationship as the one between NameNode and DataNode in HDFS, and their interaction process is also the same. Thus, in the entire system, all storage nodes in MM and CM_i are as in CM_i and DC_i. Constitute a two-tier cascading master-slave architecture and this system is called 2CMSSS. The diagram of the system's conceptual structure is shown in Fig. 1.

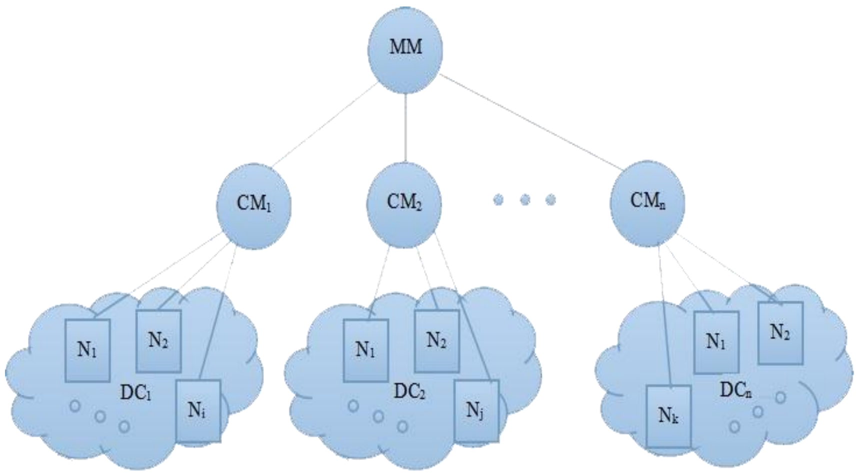

Fig. 1. The 2CMSSS concept structure diagram

2.2. Data read-write separation strategy

In a 2CMSSS storage system, when a user accesses a particular file or data object Obj, if the datacenter DC_i that receives the requests cannot find the corresponding GUID (Obj) information from the datacenter CM_i namespace, it will continue to look for it in MM namespace, and MM will return the datacenter DC_i information from the corresponding GUID (Obj) data. Then the users can access and manipulate the data Obj in accordance with the appropriate information and the corresponding 2CMSSS mechanism.

The 2CMSSS system model provides us with a specific research object for replica management strategies in cloud storage over multiple datacenters. Since the replica creation, deletion and dynamic adjustment must consider the node and datacenter of the replicas to be chosen in replica management under multi-datacenter cloud storage environments, the selection of the datacenters is the priority issue for replica management in cloud storage over multiple datacenters. Focusing on this research void, we have proposed the 2-phase locking replica consistency maintenance method for 2CMSSS model in Section 2.3, based on the corresponding replica layout mechanism.

2.3. Two-phase locking replica consistency maintenance method

With the multi-datacenter cloud storage system, the users can get closer and have faster access to the data [13]. By using the replica technology, the data and datacenters can have stronger availability, so that the system can be more fault-tolerant. However, due to the complex and erratic network environment between the datacenters, a network delay usually takes a few hundred milliseconds [14], so that the cost of storing and transmitting data replicas across multiple datacenters is high, and the consistency of synchronizing replicas across datacenters becomes quite impractical. Much work has been done in studying the data replica consistency in cloud storage over multi-datacenters, which has yielded some achievements. For example, [15] proposed the MDCC multi-datacenter consistency solution, and offered a new programming model. However, MDCC is mainly for database systems in multiple datacenters, and it is not suitable for the "Write Once Read Many" application scenarios of the 2CMSSS system.

The cloud storage system, especially with the emergence of the multi-datacenter cloud storage systems, is to deliver services to users nearby and create a better user's experience. As a consequence, in order the users to experience satisfactory access and easily utilize the read-write data, this paper principally considers the following objectives when designing the replica consistency strategy in the 2CMSSS storage system: (1) the system can provide the users with data operation services as quickly as possible, such as read or write; (2) the user access and loading from one datacenter is essentially independent of that in another, which means that the user access and loading is non-interacting as much as possible in each datacenter; and (3) data with a high level of consistency must be globally guaranteed continuance of the same high levels.

On account of the above design goals, while using the autonomy feature of each datacenter and localization of the system users in 2CMSSS, this paper designs a replica consistency strategy in the 2CMSSS storage system, following these considerations: (1) in order to be able to provide data operation services to users as fast as possible, we should take advantage of the autonomy of the various datacenters; and (2) in order to ensure a high level of data consistency, we must include a global asynchronous lock to the data.

In the 2CMSSS-based two-phase locking consistency maintenance strategy, if a user wants to update the *i*-th replica Obj_rep(*i*) of a data object Obj, it must first get through the two-locking phase.

1. Process of requesting locks (Fig. 2) is as follows:

Step 1. Prior to updating Obj_rep(*i*), the user must first obtain the user client master $i(\mathrm{CM}_i)$ lock that allows Obj_rep(*i*) operations; that is, the user should apply the lock to operate Obj_rep(*i*) from CM_i; this process is expressed as req_CMlock(Obj_rep(*i*)).

Step 2. Before the operation of locking Obj_rep(*i*), we must obtain permission from the Main Master (MM) to operate Obj_rep(*i*); that is, CM_i should apply for the lock to operate Obj_rep (*i*); this process is described as req_MMlock (Obj_rep(*i*)).

Step 3. If the main master responds to the lock requests from CM_i, MM will lock the operations to Obj in the system global namespace and return the lock to CM_i to operate Obj_rep(*i*); this process is described as MMLock (Obj_rep(*i*)).

Step 4. When CM_i receives permission from the main master to operate the lock of Obj_rep (*i*), it will immediately return to the user a confirmation message of req_CMlock (Obj_rep(*i*)); this process is described as CMLock (Obj_rep (*i*)).

Step 5. When the user completes the appropriate update operations of Obj_rep(*i*), the corresponding metadata information will be updated and Obj_rep(*i*) checksum in CM_i, and release the operation lock of Obj_rep(*i*).

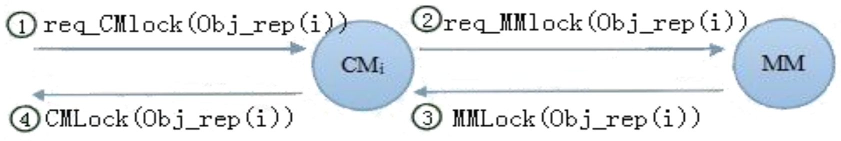

Fig. 2. Process of requesting locks

2. Process of releasing locks (Fig. 3) is as follows:

Step 1. If the user applies from CM_i to release the lock to operate Obj_rep(*i*); this process is described as CMLock (Obj_rep(*i*)).

Step 2. If CM_i responds to release CMLock (Obj_rep(*i*)), it then returns a message confirming the lock release to the user, and the datacenter where CM_i is located, will be able to see the updated Obj; this process is described as locaWriteSuccess.

Step 3. If ② happens, CM_i will apply for a lock to release Obj_rep(*i*) operation, and simultaneously send the user's updates to MM; this process is described as releaseMMLock (Obj_rep(*i*)).

Step 4. MM will synchronously update Obj_rep(*i*) in datacenters like CM_i that have Obj replica; this process is described as

Sync_write (WriteContent(Obj_rep(i))).

Step 5. After updating Obj_rep(i) in synchronous datacenters, CM_i in datacenters that have Obj replica will, in turn, return to MM the message confirming the completion of synchronous updates; this process is described as Sync_writeSuccess(WriteContent(Obj_rep(i))).

Step 6. After the process entitled

Sync_writeSuccess(WriteContent(Obj_rep(i)))

between MM and all datacenters CM_i that own Obj replica is completed, MM releases the operation lock to Obj_rep(i) in the system global namespace; this process is described as globalWriteSuccess.

Step 7. The replica consistency strategy in HDFS is adopted within the datacenter, which means the data pipeline is used. Checksum can guarantee reading consistency in the data replicas while for strongly consistent read requests of data, the decision may be handed over to the users who will then be able to select whether to compare consistency with the global data (such as a query to MM) before reading.

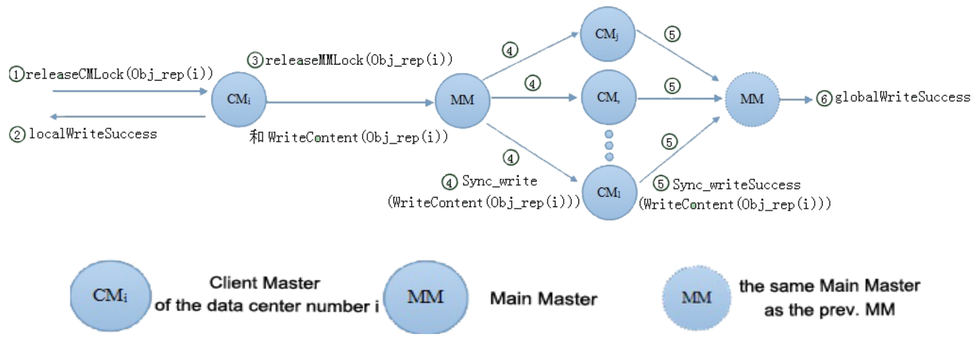
①releaseCMLock(Obj_rep(i))
②localWriteSuccess
③releaseMMLock(Obj_rep(i))
和 WriteContent(Obj_rep(i))
CMi
MM
④
④
④
④ Sync_write (WriteContent(Obj_rep(i)))
CMj
CM,
CMl
⑤
⑤
⑤
⑤ Sync_writeSuccess (WriteContent(Obj_rep(i)))
MM
⑥globalWriteSuccess
CMi
Client Master of the data center number i
MM
Main Master
MM
the same Main Master as the prev. MM

Fig. 3. Process of releasing locks

3. Experiments and analyses

To prove the efficiency of the proposed method, the cloud computing simulation environment Cloudsim is chosen [50, 51] as a simulation experiment platform, on which an expansion is made. The experiment environment simulates and randomly distributes the users' access requests. The 2CMSSS cloud storage system has 100 storage nodes, each with a storage capacity of 80 G, five data centers (assuming that each datacenter is located in a different geographic location) and 100 data source files with a size of 100 M for each file. The data redundancy is set to 10.

The experiment is compared to the replication index-based consistency maintenance algorithm [15] in terms of changes in the average update time of each replica with the number of updates. The results are shown in Fig. 4.

It can be seen from Fig. 4 that compared to the replication index-based consistency maintenance algorithm [15], the proposed replication consistency maintenance strategy based on 2CMSSS and two-phase locking has faster update speed. Because a network delay usually takes a few hundred milliseconds in the multi-datacenter environments, the replication index-based consistency maintenance algorithm needs to synchronously update all replicas, so it is more time-consuming in average. As for the 2CMSSS-based two-phase locking strategy suggested in this paper, the results can be read immediately after updating a single datacenter. Although the data updates among the datacenters are done asynchronously, in a statistical sense the average update time of the overall system is shorter. The results of the experiment verify that the replication consistency maintenance strategy based on 2CMSSS and two-phase locking has some practical value.

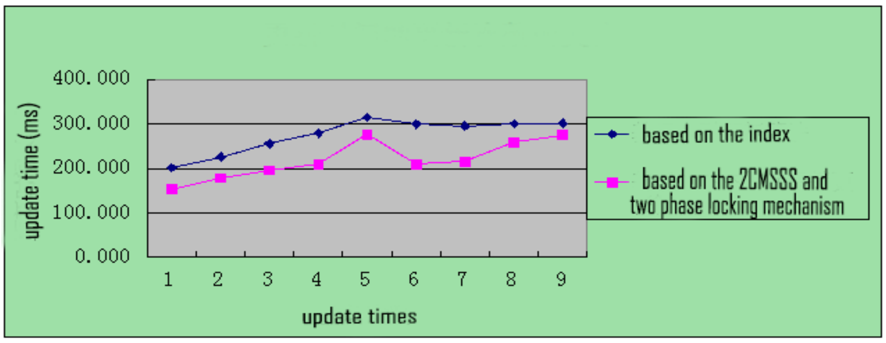

Fig. 4. Comparison diagram of the update time with the number of updates

4. Conclusions

After analyzing the cloud storage systems, multi-datacenters and traditional replication technology, this paper proposes a 2CMSSS cloud storage framework based on structural analysis of the HDFS storage system, in order to solve problems in the area, such as the architecture of the cloud storage systems over multi-datacenters, the selection strategy of a datacenter, the replica layout mechanism and data consistency maintenance strategies. This framework has simplified replica management strategies in the cloud storage over multi-datacenters and the two-phase locking data consistency maintenance method is designed based on a 2CMSSS framework. By Cloudsim simulation experiment, the efficiency of this method has been validated, and it has demonstrated some advantages over the conventional single datacenters.

References

1. Liang, X. Study of Data Replication Technology in Cloud Storage Environment. Nanjing University of Posts and Telecommunications 2013, p. 3.
2. Kraska, T., G. Pang, M. J. Franklin et al. MDCC: Multi-Datacenter Consistency. – In: Proc. of 8th ACM European Conference on Computer Systems, ACM, 2013, pp. 113-126.
3. Buyya, R., R. Ranjan, R. N. Calheiros. Modeling and Simulation of Scalable Cloud Computing Environments and the CloudSim Toolkit: Challenges and Opportunities. – In: Proc. of International Conference on High Performance Computing & Simulation, 2009, HPCS'09, IEEE, 2009, pp. 1-11.

4. L o n g, S., Y. Z h a o. A Toolkit for Modeling and Simulating Cloud Data Storage: An Extension to CloudSim. – In: Proc. of 2012 International Conference on Control Engineering and Communication Technology, IEEE Computer Society, 2012, pp. 597-600.
5. J i a n g, S., O. S o n g. Replication Index Based Consistency Maintenance Strategy in Unstructured P2P Systems. – Computer Engineering, Vol. **34**, 2008, No 19, pp. 123-126.
6. L u o, J.-Z., J.-H. J i n, A.-B. S o n g, F. D o n g. Cloud Computing: Architecture and Key Technologies. – Journal on Communications, Vol. **32**, 2011, No 7, pp. 167-173.
7. X i a o y u n, H. Researeh of Cloud Storage Service System Based on HDFS. Dalian Maritime University, 2010.
8. L i u, T., C. L i, Q. H u, G. Z h a n g. Multiple-Replicas Management in the Clound Environment. – Journal of Computer Research and Development, Vol. **48**, 2011, No 3, pp. 254-260.
9. X u, J. Research on Replication Strategy in Cloud Storag. University of Science and Technology of China, 2011.
10. D a i s u k e, T., T. S h i g e a k i, F. S a t o s h i. A Fault-Tolerant Content Addressable Network. – IEICE Transactions on Information and Systems, Vol. **89**, 2006, No 6, pp. 1923-1930.
11. B a r b e r a, P. Birds of the Same Feather Tweet Together: Bayesian Ideal Point Estimation Using Twitter Data. – Political Analysis, Vol. **23**, 2015, No 1, pp. 76-91.
12. G a o, H. A Case Study on Trade, Technology and Human Rights under the Gats. – Asian Journal of Wto & International Health Law and Policy, Vol. **6**, 2011, No 2, pp. 349-387.
13. C o r b e t t, J. C., J. D e a n, M. E p s t e i n. Spanner: Google's Globally Distributed Database. – ACM Transactions on Computer Systems, Vol. **31**, 2013, No 3, p. 8.
14. P a x t o n, W. H. A Client-Based Transaction System to Maintain Data Integrity. – In: Proc. of 7th ACM Symposium on Operating Systems Principles.
15. G u o, L. Research on Replication in Peer-to-Peer File Sharing System. University of Science and Technology of China, 2011.