ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization) Vol.2, Special Issue 1, March 2014

Proceedings of International Conference On Global Innovations In Computing Technology (ICGICT'14)

Organized by

Department of CSE, JayShriram Group of Institutions, Tirupur, Tamilnadu, India on 6th & 7th March 2014

Decentralized Approach for Balancing Load in Dynamic Cloud Environment

Karthick Smiline Britto.J[1], PrittoPaul.P[2]

ME, Department of CSE, Velammal Engineering College, Anna University, Chennai, India[1]

Asst Professor, Department of CSE, Velammal Engineering College, Anna University, Chennai, India[2]

ABSTRACT: Cloud computing is a domain which may hosts large amount of applications in its servers that are accessed by real time networks. It relies on distributed environment, but the nodes are monitored by central server which is a critical task as large no of systems needs to be monitored. Moreover in cloud, files are created dynamically which adds further load imbalance problem. Even though a load balancing algorithm was proposed to balance the load without the knowledge of the central server it did not take into its account the threshold load on nodes and the weight of the system. In terms of global knowledge is inefficient as the knowledge of the entire network is impossible to predict. As threshold is not clearly mentioned it transfers even in constant load. It also creates additional messages which add more traffic. It increases the time and movement cost. So this approach is also inefficient. Hence the objective is to propose a weight based load balancing algorithm which further rebalances the load to be uniform at each node and assigns the weight to each node considering its configuration. Thus it provides a solution to improve the performance in terms of reducing the imbalance factor and movement cost.

KEYWORDS*:* Load Rebalancing, Cloud, Distributed Hash Table, movement cost, MapReduce, Effective Weight Based Balancing

I. INTRODUCTION

Load balancing is a crucial problem that occurs in any large scale dynamic cloud computing environment which have to be resolved to a great extent .Cloud computing environment is having in it thousands of nodes in it and which may even grow up to more no in the near future. Cloud computing supports a distributed frame work environment. Operations are performed in a distributed manner. When user needs to process a fissle, the uploaded file may get processed by splitting up the file into a no of chunks and the processing is being done on each indiduval chunks. The chunks have to be uniformly allocated and then only the process on each node can be done in parallel without any imbalance problem occuring between them. The rebalancing has to be done such that no node should handle an excessive amount of the chunks in it.
Hence the network traffic would be reduced to a greater extent. Moreover, in a dynamic environment nodes may enter or leave which cause additional load on the system performance. The nodes may be added on an environment and it also leaves the environment and so that at such cases the load balancing have to be done in a different manner.

There is a central load balancer which balances the load in every system which will be difficult to perform when in the case of the dynamic environment as the large scale environment is seen is difficult to handle. Hence it proposes the global knowledge based and without knowledge based load balancing. These serve effectively in the case of the small clustered environment but are inefficient in large scale applications. The server load rebalancing technology has proven to be an greatest asset for organizations hosting widely utilized Web applications. On operating in a virtual environment load balancing execute a variety of algorithms for splitting the processing load among back-end servers. It involves periodic polling to identify the status of participating nodes can be used not only to fine tune the load distribution but also to avoid directing traffic to servers that are actually offline. Load balancers are a simple yet highly effective means for an application environment while simultaneously ensuring its availability. It exploits the capable node to improve its performance demanded. The Distributed environment the systems on the cloud can make use of the mapreduce. In such a system, the load in each node is directly proportional to the number of file chunks the node is having at a particular instant of time. Load balance among storage nodes is a critical function in clouds. Hence an load balancing considering the no of nodes its configuration and global knowledge in small scale cluster is always needed.

1. LOAD BALANCERS FOR BALANCING LOAD IN CLOUD

In Distributed services, the load balancer is usually a software program that is listening on the where external clients connect to access services. It can automatically provide the amount of capacity needed to respond to any increase or decrease of application traffic. It also prevents clients from contacting backend servers directly, which may have security benefits by hiding the structure of the internal network and preventing attacks on the kernel's network stack or unrelated services running on other ports. The load balancer also provides a mechanism such that it also performs in an efficient manner even when there is no backend server that is operating at an extent. This also involves forwarding of backup to the load balancer, and displaying any message in the case of error. It helps the team to achieve a significantly higher fault avoidance. This can defaultly calculates the rate and capacity application traffic. It also provides a computing infrastructure provides you with physical or virtual machines and other resources like virtual-machine disk image library, block and file-based storage, firewalls, load balancers, IP addresses, virtual local area networks etc. Examples: AWS Elastic Beanstalk, Heroku, Force.com, Google App Engine (ApI).

A Typical Load Balancer:

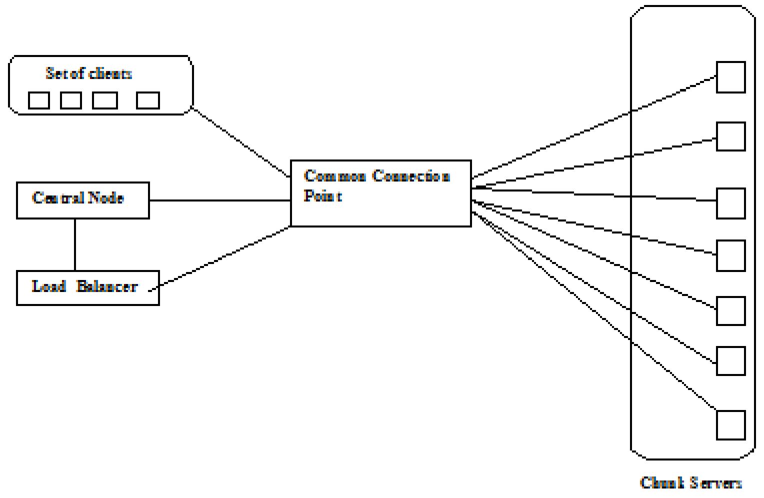

Fig No :1.1 Load Balancing

1.1 LOAD BALANCING OVERVIEW

The normal Load balancing operation is being done in typical cloud computing environment in the predicted way as follows.The Cloud actually is having in it typically a large no of servers and clients that work on the limited server space provided for them. The Server space is allocated at a particular time only for each server present in it and so that only at that time the clients can perform every client operations and can even pay for the services. This is also the same for the process of accessing the hosted applications that are present in the cloud computing scenario.
Hence dynamically a cloud can be called as a Pay as You Go model.

Moreover in the cloud every resources are also allocated dynamically and hence the client who is accessing the resource need not necessarily host the application in the system for the purpose of performing the task. Hence it assures that the distributed computing technology may plays a major role in the process of effectively utilizing the resources in a cloud computing environment. It is basically a having in it the following things.

Clients:

The clients are the user processing machines on the cloud system. The user machines may perform the application access from the cloud servers by the amount they pay for the access. The user machines may deal with performing various operations on the server systems. The general operations that are involved are the process such as storage, read and write operations e.t.c. Hence for a small process that needs to be done the client always deals with the process that has to be done in a file the client deals with the process of uploading the file to the server. The server will perform the perationby means of the data nodes and then it will return it back to the client machine. The payment is done on the basis of the time of storage and the operations done.

Central Server:

The server is the important entity in every cloud computing environment. The server performs the MapReduce task. It splits the file of a larger size into a fixed no of pieces terming each entity as a chunk. It itself also acts as an index table. It then Maps each of the entity that is splitted into a number of chunk servers that are present in its network. It selects a chunk server based on some of the following terminologies. It considers the network locality the movement cost the probability for reducing the bottle neck in the system. It performs the Mapping and Reducing task. It first splits each and every file that is obtained into a number of chunks. Then it assigns an id for every chunk that is present after assigning the id to every chunk it deals with the process of allocating them to the chunk server. After the tasks are performed it deals with the process of recollecting the chunks from each indiduval data nodes and passing it back to the client.

Data Nodes:

The Data Nodes are the processing nodes. This data nodes may deal with the process of performing the operations that are intended to be done on the file that is being uploaded by means of the client. The Data Nodes can perform the operations like reading ,writing, word count operations that can be performed. The operations are done and at the end each data node can probe the name node and after this the file is sent back to the server.

Load Balancer:

The load balancer is the important entity which deals with the process of balancing the load in the case of the balancing the load on the chunk servers when an imbalance is said to have occurred in those chunk servers that are present. The chunk servers are capable of changing the load that is occuring in the system by itself ,but the problem is that the global knowledge is difficult to predict in such systems. Moreover if the global knowledge is not available then each and every system will probe by messages the other systems to see if there is enough space that is seen for balancing hence it may result in additional traffic in such systems. Hence always a system with a balancer is always implemented and it is always necessary. On having a balancer the balancer is probed in the case of any imbalance that has occurred in the system and in when the problem is solved the status is being reported to the server.

II. LITERATURE SURVEY

Existing Load Balancing Techniques	Techniques Employed	Description
Load Rebalancing Algorithm	1)MapReduce 2)Chunk Server allocation and extraction	Mapping the files by splitting those files into small chunks and then performing a mapreduce operation .

Adaptive Load Data Migration	1)Create Data Replica 2)Migrate Data and perform Data Visit.	Setting a fixed threshold on each and every system and through that it replicates the files to maintain equal load.
Identifier Based Load Balancing	1)One to One Scheme 2)One to Many Scheme 3)Many to Many Scheme	Based on an identifier that is being assigned to each and every node the load can be transferred from the heavy loaded to light loaded
Power of two Choices Algorithm	1)Virtual Peers 2)Hash Function	Transferring the load by using a virtual server and it also involves using a hash function to select a node.

Fig 2.1 Load Balancing Techniques

The above table describes all the existing load rebalancing methods and the techniques that are involved in such methods.

2.1 MAPREDUCE UNDER LOAD BALANCING

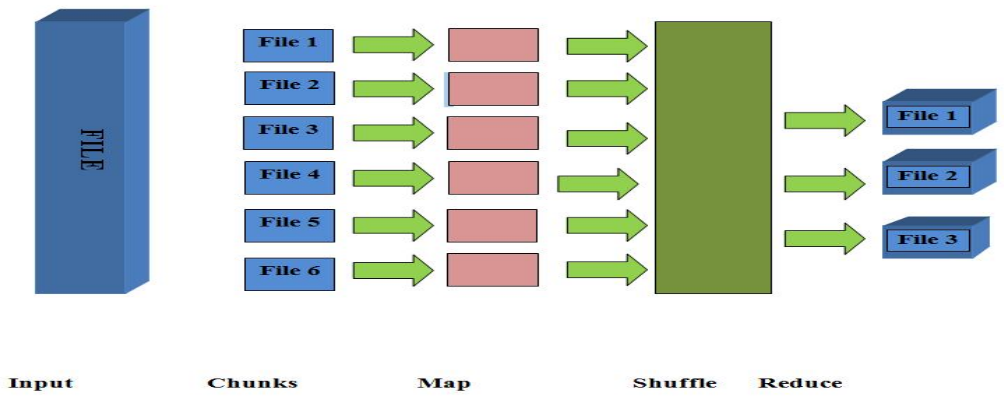

Fig 2.2 MapReduce

Map Reduce criteria a computing phenomenon that has been used in several systems. It can be used in the case of many popular implementation of systems like the Google File System and also in the Hadoop Distributed File System called as the HDFS composition system. The MapReduce phenomenon can even be done in the case of the Hardware faults also. It may basically requires only two functions in it .The functions are termed as the Map and the Reduce Functions. These functions may possess an in built library function in it that may enable to the purpose of performing the map and the reduce functions. The process by which the map and reduce tasks are done may be discussed as follows. In brief, a Map Reduce computation executes as follows:

1. MapReduce function may first deal with splitting up the input file into pieces that are of the same size. There is also a range that is specified for each file that is seen. The size of the files may range from a minimum of about 16 MegaByte and it may be up to 64 MegaByte.
2. After this a key value is assigned to each and every by the Master or the central server that is seen in the case of every distributed computing environment. The key is always assigned by means of an id, value pair. This is used in the process of reducing.

3.The Reduce tasks are done in a matter of one reduce operation at a particular time in it. It is having in it an inbuilt reduce function which may deals with the process of collocating all the pieces based on the id that has been assigned by means of the map function library.
4.The Map and the Reduce function is always done by means of the map and the reduce workers present in the system There may be M different Map tasks and also R different Reduce tasks that are available in the system. The worker who can be assigned the Map and the Reduce tasks are always selected by means of the central server that is seen.
5.Sorting is also considered as an important operation that has to be done in the case of the map and reduce operation because there are two types of sorting that has to be in the case of performing the mapreduce task

2.2 LOAD BALANCING IN STRUCTURED PEER TO PEER SYSTEMS

In the peer to peer systems any system in the network can act as a client or it can even act as a server. Hence it implements the concept of virtual server in the process of balancing the load that has occurred in it[2]. The virtual servers are some concept that involves the virtualization which can have within it some considerable amount of the load and can de.al with transferring the load from one system to another system that is present. The virtual server is employed as the distributed cloud computing environment is having within it a large amount of heterogeneous systems and they require careful transmission of the load from one system to the other through means of virtualization.
It involves the following processes and schemas in it. It deals with finding the Heavy and the Lightly loaded node at first. For this purpose it involves the usage of a threshold function that can be used for calculating the load on the servers that finds and separates the heavily and the lightly loaded nodes. Then it performs the virtual server transfer. This transfer may involve transmitting the load from a heavy node to the light node for the process of making the node heavy node light without making the light node heavy. If there is no node that can be used in transmitting the load then the virtual servers that is present can be divided into a no of smaller virtual servers and then the load can be transferred through means of the smaller virtual servers. It is having in it the one to one virtual server scheme in which the load can be transferred from one heavy node to another light node. It also has One to Many Schemes in which a hash function is involved in the process of finding the light and the heavy node.

2.3 LOAD BALANCING IN DISTRIBUTED FILE SYSTEM

The Distributed Hash Table forms a basic entity for balancing the load in any type of computing environment. Among the m = 10000 file chunks, we investigate in the experiment k =2, 4,8 replicas for each file chunk; that is, there are 5,000, 2,500 and 1,250 unique chunks in the system, respectively. The experimental results indicate that the number of nodes managing more than one redundant chunk due to our proposal is very small. Specifically, each node maintains no redundant replicas for k = 2, 4 and only 2 percent of nodes store _2 redundant replicas for k = 8.It akso involves the usage of the redirection pointer in every system such that the balancing is occurring in only the intended system and not on any other systems orthe nodes that are present in the network.

2.4 SIMULATING BALANCING

Increasing complexity of networks may result in the simulation. Simulation time often decreases the complexity in the network in terms of the time and also the memory and hardware requirement. For the simulating load balancing problem CloudSim can be used alongside having with it and the Virtual Machine components are being used and this simplifies the installment of real machines in the system for testing the balancing problem occurring in the network.. Very often performance on a distributed network is measured in a perfect environment, where the available CPUs can be used exclusively. In such a situation load balancing of parallel logic simulation can be easily achieved by an appropriate partitioning at the start of the simulation as a preprocessing phase.

III. CHORD PROTOCOL ON BALANCING

The Chord protocol supports just one operation: given a key ,it maps the key onto a node. Depending on the replication using Chord, that node might be responsible for storing a value associated with the key. Chord uses a variant of consistent hashing to assign keys to Chord nodes. Consistent hashing tends to balance load, since each node receives roughly the same number of keys, and involves relatively little movement of keys when nodes join and leave the system. Previous work on consistent hashing assumed that nodes were aware of most other nodes in the system, making it impractical to scale to large number of nodes. In contrast, each Chord node needs "routing" information about only a few other nodes. Because the routing table is distributed, a node resolves the hash function by communicating with a few other nodes. In the steady state, in an -node system, each node maintains information only about
other nodes, and resolves all lookups via messages to other nodes. Chord maintains its routing information as nodes join and leave the system; with high probability each such

3.1 SYSTEM ORGANIZATION

Load balance: Chord acts as a distributed hash function, spreading keys evenly over the nodes; this provides a degree of natural load balance.
Decentralization: Chord is fully distributed: no node is more important than any other. This improves robustness and makes Chord appropriate for loosely-organized peer-to-peer applications.
Scalability: The cost of a Chord lookup grows as the log of the number of nodes, so even very large systems are feasible. No parameter tuning is required to achieve this scaling.
Availability: Chord automatically adjusts its internal tables to reflect newly joined nodes as well as node failures, ensuring that, barring major failures in the underlying network, the node responsible for a key can always be found. This is true even if the system is in a continuous state of change. Flexible naming: Chord places no constraints on the structure of the keys it looks up: the Chord key-space is flat. This gives applications a large amount of flexibility in how they map their own names to Chord keys.

IV. LOAD REBALANCING

We first present a load-balancing algorithm, each node has global knowledge regarding the system, that leads to low movement cost and fast convergence. Yet the algorithm do not take into account the capacity of the network and the systems that a are seen in the network. Hence an algorithm that considers the capacity and the configuration of the system needs to be proposed and so that the load balancing can be done in an efficient way. Such an algorithm can be implemented by means of using virtual machines for the process of load balancing and in it the weight is also assigned for each and every machine that is existing in the system and by the weight and the network locality the load can be assigned and transferred.

4.1 EFFECTIVE WEIGHT BASED ALGORITHM

The virtualization is established as an important phenomenon in the process of balancing and simulating the load in any type of the environment. Involving virtualization technique and also considering the virtual machines in the process of simulating the load balancing can be effectively done on means of using the Effective Weight Based Algorithm.
It involves in it the set of clients used for uploading the files in the system and then it also has in it the master and the Weighted Active Balancer.

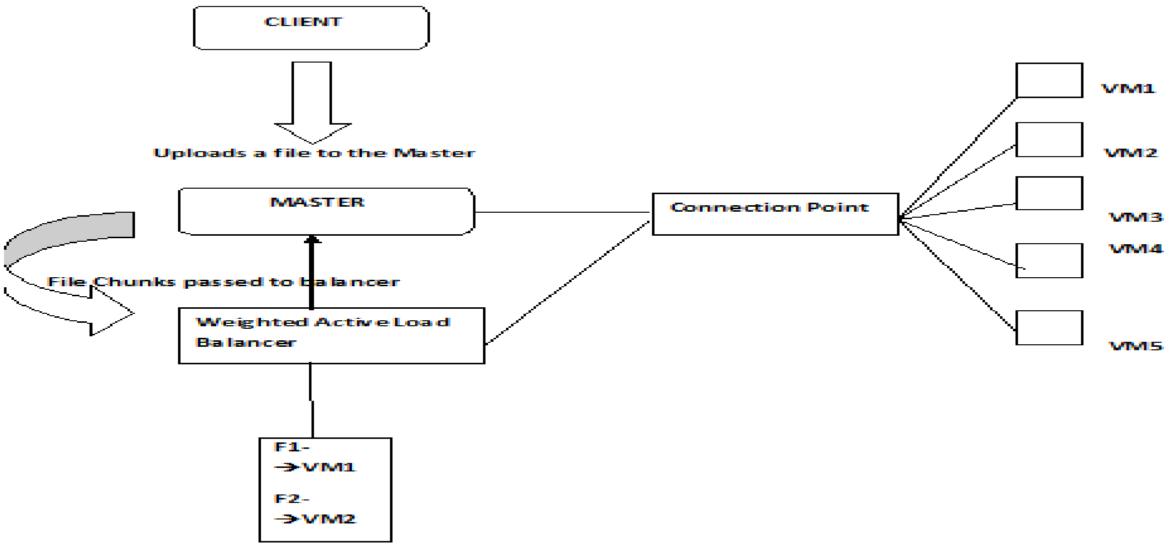
CLIENT
Uploads a file to the Master
MASTER
File Chunks passed to balancer
Weighted Active Load Balancer
Connection Point
VM1
VM2
VM3
VM4
VM5
F1-→VM1
F2-→VM2

Fig 4.1 Effective WeightBased Balancing

The Virtual Machines that are seen in the system are all allocated a certain weight in them based on the certain configuration that is present in it. It can depends upon the Configuration of the systems that are present there. A system that is having a higher capacity of Memory and the Processing speed is being assigned with it a highest weight say 10. Like wise depending on the configuration the weight may keeps on decreasing for the certain configuration of the systems that are seen . When the load is being assigned to the system the master always splits the file chunks and passes the chunks towards the balancer. Now the balancer will deal with assigning the file chunks towards various virtual machines that are seen in the network and it can be done. After assigning the chunks to the balancer the balancer will assigns them to the virtual machines an it places the ID of the file chunks in the index table that is seen. When any imbalance in the virtual machine tends to occur then at that time the virtual machine will probe the balancer only and then the balancer will find a virtual machine that in under loaded and it will assign the file to it. If two virtual machines of the same weight is found out then it assigns the load to the nearest virtual machine and the virtual machine that is free at that instant. The id will be again used by the master for the process of performing the reduce operation.

V. CONCLUSION

Here we described about a load-balancing algorithm that deals with the rebalancing in large-scale, dynamic, and distributed file systems in clouds. Our proposal strives to balance the loads of nodes and reduce the demanded movement cost as much as possible, while taking advantage of the weight of every node. It can be implemented for a small scale cloud environment by means of using the toolkit CloudSim and can be programmed on the means of using java language. It assigns the request and the files only to the virtual machine which has the largest weight in terms of power and hence due to this unwanted assignment of load to the system can be avoided to a larger extent and it may also deal with the process of minimizing the movement cost and maximize the processing speed of the distributed environment.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization) Vol.2, Special Issue 1, March 2014

Proceedings of International Conference On Global Innovations In Computing Technology (ICGICT'14)

Organized by

Department of CSE, JayShriram Group of Institutions, Tirupur, Tamilnadu, India on 6th & 7th March 2014

REFERENCES

[1] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," Proc. Sixth Symp. Operating System Design and Implementation (OSDI '04), pp. 137-150, Dec. 2004.

[2] Ananth Rao,Karthik Lakshminarayanan,Sonesh Surna,Richard Karp and Ion Stocia, "Load Balancing in Structured P2Psystems"

[3] Jasmin James et al "Efficient VM Load Balancing Algorithm For Cloud Computing Environment," International Journal On Computer Science and Engineering,vol 4,sep 2012.

[4] I. Stoica, R. Morris, D. Liben-Nowell, D.R. Karger, M.F. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications," IEEE/ACM Trans. Networking, vol. 11, no. 1, pp. 17-21, Feb. 2003.

[5] A. Rowstron and P. Druschel, "Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems," Proc. IFIP/ACM Int'l Conf. Distributed Systems Platforms Heidelberg, pp. 161-172, Nov. 2001.

[6] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels, "Dynamo: Amazon's Highly Available Key-Value Store," Proc. 21st ACM Symp. Operating Systems Principles (SOSP '07), pp. 205-220, Oct. 2007.

[7] D. Karger and M. Ruhl, "Simple Efficient Load Balancing Algorithms for Peer-to-Peer Systems," Proc. 16th ACM Symp. Parallel Algorithms and Architectures (SPAA '04), pp. 36-43, June 2004.

[8] J.W. Byers, J. Considine, and M. Mitzenmacher, "Simple Load Balancing for Distributed Hash Tables," Proc. First Int'l Workshop Peer-to-Peer Systems (IPTPS '03), pp. 80-87, Feb. 2003.

[9] G.S. Manku, "Balanced Binary Trees for ID Management and Load Balance in Distributed Hash Tables," Proc. 23rd ACM Symp. Principles Distributed Computing (PODC '04), pp. 197-205, July 2004.

[10] H. Shen and C.-Z. Xu, "Locality-Aware and Churn-Resilient Load Balancing Algorithms in Structured P2P Networks," IEEE Trans. Parallel and Distributed Systems, vol. 18, no. 6, pp. 849-862, June 2007.

[11] Q.H. Vu, B.C. Ooi, M. Rinard, and K.-L. Tan, "Histogram-Based Global Load Balancing in Structured Peer-to-Peer Systems," IEEE Trans. Knowledge Data Eng., vol. 21, no. 4, pp. 595-608, Apr. 2009.

[12] H.-C. Hsiao, H. Liao, S.-S. Chen, and K.-C. Huang, "Load Balance with Imperfect Information in Structured Peer-to-Peer Systems," IEEE Trans. Parallel Distributed Systems, vol. 22, no. 4, pp. 634-649, Apr. 2011.