2013 IEEE 11th International Conference on Dependable, Autonomic and Secure Computing

Research on Improvement of Dynamic Load Balancing in MongoDB

Xiaolin Wang, Haopeng Chen, Zhenhua Wang
School of Software
Shanghai Jiao Tong University
Shanghai, P.R.China
qiuye454@163.com,chen-hp@sjtu.edu.cn, aspiration@foxmail.com

***Abstract*— As a representative of NO-SQL database, MongoDBis widely preferred for its automatic load-balancing to some extent, which including distributing read load to secondary node to reducethe load of primary one and auto-sharding to reduce the load onspecific node through automatically split data and migrate some ofthem to other nodes. However, on one hand, this process isstorage-load –based, which can't meet the demand due to the factsthat some particular data are accessed much more frequently thanothers and the "heat" is not constant as time going on, thus the loadon a node keeps changing even if with unchanged data. On the otherhand, data migration will bring out too much cost to affectperformance of system. In this paper, we will focus on the mechanismof automatic load balancing of MongoDB and proposean heat-based dynamic load balancing mechanism with much lesscost.**

Keywords—MongoDB; heat-based; load balancing; resourcemanagement; auto-sharding; cloud computing

I. INTRODUCTION

Recent years have witnessed a big explosion of unstructuredand semi-structured data that are generated by the rapiddevelopment of internet. To store these large amount of dataefficiently, Cloud computing, as a current commercial offering,started to become apparent in late 2007 [1].Including AmazonS3[2], Google Cloud Storage[3],Microsoft Azure[4] and so on.In this context, NO-SQL storage including Google'sBigtable [5]and its open-source implementation HBase[6],Amazon'sDynamo[7],Facebook'sCassandra[8],andLin-kedIn'sVoldemort [9] are becoming the focus of attentionand large-scale distributed system is chosen to meet the need ofstorage capacity. In large-scale distributed system, fastresponse and high availability are two important performanceobjectives pursued by end users and applications. Theseperformances are largely determined by the data allocationstrategies and the load condition in the system because evenlydistributed workload can help to optimize resource utilization,maximize the throughput and eliminate the potential overload.However, on the one hand, most load balancing for NO-SQL isnot that mature or with low adaptability because of its shorttime development, on the other hand, the existing loadbalancing strategies for relational database are not fit NO-SQLdatabase, thus we should develop new load balancing strategywhich is more efficient and with higher applicability.

To design a proper load balancing strategy, three pointsshould be taken into consideration. Firstly, due to the facts thatsome particular data are accessed much more frequently thanothers and the "heat-diffusion" is not constant as time going on, thus theload on a node keeps changing even if with unchanged data. Sothe load balancing strategy should be heat-based, the heat in load balancing is the frequency of processing of the dataSecondly,load balancing should be a life long journey rather than justallocating the data when adding data to database, in other words,the location of data should not be immutable. Thirdly, whencome to data migration, the amount of data together withmigrate distance and other factors should be taken into accountto decide to cost of migration. Besides, replica set should beused to share load except to ensure automated failover.

Automatic load balancing is an ideal target of databasedesign, many storage products try to obtain it. However, theyare mostly storage-based. For example, Dynamo achievesautomatic load balancing mainly through "uniform hashing".The solution of load balancing in Bigtable is based ontraditional server-farm. That is a master server monitors theload conditions of TServers continuously, and then migratesdata from the overload server to an underloaded one.MongoDB [10] is precisely widely favored for the reason that itcan achieve read and write extension automatically. Readextension is achieved mainly by replica set. Write extension isachieved mainly by auto-sharding. Sharding refers to theprocess of splitting data up and storing different portions of thedata on different machines. It migrates chunks among differentshards; each represents a smaller range of values within theshard's range. However, in our opinion, the load balancingstrategy in Mongo DB has at least three aspects that are notperfect. Firstly, Mongo DB determines imbalance based solelyon the difference of chunk number of each shard, in other words,it identifies imbalance only based on the amount of data. Thus itdoes not meet the common scene of the uneven distribution of hot spots. Secondly, Mongo DB conducts chunk migration onlytaking the amount of data into consideration, withoutconsidering the physical distance and other specific shard'sload information. Thirdly, Mongo DB only has the scale upmechanism while scale down has not been involved in, so itneeds to be optimized.

In this paper, we will take MongoDB for example to illustrate a new algorithm aiming to achieve automatic shardingbased on heat diffusion and make full use of replica set to share load and ensure high availability at the same time. Section 2 reviews some related work and discusses the motivation of thiswork. Section 3 addresses the overview of the framework of ourload balancing strategy. Then in Section 4, we propose the loadbalance strategy with detailed description of the algorithm.Experimental evaluation is presented in Section 5 with somesimulation results. Finally Section6 concludes the paper.Section 7 will give an acknowledgement to illustrate some additional information.

978-1-4799-3381-5/13 $31.00 © 2013 IEEE
DOI 10.1109/DASC.2013.49

II. RELATED WORK AND MOTIVATION

Some of previous works have focused on load measurementand finding sources of imbalance. Efficient, scalablemeasurement of load [11] identifies whether load imbalance isa problem for a particular application. If the imbalance is aproblem, then try to fix it, otherwise, ignore it. In [12], they tryto find the cause of imbalance. Imbalance attribution [13]provides insight into the source code locations that cause ofimbalance. However, these two works need a goodunderstanding of application elements, which is too hard toachieve in most scenario. In [14], they introduce a dynamic andintegrated resource scheduling algorithm (DAIRS), whichtreats CPU, memory and network bandwidth integrated for bothphysical machines and virtual machines and develop integratedmeasurement of the total imbalance level of a Cloud datacenteras well as the average imbalance level of each server forperformance evaluation. In [15], they model the VE by dividingit regularly into a large number of square cells, each cellcontains some objects, and its load is determined by the numberof objects inside it, the author in[16]use virtual IDs to refer to theperformance and capacity of nodes. Different IDs can handledifferent load levels and different processing time helps to findthe idle servers, but these are storage-based measurement,which is not appropriate unless with even access.

Other researchers have focused on the data migration problemin a load rebalancing scenario .they try to identify a bettermethod according to the cost or the speed of data migration. Forexample, in [17], two interference-aware prediction models arebuilt to predict the migration time and performance impact foreach action using statistical machine learning and then create acost model to strike a right balance between the two ingredientsof cost; H. C. Lim et al. [18] try to address the problem ofautomatic control for elastic storage. Two controllers areresponsible for adding (or removing) a storage node andcontrolling the data migration respectively. However, these twoworks only pay too much attention on network resources,without considering which part of data to migrate and its impacton the operation on the database. In [19], they propose a hybridcontrol strategy for load balancing. This strategy would use thecentralized load balancing strategy to redistribute the load ofthe node. From the global point of view, a controllable nodecontrols two storage node clusters, and each storage nodecluster belongs to two controllable nodes. Load balancingbetween the overlapping nodes continues timely operating theiterative batch load method to distribute the load to the global.But it is not a real-time method.

Actually, there are other researches on the improvement ofrebalancing on MongoDB. For example, in [20], for each chunkin MongoDB, its load determined by the number operationsincluding adding, deleting and querying on it. However, theweights of different operations are hard to assign only byexperience.

In conclusion, heat-based load balancing of NO-SQLdatabase is necessary for large-scale distributed system.However, there is not a comprehensive solution has beenproposed, especially a method combined with architecture ofdatabase, like using replica set to reduce the rebalancing cost.Thus a better method should be proposed.

III. FRAMEWORK

We reformed the framework of MongoDB, especially thedecision making group as follow, though it is based on theframe work of MongoDB, it is widely applicable.

Our load balancing strategy can balance the workload amongdifferent physical servers and virtual servers based on heatdiffusion in a dynamic way. Fig. 1 features the corecomponents in our load balancing strategy as follows and wemainly reformed decision making group and added load agentto replica management. Thus we will describe the first two partsbriefly and the third part in detail.

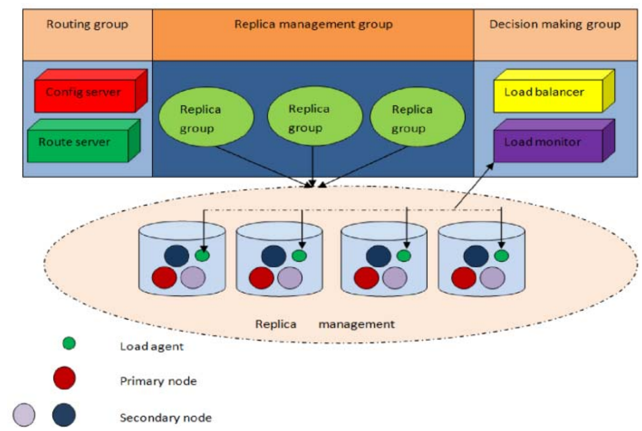

Fig. 1.Framework

A. Description ofRouting Group

As we all know, in most NO-SQL database includingMongoDB, different parts of data are partitioned to differentnodes, thus, each time when there is a client request, it is routedto routing group to decide which node to get the specific data.That is, routing group act as a proxy between client and storagesystem. Besides, in MongoDB, if it is only a read request,routing group will map it to a secondary node to share the loadof primary one.

Config server: It does not only keep the map between keyrange and VM server, but also keep a map between VM serverand physical server.

Route server: For updating or inserting, rout server will routrequest based on key and the information of config server, forquerying, it will route request to a secondary node if the criteriacontains specific key. Otherwise, it does scatter and gather onsecondary node.

B. Description of Replica Management Group

Data Nodes: As described in figure1, on one hand, the dataare distributed on multiple shards; each shard acts as a replicagroup, including one primary node and one or several secondnodes. Primary node is in charge of writing request, updatingrequest, and read request, but only read request can act onsecondary node. On the other hand, chunks are hosted in shards,each chunk specifies a key range, and chunks together span thewhole key space.

Load Agent: The purpose of load agent is to provide loadinformation of the server, physical or virtual; it providesinformation to the load monitor. Because different replicagroup might have replicas in the same physical server, it is hardto presume the percentage of resource is being used by whichpart at particular moment. Therefore, we think it is betterallocate one replica on each VM. In addition, it is easy tooperate and the cost is reduced when come to data migration.

C. Description of Decision Making Group

The Decision making group acts as an important part in loadbalancing system and keeps load evenly across the cluster. It isin charge of making decision whether to take load balancingsteps according to our load balancing policy. In MongoDB, loadbalancing strategy is to transfer data in chunk unit across shardsaccording to the number of chunks, but in our reformedframework, we try to avoid data migration which brings lots ofcost and with the help of load monitor, rebalancing decision isbased on the load on each node.

Load Monitor: Load monitor is a new part in the framework;it collects load information from every load agent within certaintime interval. The load information should be refreshed at asuitable interval so that the information provided is not expired.In addition, it will compare the load with the normal load range,if the load is found to be abnormal, and then it will pass a sign toload balancer.

Load balancer: load balancer is a reconstructed part by us; itis in charge of taking rebalancing steps according rebalancingworkflow and algorithms. Whenever load balancer is arousedby Load Monitor, firstly it will figure out whether the node is aphysical node and try to rebalance the cluster according tospecific workflow. In MongoDB, the migration chunk is justthe top chunk of each shard, ignoring the heat on it. But in ourframework, the chunk with largest heat will be split to avoidfailing to distribute the access load. Thus; it works as activist ina core part.

IV. HEAT DIFFUSION BASED LOAD BALANCING WORK FLOWAND ALGORITHMS

Routing group in MongoDB could be viewed as a normal shard, because it is also constituted with a replica set,and the load on it will be desperate to one of idle node or other routing node will less load.Thus, we will describe how to define overloaded and underloaded node with specific algorithm at first, then we will describe the improved automatic shardingfrom two parts, VM overloaded balancing and physical overloaded balancing and illustrate the procedure with flow diagrams, and then underloaded process will be described finally.

A. Exception Detection Algorithm

The overload criterion is so important that we take many experiment to define it.We used the utilization of each node to evaluate the load on it. Actually, there are many types of resources on either physical node or virtual node, such as CPU, memory,bandwidth, disk and so on. However, we choose the first three resources to define the load on each node for simplification. Todescribe the algorithms, we will define all parameters asfollows.

TABLE I
PARAMETERS OF LOADBALANCING

Parameters	Description
U_{cmi}	The monitored utilization of CPU of node i
U_{mmi}	The monitored utilization of memory of node i
U_{bmi}	The monitored utilization of bandwidth of node i
U_{cu}	The upper bound of utilization of CPU
U_{mu}	The upper bound of utilization of memory
U_{bu}	The upper bound of utilization of bandwidth
U_{cl}	The lower bound of utilization of CPU
U_{ml}	The lower bound of utilization of memory
U_{bl}	The lower bound of utilization of bandwidth
W_c	The weight of utilization of CPU
W_m	The weight of utilization of memory
W_b	The weight of utilization of band width
U_{ii}	The integrated utilization of node i
U_l	The lower bound of utilization of node i

Firstly, we defined the upper bound of indexs, Ucu,Umu,Ubu,and lower bound of threeresources of each node. Then we define overloaded andunderloaded node by comparing the monitored utilization withits upper bound and lower bound.

Because when the utilization of any resources on the node isover its upper bound, the availability on it will be seriouslyaffected, but to identify whether a node is underloaded, weshould take monitored utilization of each resource intoconsideration to get an integrated measurement.

$$U_{ii} = W_c * U_{cmi} + W_m * U_{mmi} + W_b * U_{bmi} \tag{1}$$

$$U_1 = W_c * U_{c1} + W_m * U_{m1} + W_b * U_{b1} \tag{2}$$

B. VM Overloaded Balancing

As described in (a), when specific VM is detectedoverloaded, it means that there is at least one hot point on thenode. Thus firstly we locate specific hot chunk according to thenumber of request on it, then split the chunk at the middle of thekey range of the chunk, this step will avoid failing to distributethe access load and initiate the auto-sharding of MongoDB, asdiscussed in [10], whenever the difference of chunk number isover 2, which could be set, the auto-sharding will be started. Ifthe condition is not resolved yet, the process will be done onemore time until the load on all VM nodes turned to be normal.

C. VM underloaded Balancing

For underloaded VM node, we wouldn't process it at once because it is possible that an overload one will find this node to merge after a while. So we will start a timer and wait for a specific interval. If until time up, there is no request, we could judge there is no proper overload node in the cluster. Then we will try to find a paired underloaded node. If succeed, we will migrate the chunks on the shard will less data to the paired underloaded VM, and After the migration operation, all of the indexes of both the underloaded VM node are not over the up bound.

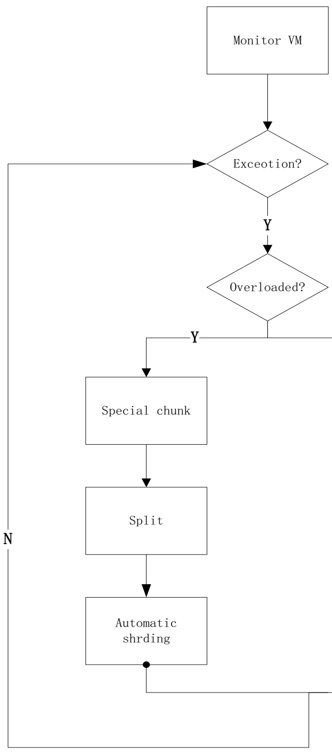

Fig. 2.VM load balancing work flow

D. Physical Overloaded Balancing

As described in (b), when specific physical node overloaded,we will not migrate data immediately, instead, we will try tochange primary node to secondary one on the overloadedphysical node which will reduce the cost of rebalancingsubstantially, because the write requests could be only executedon primary node, it is possible that the switch action couldreduce the load on the physical load and lead it to a normal one.

TABLE 2
Parameters of Algorithm

Parameters	Description
L_{pi}	The load on primary node of replica set i
L_{ii}	The load on secondary node i of replica set i
R_i	Replica set of i
V_i	Virtual node i
P_i	Physical node i
L_i	Load on Physical node p

TABLE 3
Algorithm

Algorithm: Algorithm
Inputs : L_{pi} on V_p of P_i , L_{ii}on V_i ,Ri of P_j Output : result:enum<1, 0>
For V_i in R_i(i=1,2,3,…) if Li is lowest in R_i a=L_i - L_{pi} + L_{ii} b=Lj+L_{pi} - L_{ii} a is in normal interval and b is in normal interval then result = 1 else result = 0 return result

Firstly, we will predict whether the switch action couldresolve the problem according to predict algorithm above, if itcan achieve the target, then we switch the primary node, that is,he primary node is switched to second node of the replica set,while the secondary node is switched to primary one. If it isn’tsucceed, we should try to search a paired physical node with more spare resource and migrate a specific VM to it, the data onthe VM must be as small as possible. After the migrationoperation, either the overloaded physical node or the paired oneis normal. However, if there is no physical node could be aired,we should add a new physical node to the cluster, and specificVM will be migrated to it by paired operation. Actually, in mostcases, we could just modify the map between physical node andVM node without data migration.

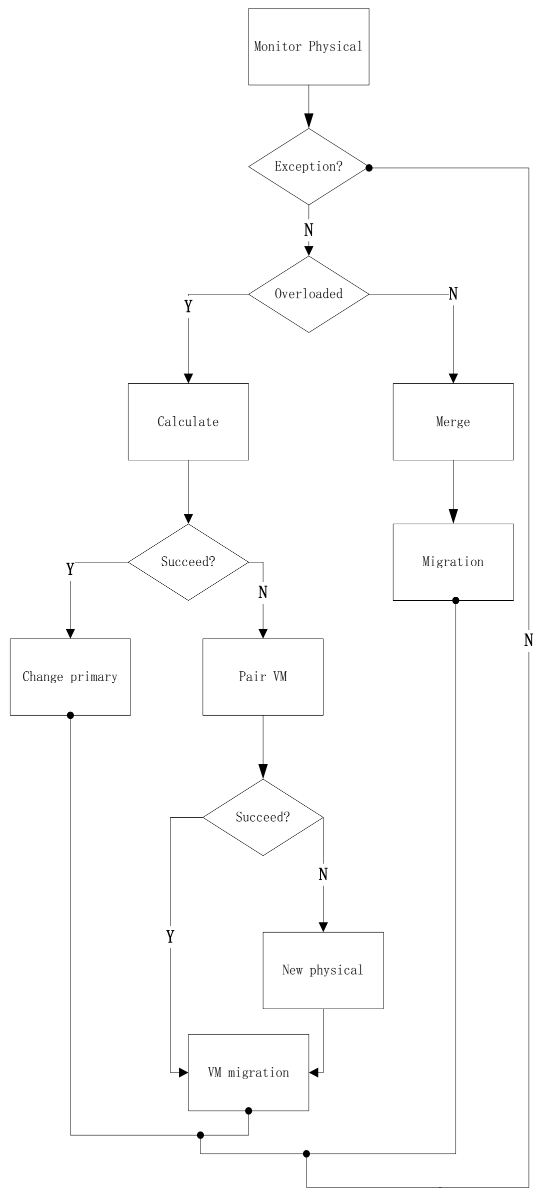

Fig. 3.Physical load balancing work flow

E. *Physical Underloaded Balancing*

For underloadedphysical node, we wouldn't process it at once because it is possible that an overload one will find this node to merge after a while. So we will start a timer and wait for a specific interval. If until time up, there is no request, we could judge there is no proper overload node in the cluster.

For physical node, we will search for other underloaded node to pair, if succeed to find one, and then migrate the node with less data to another one to merge them. Thus, one of them will become idle. Otherwise, restart the timer and wait for another pair request.

V. EXPERIMENTS

A. *Experiment Configurations*

To demonstrate the effectiveness of the propose framework, we build an experiment environment with 8 physical nodes and 25 virtual nodes.

In the physical nodes, XenServer is set up as the virtualization server. And in each virtual node, our load balancing algorithm is deployed and interacts with the API of XenServer.

In the virtual nodes, Ubuntu 12.04 Server is installed in each node. And shards as MongoDB storage nodes are deployed in these virtual nodes.

The details of the physical machines used to set up the experiment environment are as follows.

Table 5. The detail of physical machine

Category	Name	CPU	Memory (GB)	Disk (GB)
Virtualization Server	XenServer1	Intel i5 3.30G Hz	4	500
	XenServer2		4	500
	XenServer3		4	500
	XenServer4		4	500
Proxy Node	Proxy1	Intel i3 3.30G Hz	4	500
	Proxy2		4	500
Client	Client1		4	500
	Client2		4	500

The details of the values of the parameters mentioned above for work-load analysis are as follows.

Table 4. Configuration of parameters for work-load analysis

Parameters	Values for physical nodes	Values for virtual nodes
opt of CPU	0.6	0.7
opt of Memory	0.95	0.95
opt of NetworkIO	0.95	-
weight of CPU	0.1	0.2
weight of Memory	0.2	0.8
weight of NetworkIO	0.2	-
threshold of underloaded	0.3	0.3

The choice of the parameters should weight the costs and performance. Presently, we choose the parameters according to the performance priority principle to guarantee the storage system performance. And the method for determining the values of parameters will be presented by the later work.

Physical machine Clinet1 and Client2 are used to simulate clients for data access. 200 users are simulated by them respectively. At the beginning of the experiment, there is no user access to the storage system. It lasts for 1000s and the work-load of the entire system is quite light in the first 1000s. Then, the simulation of user access starts. The number of concurrent simulated users increases evenly in the first 1000s. Then it reaches steady state. The simulated users send their data requests continuously and the interval between two requests is 100ms. The state lasts for 5000s then the simulation stops. we use ycsb as test tools,its an effective NoSQL benchmark tool.

B. *Resource Management Impact Evaluations*

Ideally, when the system is busy, more physical machines should be in service to guarantee the basic performance of the entire system. However, when the system is free, some free physical machines should be shut down or go to sleep for energy-saving. As the follow figure 4 shows, at the beginning of the experiment, we deliberately set the number of physical nodes in service to be 4. But the work-load of the entire system is extremely low since there is no simulated user access to the system at the beginning. The resource management algorithms work and the number of physical machines in service is reduced from 4 to 3 for energy-saving. Then the user access simulation starts. Thus the work-load of the system gets heavier and heavier. At this time, the resource management algorithms work and during the heavy work-load period, the number of physical nodes in service is increased from 3 to 4 to guarantee the basic performance of the system. Since the work-load caused by simulation user access is not steady, the number of physical nodes in service changes.

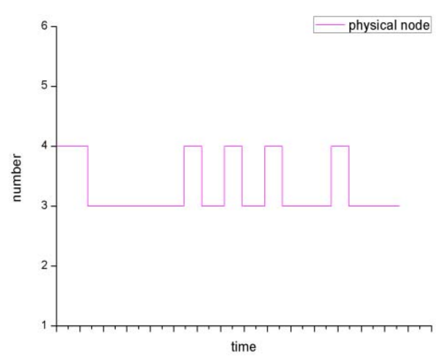

Fig. 4.Physical node in the cluster

C. *Dynamic Load Balancing Impact Evaluations*

When the work-load exception is detected, the dynamic load balancing algorithm begins to work. As figure shown, this physical node is extremely heavy during the work-load heavy period. Thus, its resource utilization exceeds the defined threshold. Fortunately, with the impacts of the dynamic load balancing algorithms, its extremely heavy work-load is reduced as shown in the figure.

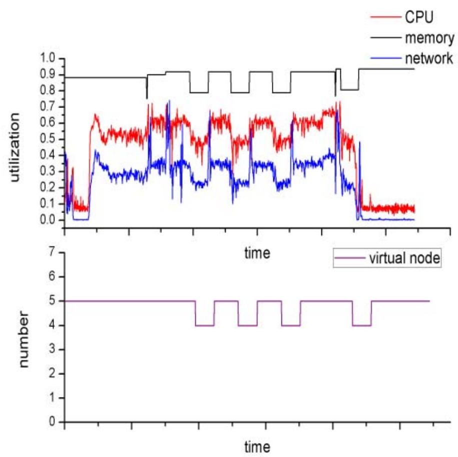

Fig. 5.Physical node load balancing

In addition, the work-load is transferred to another physical node. As shown in the follow figure6, at the beginning, this physical node is determined to be free and with the impact of the resource management algorithm, all of its virtual nodes have been migrated to other physical nodes. But because of the heavy work-load caused by the simulation user, some virtual nodes are moved to it to guarantee its basic performance. Thus its work-load is increased during the heavy work-load period.

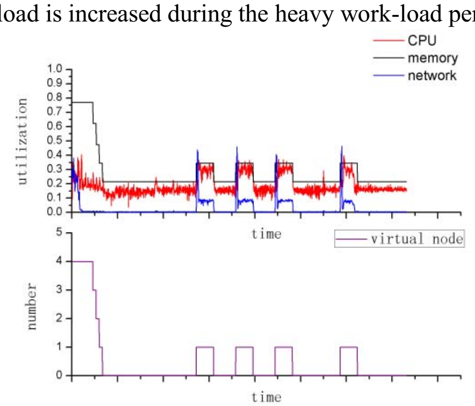

Fig. 6. xen-p2 resource utilization and the number of virtual nodes over time

And as the follow figures show, all the computation resource utilization is controlled under its defined threshold. It further demonstrates the effective of the dynamic load balancing algorithm. The benefit is that it can avoid the system bottlenecks caused by the excessive utilization of computation resource.

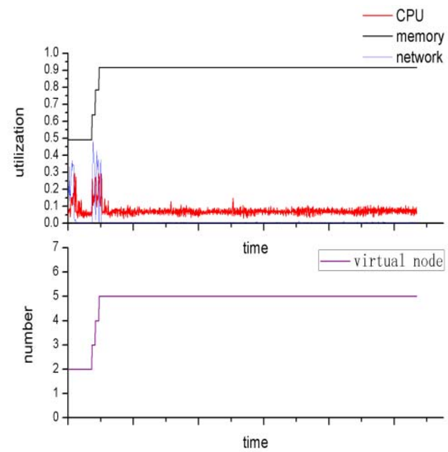

Fig. 7. xen-p3 resource utilization and the number of virtual nodes over time

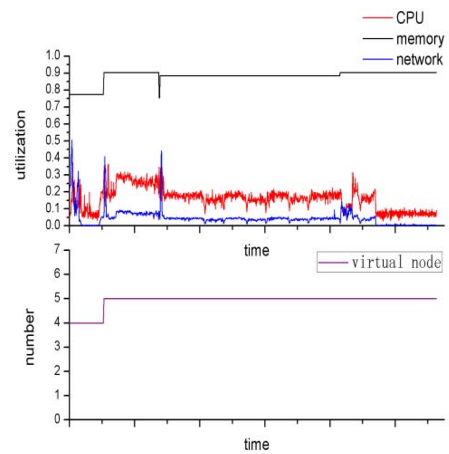

Fig. 8. xen-p4 resource utilization and the number of virtual nodes over time

VI. CONCLUSION

As discussed above, dynamic heat diffusion-based load balancing is important to storage application with hotspot data.Traditional data amount-based load balancing cannot meet the need in real scenario, including the load balancing mechanism of MongoDB. In addition, replica set is more than just a backup; it also can help to balance the load in the system.

At presented before, our work is effective to some extent.It effectively rebalanced the load in the system based on heat-diffuse, and made full use of replica set to reduce the migrate cost.

In addition, our mechanism is not only fit to MongoDB, it could be used into other database to optimize the performance of load balancing.

VII. ACKNOWLEDGEMENT

This paper is supported by Shanghai Municipal Science and Technology Commission under Grant No.11dz1502500.

REFERENCES

[1] IBM. IBM Introduces Ready to Use Cloud Computing.IBM PressRelease, 15th November 2007.

[2] Amazon, Amazon S3 [OL], available at: http://aws.amazon.com/s3.

[3] Google,GoogleCloudStorage[OL],availableat:http://www.google.com/enterprise/cloud.

[4] Microsoft, Microsoft Azure [OL], available at:http://www.windowsazure.com/en-us/

[5] Amazon SimpleDB[OL]. available at:http://aws.amazon.com/simpledb/.

[6] HBaseHomepage[OL]. availableat:http://hbase.apache.org.

[7] G. DeCandia et al. Dynamo: Amazon's Highly Available Key-value Store.In SOSP, page 220, 2007.

[8] A. Lakshman and P. Malik. Cassandra: A Decentralized Structured Storage System.SIGOPSOper. Syst. Rev.,44(2):35{40, 2010}.

[9] Project Voldemort[OL]. available at : http://project-voldemort.com.

[10] MongoDB.features. [OL]. available at : http://www.mongodb.org/

[11] K. A. Huck and J. Labarta. Detailed load balance analysis of large scale parallel applications. In Par.Processing, 2010.pp.102-110

[12] N. R. Tallent, J. M. Mellor-Crummey, M. Franco, R. Landrum, and L. Adhianto. Scalable fine-grained call path tracing. In Intl. Conf. on Supercomputing, 2011..pp. 123-125.

[13] N. R. Tallent, L. Adhianto, and J. M. Mellor-Crummey.Scalable identification of load imbalance in parallel executions using call path profiles. In Supercomputing, 2010. pp. 103-112.

[14] WenhongTian,YongZhao,University of Electronic Science and Technology of China,Chengdu,China A DYNAMIC AND INTEGRATED LOADBALANCING SCHEDULING ALGORITHM FOR CLOUD DATACENTERS Proceedings of IEEE CCIS2011. pp. 213-220.

[15] Deng, Yunhua and Lau, Rynson W. H. (2010): Heat diffusion based dyn amic load balancing for distributed virtual environments. In: Proceeding s of the 2010 ACM Symposium on Virtual Reality Software and Techno logy 2010. pp. 203-210.

[16] Lin Xia, Han-Cong Duan, Xu Zhou, Zhifeng Zhao, Xiao-Wen Nie,“Heterogeneity and Load Balance in Structured P2P Syste” 2010. pp.303-310.

[17] Yonghui Zhang, Chunhong Zhang, Yang Ji, Wei Mi, “A Novel Load Balancing Scheme for DHT-BASED Server Farm”,2010 3rd IEEE International Conference on Broadband Network and Multimedia Technology (IC-BNMT), 2010, pp. 980-984.

[18] Xiulei Qin, Wenbo Zhang, Wei Wang, Jun Wei, Xin Zhao, Tao Huang: Towards a Cost-Aware Data Migration Approach for Key-Value Stores. CLUSTER 2012: 551-556

[19] H. C. Lim, S. Babu, J. S. Chase. Automated control for elastic storage. In: Proc. of the 7th international conference on Autonomic computing (ICAC '10). 2010. pp. 1-10.

[20] YimengLiu ; Sch. of Comput. & Inf. Technol., Beijing Jiaotong Univ., Beijing, China ; Research on The Improvement of MongoDB Auto-Sharding in Cloud Environment.Computer Science & Education (ICCSE), 2012 7th International Conference on Date of Conference: 14-17 July 2012 . pp.851 - 854