Water-Filling: A Novel Approach of Load Rebalancing for File Systems in Cloud

Divya Diwakar
Department of Information Technology
SATI College,
Vidisha (M.P.), India

Sushil Chaturvedi
Department of Information Technology
SATI College,
Vidisha (M.P.), India

S.K. Shrivastava, PhD
Department of Information Technology
SATI College,
Vidisha (M.P.), India

ABSTRACT
File systems serves as the backend for cloud computing and load balancing is the relevant issue in context of resource utilization for distributed file systems in cloud. Prior to this, it is fruitful to identify the load on the storage servers (nodes) which is equivalent to number of file chunks it stored. Here is an extension of load balancing i.e. water-filling load rebalancing operated on distributed approach based on water-filling methodology, contrasting all the earlier algorithms that were grounded on centralized and distributed approaches, is used for balancing the load on servers by distributing file chunks making it more amplified to perform map reducing tasks. Water-filling approach enhances the scope of algorithm by calculating the total load exchange cost and rejoining cost in terms of file chunks migrated. Besides this, distributed approach, which employs self reliant load balancing on each node, is preferred due to its effortlessness. In distributed approach the node having highest and the lowest load is preferred to exchange chunks but often not on least possible load exchange cost. In this paper, an improved load distribution task based on physical network locality significance is calculated by water-filling algorithm, is used as metric for minimizing the load exchange cost to improve the load balancing for overcoming the shortcomings of centralized and distributed approach. Experimental results reports that water-filling load balancing algorithm is 81% better in load distribution than distributed load rebalancing, coagulates less load movement cost and even predicting reduced rejoining cost for migration of chunks in the panoptic environment of cloud.

General Terms
Parallel and distributed systems, file systems, Algorithms

Keywords
Cloud computing, distributed file systems, load balance, water-filling.

1. INTRODUCTION
Cloud computing is the enthralling technology in the field of computer science. Cloud is the provider of dynamic services and is a collection of computing and communication resources located over distributed datacenters that is shared among different users. Reliability and scalability are the key features of cloud and is achieved by using the compelling technologies of cloud such as Map Reducing programming [1], distributed file systems [2], virtualization [3], etc. Distributed file systems for cloud applications provide the nodes for the storage of files and computation over them. A file in distributed file system is divided into number of chunks allocated to specific node in order to perform map reduce task parallel over the nodes. Load balancing is the major issue in distributed file system. Load of a node refers to the number of chunks stored on the node. Since nodes in cloud can be dynamically upgraded, deleted or added in the file system and files are also created, deleted and appended so load on the nodes can vary, resulting in the non uniform distribution of file chunks over the nodes which leads to load imbalance state. In order to utilize the resources well and maximize the performance of map reduce task, load among the nodes has to be balanced. Hadoop distributed file system [4] (HDFS) and Google file system [5] (GFS) are used to overcome the issue of load balancing but they rely on single name node or master node to manage metadata of the file systems and to balance the load. This centralized approach is though easy to implement but inefficient if number of files and its access increase resulting in extra workload over the single central node and thus, central node will become performance bottleneck. Multiple name nodes architecture in HDFS has also been designed but failed, as workload among the name nodes can also change over times which lead to load imbalance state among the name node.

A distributive approach to solve this load balancing problem in structured peer to per systems[6],[7],[8],[9] has also been proposed to allocate the file chunks as uniformly as possible without relying on central node for load balancing task. The basic idea of distributives approach was to offload the load balancing task to the storage node so that an extemporaneous balancing can be done by nodes thus eliminating the dependency on central nodes. This approach also aimed to minimize the movement cost caused by rebalancing the loads on nodes to maximize the network bandwidth. The storage nodes in this approach were structured using distributed hash tables (DHT) [10],[11],[12] network in order to discover file chunks on nodes by providing unique identifier to each chunk. Rebalancing among the nodes is performed without having global knowledge about the load of other nodes. The previously used method for load balancing does not considered the movement cost and node heterogeneity which incur network traffic.

Distributive approach looked for the reduction of movement cost and algorithm overhead of load distribution introduced in distributed DHT network. In contrast to HDFS and GFS, distributive approach of load rebalancing for distributed file systems perform remarkably well in terms of load imbalance factor, movement cost, and algorithm overhead. Therefore, we propose a novel approach of load rebalancing using water filling technique [13] for uniform load distribution same as water distributes itself uniformly at uneven surface. Iterative water filling algorithm is used in communication theory for maximizing the power capacity on the sub channels of various channel units in the network.

Water-filling algorithm is analogous to load balancing and gives the idea of load distribution over light nodes to bring their load at threshold by shedding load from heavy nodes making even distribution of load on each node.

Contrasting all the other algorithms, this algorithm balances the load of the nodes in distributed file systems with lesser movement cost and less algorithm overhead. The idea followed is that this algorithm introduced a method to search a light node physically closet and having fewer loads to depart thus reduces the iterations for load balancing in dynamic environment. Lastly, the effectiveness of the proposed algorithm is validated by various experiments and concludes that the proposed algorithm can ultimately enhance the load balancing ability of the distributive approach, decrease the movement cost, algorithm overhead and finally reduce the response time of the whole system.

2. RELATED WORKS

In context of load rebalancing for distributed file systems in cloud several papers have been studied and few of them are found relevant with our works which are summarized as follows:

A. Rao et al. [14], worked for addressing the problem of load balancing in heterogeneous p2p systems that provides DHT abstraction distribute data among different peer nodes by randomly selecting the nodes resulting in $(O\log N)$ imbalance. There work designs a load balancing algorithms and presented three techniques of load balancing and their optimality. First technique is implemented in the dynamic systems involving continuous insertion and deletion of items. Second technique is developing a theoretical underpinning of proposed techniques and third technique is to build a prototype of load balancing on top of Chord lookup system.

H. Shen et al. [15] presented the locality aware randomized load balancing algorithm which took into account both proximity [16] and dynamic features of DHTs [17]. For dynamic feature randomized matching between heavy and light nodes can be done .But they do not consider physical proximity of node. There are locality-aware methods in load balancing to deal with this problem but they are costly in terms of network construction and maintenance. Their algorithms distribute application load among the nodes by "moving items" according to node capacities, as well as node proximity information in topology-aware DHTs. They provided a method. The proposed algorithm lifts up the performance of key value caching system. They presented a new scheme of load balancing for key value cache system in cloud environment in consideration with the effect of load balancing and the scope of invalid cache. Cache-invalidation-scope model is established to improve the effect of load balancing.randomized factor in the searching process to deal with proximity. Also they improved the efficiency of load balancing by d-way probing.

Hung-Chang Hsiao et al. [18], proposed the load rebalancing algorithm for distributed file systems in cloud to cope up with the problem of centralized approach of load balancing where in the dynamic environment, nodes simultaneously serve storage and computing, files are dynamically added, removed and updated in the system ,load balancing done by central load balancer by dividing the file into chunks and reallocating on different nodes, is put under considerable workload, becomes performance bottleneck and lead to single point failure. They proposed the distributed load rebalancing algorithm which is compared against a centralized approach in terms of load imbalance factor, movement cost and algorithmic overheads. Their simulation worked to search for light and heavy weighted nodes where weight resembles the number of file chunk on each node depending on threshold and their algorithm outperforms to balance the load among the nodes by chunks transfer among nodes without any central load balancer.

Revathy R et al. [19] gave new idea of efficient load rebalancing for distributed system in cloud. Their other objective was to reduce the network inconsistencies and network traffic responsible to load imbalance factor among hundreds of nodes. The reduction of network inconsistency can result in maximization of network information measure in order that large applications will run in it. Because of property of quantifying they are able to add, delete, and update new nodes in order that it supports heterogeneity of the system. Their proposal worked to balance the load of nodes and scale back the demanded movement price the maximum amount as potential, whereas taking advantage of physical network locality and node heterogeneity. Leave space for vendors to boost and optimize a completely unique load balancing algorithms to modify the load-rebalancing drawback in large-scale, dynamic, and distributed file systems in clouds has been conferred during their work. Best algorithmic rule is commonly topology specific.

Tao Wang et al. [20] presented a new approach in which load balancing algorithm is combined with greedy algorithm; the scheme provided a better and efficient load balancing algorithm for various load cases (CLB). CLB algorithm uses entropy and the scope of invalid cache as the rating basis of load balancing outcome. In this paper, a cache-invalidation-scope model based on the improved consistent hash algorithm is taken into consideration. The overall objective was to improve the existing consistent hash algorithm and make it suitable for load balancing, besides, a cache invalidation-scope model is proposed providing a favourable load balancing method. The proposed algorithm increased the performance of key value caching system. They put forth a new scheme of load balancing for key value cache system in cloud environment in consideration with the effect of load balancing and the scope of invalid cache. Cache-invalidation-scope model is established to improve the effect of load balancing.

3. PROPOSED ALGORITHM

Let the set of chunk servers denoted as C in distributed file systems in a cloud, where the cardinality of C is $|C| = n$. Typically, n can be 1,000, 10,000, or more. Let the set of files is denoted as F stored in n chunk servers. Each file $f \in F$ is partitioned into a number of disjointed, fixed size chunks denoted by C_f . The load of a chunk server is proportional to the number of chunks hosted by the server.

Load distribution is the assigning of file chunks to the nodes in order to achieve even number of chunks on each node. Load exchange cost is the total number of file chunks migrated from one node to another. Rejoining cost is the number of hops a node traveled from its initial to new position in DHT network.

Let $\tilde{A}$ be the ideal number of Chunks that any Chunk server is required to manage in any system-wide load-balanced state

That is,

$$\tilde{A} = \sum_{f \in F} |C_f| \Big/ n \qquad (1)$$

Then, our load rebalancing algorithm aims to minimize the load imbalance factor in each chunk server i as follows:

$$\left\| L_i - \tilde{A} \right\| \quad (2)$$

Where L_i denotes the load of node *i* (the number of file chunks hosted by i) and $\|\bullet\|$ represents the absolute value function. The DHT lookup operation is performed to discover the file chunk implemented with Chord or pastry protocol. Nodes and chunks have unique id with adjacent node id are geometrically close using space-filling curve technique [21]. If node departs, its load is migrated to its successor and if node joins it allocates with load from its predecessor in DHT network.
Nodes in network implements Gossip based aggregation protocol [22] to collect the statuses of a sample of randomly selected nodes and built a vector S. A vector consists of node entries, node id and its load status. Using gossip-based aggregation protocol nodes can share its vector entries with neighbors.

The algorithm 1 and algorithm 2 detail waterfilling load balancing proposal as follows:

3.1 Algorithm 1: Seek (heaviest node j seek light node i to relieve its load)

Input: Vector S of s node entries, Average load A , Δ_L and Δ_U are system parameters such that $0 \le \Delta_L \le 1$ and $0 \le \Delta_U \le 1$

Output: Under loaded node i in S.

Step 1 Calculate the Average load A of s nodes in S as estimation of $\tilde{A}$.

Step 2 Calculate whether the node in S is under loaded or overloaded as follows:

A node is under loaded if Number of chunks < $(1-\Delta_L)A$

A node is over loaded if Number of chunks > $(1-\Delta_U)A$

Step 3 Let U_i be the sorted list of under loaded nodes with load l_i and O_j be the sorted list of overloaded nodes with load l_j in descending order in S.

Step 4 Calculate the load of top-O_j (node with maximum load) to shed on node i in U_i as follows:

$$l_o^{exch} = \left\| l_j - A \right\| \quad (3)$$

Step 5 Calculate the load exchange cost between each node in U_i and top-O_j as follows:

$$Cost_{obj}(i) = \sum_{j=1}^{m} Hopcount(j) \times l_o^{exch} + err(i) \quad (4)$$

Where Hop Count (j) is total the number of hopes the chunk has to travel from top O_j to i and m is number of nodes.

$$err(j) = \left| l_{i+1} + l_i - A \right| \quad (5)$$

Step 6 Find Minimum value of $Cost_{obj}(i)$ from list of costs and its linked node i.

Step 7 Output i

3.2 Algorithm 2: Migrate (file chunks migrates from node j to node i)

Input: An under loaded node i and an overloaded node j

Output: $l_j = A$

Step1 i migrate its locally hosted chunks to $i+1$.

Step2 i leaves the system and re-join the system as j's successor by having $i \leftarrow j+1$.

Step3 if $l_j > 2A$

then $t \leftarrow A$

else

$t \leftarrow l_j - A$

i Allocates t chunks with consecutive ids from j

Step4 j removes the chunks allocated to i and renames its id in response to the remaining chunks it manages.

Algorithm 1 seeks the light node in the system for shedding the load of heavy node using water-filling technique of calculating the total amount of water to spread over the uneven surface and amount of surface area to shed to bring it to unique water level.

Algorithm 2 migrate the load of the heavy load onto the searched light node in same way as water-filling technique spread the water to fill the particular area of uneven surface .

This proposal is distributed since all the nodes perform both the algorithm simultaneously without any global knowledge. Both the algorithm repeats iteratively for each heavy node in system by all the nodes to release the extra load in system. Load balancing algorithm is performed periodically and in parallel by nodes in the system and put their best effort to minimize the movement cost and time complexity in performing the algorithm in dynamic environment.

Example: Fig 1 shows a working example of our proposed work. There are 10 chunk servers N1, N2, N3, N4 ,N5 ,N6, N7, N8, N9, N10 and assume Δ_L and Δ_U be 0. Each node perform load balancing independently and we choose N1 as example to explain the work. Let N1 generate its sample vector with node entries {N3, N6, N8, N9}.Based on this sample calculate average load,

$$A = N1 + N3 + N6 + N8 + N9/5 \quad (6)$$

and finds the heavy node to share its load. Let N9 found itself as heavy node, then it search for all light node to shed its load, here N1 and N3 are light node. It calculate the load exchange cost with both light node and request the load having minimum load exchange cost to share its load. Here N3 has min exchange cost so it shed its load to its successor N4 and rejoin the system as successor of N9. N3 allocate minimum

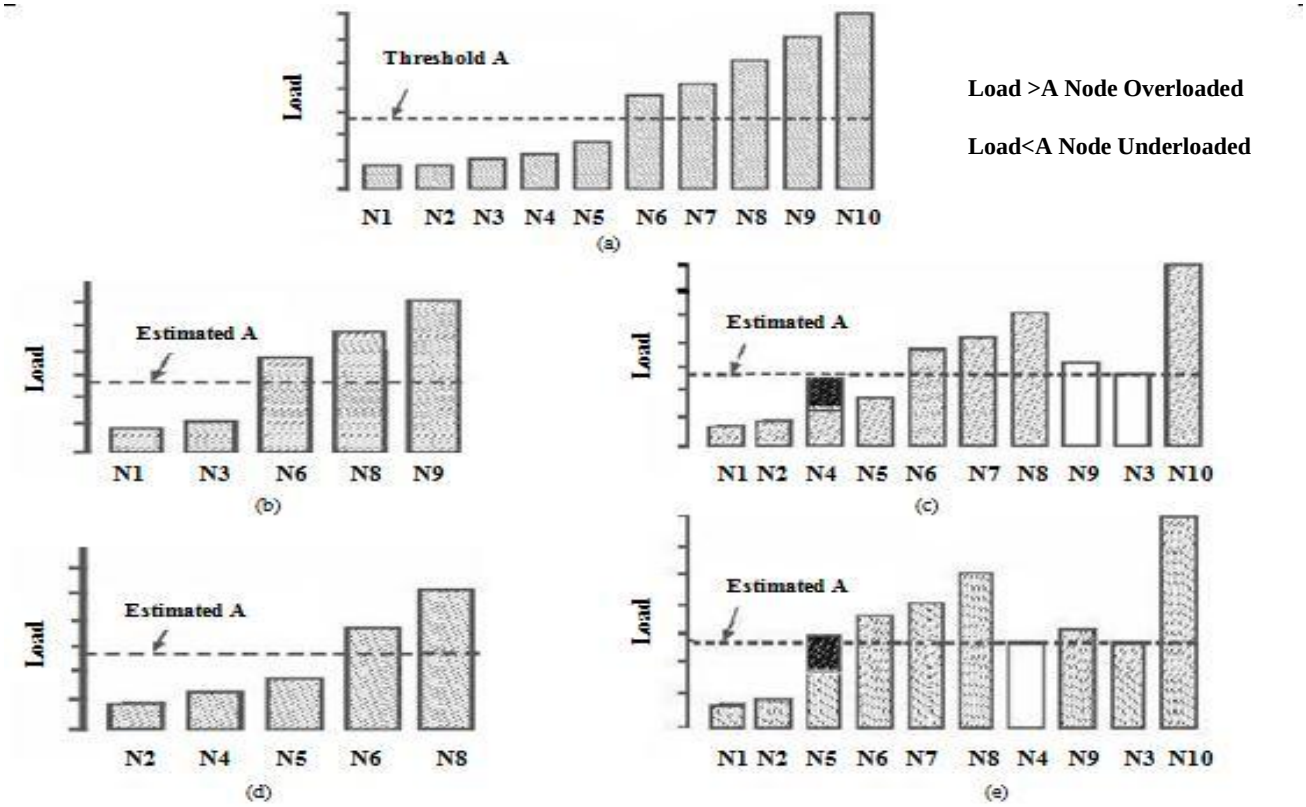

Fig1. An example illustrating proposed algorithm where (a) initial load on storage nodes N1, N2….N10 (b) N1 sample the load of N3, N6, N8, N9 (c) N 3 leaves and transfer its load to its successor N4 and rejoins as successor of N9 by allocating A chunks from N9 (d) N4 collect sample N2, N5, N7, N8 (e) N4 migrate and transfer its load to N5 and then rejoin as successor of N8 by allocating A chunks.

$\text{Min}\{L_{N9} - A, A\}$ chunks from N9. Suppose N4 is also performing load balancing with random samples {N2, N5, N7, N8} and N8 determine to shed its load with N4 and N4 transfer its load to N5 and rejoin as successor of N8.

4. EXPERIMENTAL RESULTS AND ANALYSIS

The performance of the proposed waterfilling algorithm is evaluated to rebalance the load for distributed file system through simulation on MATLAB 2013rb implemented on Intel (R) Core (TM)2 Duo CPU T6600@ 2.2 GHz, 2 GB RAM and 64 bit Window 7 home basic operation system. This proposal is carried out based on Chord DHT protocol [10] and gossip-based aggregation protocol [21]. To the best of our knowledge, no realistic workload s available .So, the number of nodes and file chunks in system may vary. In the default setting, number of nodes n= 50 and number of file chunks f = 500. System parameters Δ_L = 0.3 and Δ_U = 0.2. Sample vector S of each node consist of 10 sample nodes. The nodes simulated have identical capacity. The simulating result of load distribution for given workload is shown in Fig 2. Indicating uniformity of load distribution in waterfilling proposal is more than the previous approach.

The proposed waterfilling algorithm is compared with the previous distributed approach of load rebalancing for distributed file system. The simulation results in Fig 3.shows that the proposed waterfilling algorithm remarkably outperforms pervious distributed load rebalancing algorithm in terms of load imbalance factor. A variance near to zero indicates that the number of chunks on each node is identical. The waterfilling proposal converge in 60 iterations while previous approach took 200 iterations to complete the balancing with 6 times more variance value, indicating more the response time and less uniform distribution than the waterfilling approach.

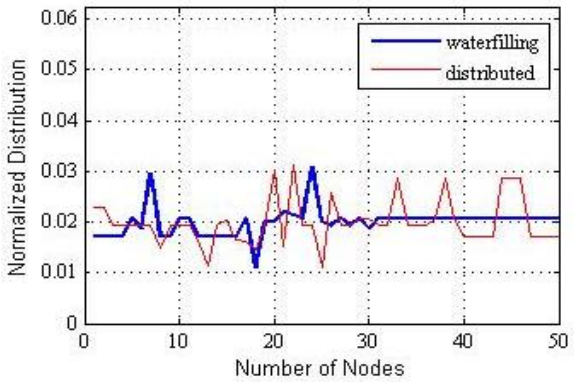

Fig. 2 Load Distribution

Fig 4 shows the load exchange cost at nth iteration of distributed algorithm and waterfilling load rebalancing algorithm. Distributed algorithm has 4 times more load exchange cost than waterfilling algorithm. The proposed waterfilling algorithm search for most physically closed underloaded node in Chord ring and prior calculate and select the consequent extra load on its successor resulting in minimum load exchange cost in this waterfilling design.

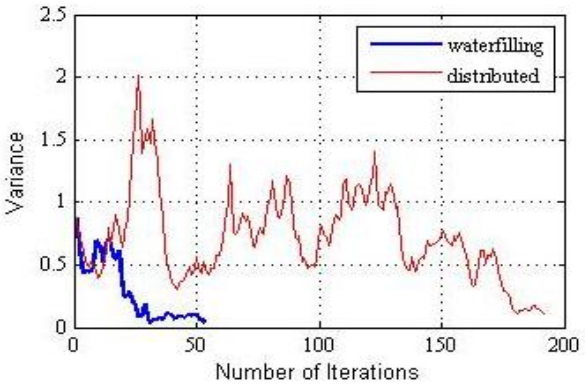

Fig. 3 Load Distribution Variance

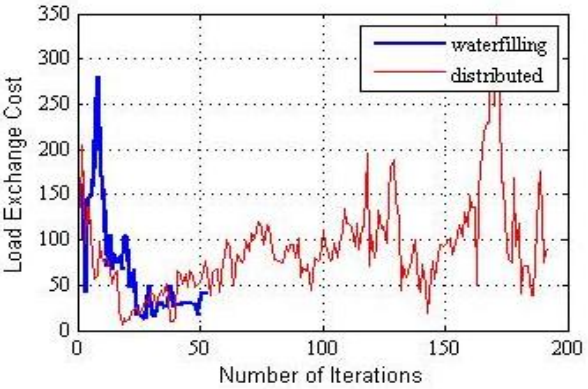

Fig 4 Load exchange cost at nth iteration

In contrast to distributed approach where top overload and top underloaded node exchange the load without considering physical network locality resulting in more movement cost. Fig 5 describes the result of load exchange cost up to nth iterations indicating that waterfilling proposal took 60 iterations to converge near to zero value with lesser load exchange cost for uniformity in contrast to previous algorithm.

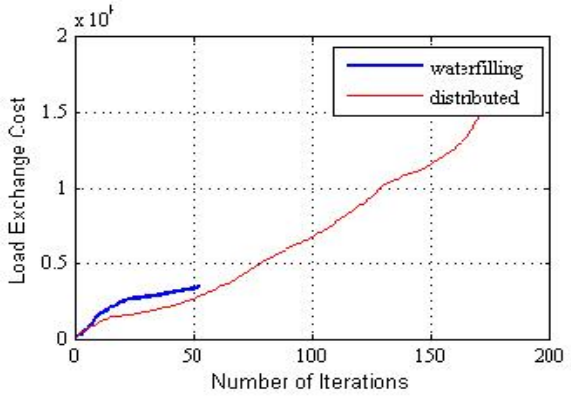

Fig 5 Load exchange cost after n iterations

Both the approaches depends on Chord DHT network where nodes may leave and join the network for load rebalancing thus increasing the overhead of rejoining operations. Fig 6 shows the rejoining cost at nth iteration in distributed approach and the proposed waterfilling algorithm. It shows that distributed approach has 0.5 times more rejoining operation than proposed waterfilling algorithm. Since waterfilling algorithm rejoins light nodes as successor of heavy node more precisely than distributed algorithm. Fig 7 describes the result of rejoining cost up to nth of iterations, showing convergence of waterfilling proposal in 60 iterations with less rejoining cost than previous algorithm still striving to converge in 200 iterations.

Both the distributed approach and proposed waterfilling algorithm have similar message overhead since both algorithm gather partial system information about their neighbors. For each experimental run calculation was done to evaluate the time elapsed to complete the load rebalancing algorithms, both for waterfilling proposal and distributed algorithm. Approx 10 experimental runs were performed for a given workload and calculated the average time required for executing a load rebalancing algorithm. It was find that on the

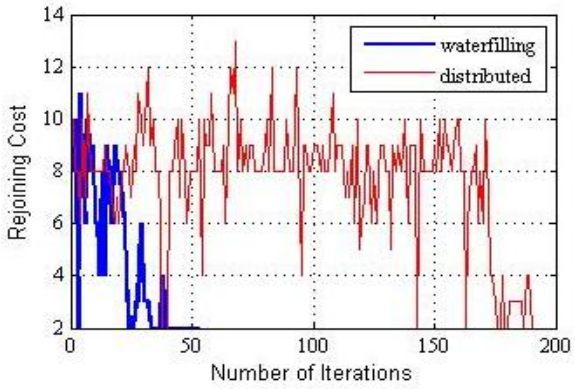

Fig 6 Rejoining cost at nth iteration

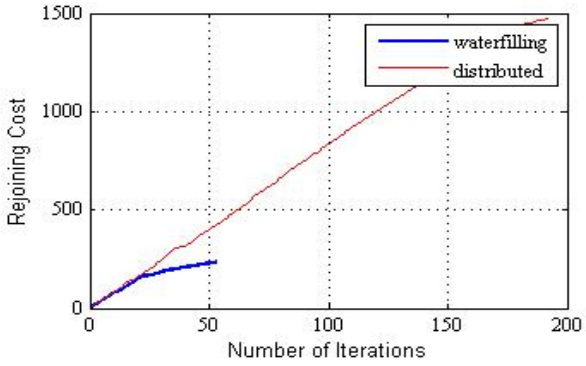

Fig 7 Rejoining Cost after n iterations

basis of the number of storage nodes and number of file chunks distributed, the proposed waterfilling algorithm performs well by less time consumed than previous algorithm because the number of iterations our proposal took to balance the load completely is much lesser than the previous algorithm resulting in less time to generate the results.

Three replicas of file chunks is assumed and average values of load exchange cost , rejoining cost, variance and time elapsed in experimental run has been calculated for 10 runs in waterfilling experiment. For different workload, number of file chunks nodes can be changed in waterfilling proposal for further experiments.

5. CONCLUSION AND FUTURE WORK

Load balancing is prone to more research and development in areas of distributed file systems in cloud. There are numerous load balancing approaches among which distributed approach is preferred most favorably by researchers due to its clarity and easy convergence. Despite of having these characteristics, Distributed approach has some shortcomings. In distributed approach, the node which is having maximum and minimum load is chosen for exchanging load. But, there is no surety that

these be the nodes nearest in the network. Moreover, the load on the successor light node is unpredictable and prone to increase the rejoining cost. Both the method, distributed and centralized ensures optimal load balancing but the accuracy is not remarkable; also this method cannot accommodate large number of file accesses in the real world workload.

The concept of the novel and enhanced load rebalancing based on water-filling model has been proposed. Here, the movement cost and rejoining cost are added to calculate the total load exchange cost. The main objective to reduce the movement cost of file chunks incurred during load balancing in the physical network by prior calculating the cost of exchanging the load and error (extra) load in future. Though this fusion of water-filling and distributed approach cannot tolerate possible workload in real world but it results in better load distribution as compared to the algorithms in file systems like HDFS, GFS, and all discussed in the previous section. In the future, load balancing can further be enhanced using node heterogeneity and replica of file chunk management into consideration in the panoptic environment of cloud.

6. REFERENCES

[1] J. Dean and S. Ghemawat Dec. 2004. "MapReduce: Simplified Data Processing on Large Clusters," In Proc. 6th Symp. Operating System Design and Implementation (OSDI'04), 137–150.

[2] "Hadoop Distributed File System", http:// hadoop. apache. org/hdfs/.

[3] "VMware", http://www.vmware.com/.

[4] "Hadoop Distributed File System", "Rebalancing Blocks,"http://developer.yahoo.com/hadoop/tutorial/module2.html#rebalancing.

[5] S. Ghemawat, H. Gobioff, and S.-T. Leung Oct. 2003, "The Google File System," In Proc. 19th ACM Symp. Operating Systems Principles (SOSP'03), 29–43.

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

[7] J. W. Byers, J. Considine, and M. Mitzenmacher Feb. 2003. "Simple Load Balancing for Distributed Hash Tables," In Proc. 1st Int'l Workshop Peer-to-Peer Systems (IPTPS'03), 80–87.

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

[9] Q. H. Vu, B. C. Ooi, M. Rinard, and K.-L. Tan, Jan. 1959. "Histogram-Based Global Load Balancing in Structured Peer-to-Peer Systems," IEEE Transactions on Knowledge and Data Engineering, 11(1), 34–39.

[10] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan Feb. 2003. "Chord: a Scalable Peer-to-Peer Lookup Protocol for Internet Applications," IEEE/ACM Transactions on Networks, 11(1), 17–21.

[11] A. Rowstron and P. Druschel Nov. 2001. "Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems," LNCS 2218, 161–172.

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

[13] Daniel Pérez Palomar, Member, and Javier Rodríguez Fonollosa Feb 2005. "Practical Algorithms for a Family of Waterfilling Solutions," IEEE Transactions on signals processing, 53(2), 686-695.

[14] A. Rao, K. Lakshminarayanan, S. Surana, R. Karp, and I. Stoica Feb. 2003. "Load Balancing in Structured P2P Systems," In Proc. 2nd Int'l Workshop Peerto- Peer Systems (IPTPS'02), 68–79.

[15] H. Shen and C.-Z. Xu June 2007. "Locality-Aware and Churn-Resilient Load Balancing Algorithms in Structured P2P Networks," IEEE Transactions on Parallel and Distributed Systems, 18(6), 849–862.

[16] Y. Zhu and Y. Hu Apr. 2005. "Efficient, Proximity-Aware Load Balancing for DHTBased P2P Systems," IEEE Transactions on Parallel and Distributed Systems, 16(4), 349–361.

[17] S. Surana, B. Godfrey, K. Lakshminarayanan, R. Karp, and I. Stoica Mar. 2006. "Load Balancing in Dynamic Structured P2P Systems," Performance Evaluation, 63(6), 217–240.

[18] Hung-Chang Hsiao, Hsueh-Yi Chung, HaiyingShen, and Yu-Chang Chao May 2013. "Load Rebalancing for Distributed File Systems in Clouds," IEEE Transactions on Parallel and Distributed Systems, 24(5), 951-961.

[19] Revathy R, A.Illayarajaa May. 2013. "Efficient Load Re Balancing Algorithm for Distributed File Systems," International Journal of Innovative Technology and Exploring Engineering, 2(6), 135-138.

[20] Tao Wang, Xin Lv, Fang Yang, Wenhuan Zhou, Rongzhi Qi, HuaiZhi Su Nov 2014 . "A Load balancing scheme for distributed key-value caching system in cloud environment," In Proc. 13th Symp. Distributed Computing and Applications to Business, Engineering and Science (DCABES'14), 63-67.

[21] H. Sagan 1994. "Space-Filling curves," Springer.

[22] M. Jelasity, A. Montresor, and O. Babaoglu Aug. 2005. "Gossip-Based Aggregation in Large Dynamic Networks," ACM Transactions on Computer Systems, 23(3), 219–252.