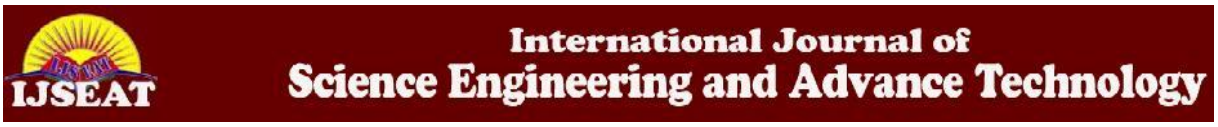
International Journal of Science Engineering and Advance Technology, IJSEAT, Vol 3, Issue 2, February - 2015
ISSN 2321-6905



Clustered Node Based Load Balancing In Distributed Environment

Padmaja.T[1], Suresh.N[2], Lakshmi Tulasi.R[3]
[1]Student of M.Tech (CSE),[2] Associate.Prof,[3]HOD,Professor
Department of Computer Science Engineering, Qis Institute of Technology, Ongole.

Abstract: Cloud computing having tremendous growth on recent years but it is not segregation on shared clouds. Distributed file systems are key building blocks for cloud computing applications based on the Map Reduce programming paradigm. In such file systems, nodes simultaneously serve computing and storage functions; a file is partitioned into a number of chunks allocated in distinct nodes so that Map Reduce tasks can be performed in parallel over the nodes. Data storage and communication which are to be done in huge amount, in such cases clouds are most provably used. "The cloud", also focuses on increasing the effectiveness of the public resources. Cloud resources are usually not only shared by multiple users but are also vigorously reallocated per demand. This can work for apportioning resources to users .But In the time of apportionment these are indeed .So In this paper we are introducing novel mechanism. We investigate to implement security provided for cloud computing and Evaluate the Quality of Service-QOS (Ex. Response Time) of whole system. In cloud computing one server controls number of sub servers, files, it can add, delete, and append dynamically Freight stabilization in the cloud computing surroundings has an imperative impact on the performance. Excellent freight stabilizing makes cloud computing more efficient and improves user satisfaction. In this paper we are presenting freight stabilizing techniques for cloud segregating.

Keywords: Freight Stabilizing, Cloud Segregating, Freight Stabilizing Models, Shared Cloud.

I Introduction:

The Distributed file systems an important issue in DHTs is load-balance the even distribution of items to nodes in the DHT. For load balance all DHTs make some efforts; The DHT address associated with each item is randomized with a “good enough” hash function and making each DHT node responsible for a balanced portion of the DHT address space. A prototypical example of this approach is Chord: Each node is responsible for only a small interval of the ring is nothing but “random” hashing of node to a ring while only a limited number of items land in the (small) ring interval owned by any node is nothing but random mapping of items. The cloud computing Current distributed hash tables do not evenly partition the address space into. Cloud computing (or cloud for short) is a compelling technology. In clouds, clients can dynamically allocate their resources on-demand without sophisticated deployment and management of resources. Key enabling technologies for clouds include the Map Reduce programming paradigm [1], distributed file systems (e.g., [3], [4]), virtualization (e.g., [4], [5]), and so forth. These techniques emphasize scalability, so clouds (e.g., [6]) can be large in scale, and comprising entities can arbitrarily fail and join while maintaining system reliability. For cloud computing applications which are based on the Map Reduce programming paradigm, distributed file systems are key building blocks. In such file systems, computing and storage functions between nodes are performed simultaneously; Partitioning of file into number of chunks and allocation of chunks to distinct nodes takes place. so over the nodes Map Reduce tasks can be performed in parallel. For example, consider a word count application. In large file, this word count application will count the number of distinct words and frequency of each unique word. In such an application, a cloud partitions the file into a large number of disjointed and fixed-size pieces (or file chunks) and assigns them to different cloud storage nodes (i.e., chunk servers). Each storage node (or node for short) then calculates the frequency of each unique word by scanning and parsing its local file chunks. In this paper, the load rebalancing problem in distributed file systems specialized for dynamic, large-scale and data-intensive clouds. (The terms “rebalance” and “balance” is interchangeable in this paper).There will be hundreds and thousands of nodes in such a large-scale cloud (and may reach tens of thousands in the future). Our objective is to allocate the chunks of files as uniformly as possible

among the nodes such that no node manages an excessive number of chunks. Additionally, we aim to maximize the network bandwidth by reducing the network traffic caused by rebalancing the loads of nodes as much as possible available to normal applications. Moreover, as failure is the norm, nodes are newly added to sustain the overall system performance [3], [4], resulting in the heterogeneity of nodes.

We can add, delete and update nodes dynamically for heterogeneity of the nodes. Heterogeneity of the nodes will increase the scalability and system performance. In Distributed File System the main functionalities of nodes is to serve computing and storage functions. In this paper, we introduced new load rebalancing algorithm in distributed file systems, aim to reduce network traffic (or movement cost) caused by rebalancing the loads of nodes as much as possible to increase the network bandwidth to cloud applications. For security in cloud computing, we maintained these files in encrypted format using cryptographic algorithms. Finally, for performance improvement, high speed, reducing time consistency, we calculate the response time of whole system.

II. Related Work

When you submit your Load balancing in cloud computing system [8] Ram Prasad Padhy, P Goutam Prasad Rao discussed on basic concepts of Cloud Computing and Load balancing and studied some existing load balancing algorithms, which can be applied to clouds. In addition to that, the closed-form solutions for minimum measurement and reporting time for single level tree networks with different load balancing strategies were also studied. The performance of these strategies with respect to the timing and the effect of link and measurement speed were studied. A Comparative Study into Distributed Load Balancing Algorithms for Cloud Computing [9] M. Randles, D.Lamb, and A.Taleb-Bendiab discussed on Distributed Load Balancing Algorithms are Honeybee Foraging Behavior, Biased Random Sampling, and Active Clustering. Honeybee

Foraging Behavior investigated a decentralized honeybee-based load balancing technique that is a nature-inspired algorithm for self-organization. Through local server actions it achieve global load balancing. With increased system diversity, performance of the system is enhanced but with an increase in system size, throughput is not increased. It is best suited for the conditions where the diverse population of service types is required. Biased Random Sampling investigated a distributed and scalable load balancing approach that uses random sampling of the system domain to achieve self-organization thus balancing the load across all nodes of the system. With high and similar population of resources, the performance of the system is improved thus by effective utilization of increased system resources results in increased throughput. With increased diversity, performance of the system will be degraded. Active Clustering investigated a self-aggregation Load Balancing Techniques that is self aggregation algorithm to optimize job assignments by connecting similar services using local re-wiring. With high resources, the performance of the system is enhanced, thereby using these resources effectively, throughput will increase. With increase in system diversity, performance of the system will be degraded. Game-theoretic static load balancing for distributed systems [10] Penmatsa and Chronopoulos discussed on static load balancing strategy based on game theory for distributed systems. And this work provides us with a new review of the load balance problem in the cloud environment. As an implementation of distributed system, the load balancing in the cloud computing environment can be viewed as a game. Load Balancing in Structured P2P Systems [11] A.Rao, K.Lakshminarayanan, S.Surana, R.Karp and I.Stoica based on the concept of virtual servers the many-to-many framework is to cope with the load imbalance in a DHT. In the many-to-many framework, directories are some dedicated nodes where light and heavy nodes register their loads. Matches between heavy and light nodes are computed by the directories and then respectively, request the heavy and light nodes to transfer and to receive designated virtual servers. Load Balancing in Dynamic Structured P2P Systems [12] S.Surana, B.Godfrey, K. Lakshmi narayanan, R. Karp and I.Stoica discussed on the many-to-many framework essentially reduces the load balancing problem to a centralized algorithmic problem. The directory nodes may thus become the performance bottleneck and single point of failure as the entire system heavily depends on the directory nodes. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications [13] Ion Stoic a, Robert Morris, David Karger, M. Frans Kaashoek and Hari Balakrishnan propose the notion of virtual servers as a means of improving load balance. By allocating log N virtual servers per physical node, Chord ensures that with high probability the number of objects on any node is within a constant factor of the average. However, these schemes assume that nodes are homogeneous; objects have the same size, and object IDS are uniformly distributed. Simple Load Balancing for Distributed Hash Tables [14] John Byers, Jeffrey Considine, and Michael Mitzenmache has proposed the use of the power of two choices paradigm to

achieve better load balance. However, this scheme was not analyzed or simulated for the case of heterogeneous object sizes and node capacities, and in any case is not prepared to handle a dynamic system of the kind which we have described. This is largely complementary to the work presented in this paper. Competitive Hill-Climbing Strategies for Replica Placement in a Distributed File System [15] J. R. Douceur and R. P. Watlenhofer have proposed algorithms for replica placement in distributed file systems which are similar in spirit with our algorithms. However, their primary goal is to place object replicas to maximize the availability in an un-trusted P2P system, while we consider the load balancing problem in a cooperative system.

III. Our Proposal

The chunk servers in our proposal are organized as a DHT network; that is, each chunk server implements a DHT protocol such as Chord [18] or Pastry [19]. A file in the system is partitioned into a number of fixed-size chunks, and "each "chunk has a unique chunk handle (or chunk identifier) named with a globally known hash function such as SHA1 [24]. The hash function returns a unique identifier for a given file's pathname string and a chunk index.

Each chunk server also has a unique ID. We represent the IDs of the chunk servers in V by 1n, 2n, 3n, · · · , nn; for short, denote the n chunk servers as 1, 2, 3, · · · , n. Unless otherwise clearly indicated, we denote the successor of chunk server i as chunk server i + 1 and the successor of chunk server n as chunk server 1. In a typical DHT, a chunk server i hosts the file chunks whose handles are within (i−1n , in], except for chunks whose handles are in (nn,1n] which are managed by chunk server n.To discover a file chunk, the DHT lookup operation is performed. In most DHTs, the average number of nodes visited for a lookup is O(log n) [18], [19] if each chunk server maintains log2 n neighbours, that is, nodes i + 2k mod n fork = 0, 1, 2, · · , log2 n − 1. Among the log2 n neighbours, the one i+20 is the successor of i. To look up a file with l chunks lookups are issued.

Our proposal uses DHTs for the following reasons:

A. As the arrivals, departures, and failures, simplifying the system provisioning and management, the chunk servers self-configure and self-heal in our proposal.

B. Migration of locally hosted chunks to its successor takes place if a node leaves.

C. Allocation of chunks whose IDs immediately precede the joining node from its successor to manage takes place if a node join.

Our proposal heavily depends on the arrival of nodes and departure operations to file chunks migration among nodes.

IV. Verification Framework Overview

Three different entities where system architecture involves:

Clients who have a large amount of data to be stored in multicloud and have the permissions to access and manipulate stored data. Cloud Service Providers (CSPs) who work together to provide data storage services have enough storages and computation resources. Trusted Third Party (TTP) is trusted to store verification parameters for integrity checking and offer public query services for these parameters.

This architecture Fig.1 has considered the existence of multiple CSPs to cooperatively store and maintain the data outsourced by client. To verify the integrity and availability of stored data in all CSPs,a cooperative PDP is used. For security of outsourced data, we use trusted third party as CSPs are not fully trusted by the Data Owner. Backup servers are also used by the system. The verification method is described as follows: Firstly, a client (data owner) uses the secret key to pre-process a file which consists of a group of blocks, produces a set of public verification information that is stored in TTP, CSPs will be transferred with the file and some verification tags, local copy may also deleted. Then client can issue a challenge for one CSP by using verification protocol to check the integrity and availability of outsourced data with respect to public information stored in TTP.

V Implementation Experimental Setup

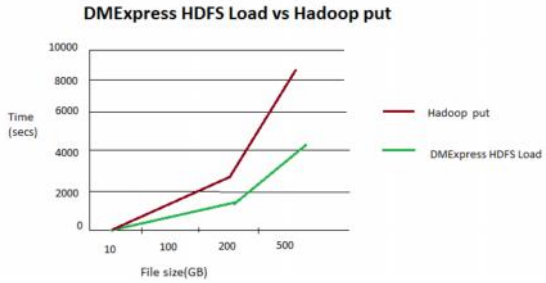
I have implemented the proposal in Hadoop HDFS 0.21.0, and assessed our implementation against the load balancer in HDFS. The implementation is demonstrated through a small-scale cluster environment consisting of a single, dedicated name node and 25 data nodes, each with Ubuntu10.10]. Specifically, the name node is equipped with Intel Core 2 Duo E7400 processor and 3 Gbytes RAM. As the number of file chunks in our experimental environment is small, the RAM size of the name node is sufficient to cache the entire name node process and the metadata information, including the directories and the locations of file chunks.

In the experimental environment, a number of clients are established to issue requests to the name node. The requests include commands to create directories with randomly designated names, to remove directories arbitrarily chosen, etc.

Particularly, the size of a file chunk in the experiments is set to 16 Mbytes. Compared to each experimental run requiring 20 to 60 minutes, transferring these chunks takes no more than 328 seconds ≈ 5.5 minutes in case the network bandwidth is fully utilized. The initial placement of the 256 files chunks

VI Performance Evalution

We run a varying number of players. The players move through the world according to a random waypoint model, with a motion time chosen uniformly at random from seconds, a destination chosen uniformly at random, and a speed chosen uniformly at random from (0, 360) pixels per second. The size of the game world is scaled according to the number of players. The dimensions are 640n _ 480n, where n is the number of players. All results are based on the average of 3 Experiments, with each experiment lasting 60 seconds. The experiments include the bent of log n sized LRU cache long pointers. The HDFS load balancer and our proposal. Our proposal clearly outperforms the HDFS load balancer. When the name node is heavily loaded (i.e., small M's), our proposal remarkably performs better than the



HDFS load balancer. For example, if M = 1%, the HDFS load balancer takes approximately 60 minutes to balance the loads of data nodes. By contrast, our proposal takes nearly 20 minutes in the case of M= 1%. Specifically, unlike the HDFS load balancer, our proposal is independent of the load of the name node. In particular, approximating the unlimited scenario is expensive, and the use of blog2 nc virtual peers as proposed in introduces a large amount of topology maintenance track but does not provide a very close approximation. Finally, we observe that while we are illustrating the most powerful instantiation of virtual peers, we are comparing it to the weakest choice model further improvements are available to us just by increasing d to 4.

VII. Conclusion

By using proposed idea where it consider physical network locality and heterogeneity of node, balancing the loads of nodes and demanded cost of movement can be reduced as much as possible. Competing algorithms through synthesized probabilistic distributions of file chunks are compared with proposed idea. Emerging distributed file systems in production systems strongly depend on a central node for chunk reallocation. Because of workload on central load balancer which is linearly scaled with the system size, leads to performance bottleneck and single point of failure. This dependence is deficient in large-scale, failure-prone environment. Centralized approach in a production system and a competing distributed method are compared with proposed algorithm. Load imbalance factor, movement cost, and algorithmic overhead are handled by developed algorithm efficiently. To securing the data, implemented the RSA algorithm. Examine the Performance measures of whole system.

VIII. Future Work

Use any other balancing algorithm strategy in cloud computing environment. In this paper only file sharing (file download & upload). In future sharing multimedia files (audio & video).This algorithm used between two various cloud computing environments. Use various cryptographic algorithms for security purposes.

References:

[1] Heiser J. What you need to know about cloud computing security and compliance, Gartner, Research, ID Number: G00168345, 2009.

[2] Seccombe A.., Hutton A, Meisel A, Windel A, Mohammed A, Licciardi A, (2009). Security guidance for critical areas of focus in cloud computing, v2.1. Cloud Security Alliance, 25 p.

[3] Mell P, Grance T (2011) The NIST definition of Cloud Computing. NIST, Special Publication 800–145, Gaithersburg, MD.

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

[5] J. Byers, J. Considine, and M. Mitzenmacher, Simple load balancing for distributed hash tables, in Proceedings of IPTPS, Berkeley, CA, Feb. 2003.

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

[7] Hadoop Distributed File System "Rebalancing Blocks," ,"http://developer.yahoo.com/hadoop/tutorial/module 2.html#r ebalancing, 2012.

[8] Ram Prasad Padhy (107CS046), PGoutam Prasad Rao (107CS039)."Load balancing in cloud computing system" Department of Computer Science and Engineering National Institute of Technology, Rourkela Rourkela-769 008, Orissa, India May, 2011.

[9] M. Randles, D. Lamb, and A. Taleb-Bendiab, —A Comparative Study into Distributed Load Balancing Algorithms for Cloud Computing‖, Proceedings of 24th IEEE International Conference on Advanced Information Networking and Applications Workshops, Perth, Australia, April 2010, pages 551-556.

[10] S.Penmatsa and T.Chronopoulos, Game-theoretic static load balancing for distributed systems, Journal of Parallel and Distributed Computing, vol.71, no.4, pp.537-555, Apr. 2011.
[11] A.Rao, K.Lakshmi narayanan, S.Surana, R.Karp and I.Stoica, Load Balancing in Structured P2P Systems‖, Proc. Second Int„l Workshop Peer-to-Peer Systems (IPTPS „02), pp. 68-79, Feb. 2003.
[12] S.Surana, B.Godfrey, K.Lakshminarayanan, R.Karp and I.Stoica,—Load Balancing in Dynamic Structured P2P Systems,‖ Performance Evaluation,vol.63, no. 6, pp. 217-240, Mar. 2006.
[13] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek and Hari Balakrishnan. —Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications.‖ in Pmc. ACM SIGCOMM. San Diego, 2001, pp. 149-160.
[14] John Byers, Jeffrey Considine, and Michael Mituenmacher, —Simple Load Balancing for Distributed Hash Tables.in Pmc.IPTfS, Feb. 2003.
[15] J. R. Douceur and R. P. Watlenhofer, "Competitive Hill-Climbing Strategies for Replica Placement in a Distributed File System,‖ in Pmc. DISC,2001.
[16] R.L. Rivest, A. Shamir, and L. Adleman. "A Method for Obtaining Digital Signatures and Public key Cryptosystems", Communications of the ACM, 21(2), 120-126, February 1978.
[17] RSA Laboratories, High Speed RSA Implementation, RSA Data Security, November 1994.

Authors:

Padmaja.T is a student of Computer Science Engineering from QIS Institute of Technology,Presently pursuing M.Tech (CSE) from this college. She received B.Tech from JNTUK in the year of 2012.

Suresh.N is a Associate Professor of QIS Institute of Technology,Ongole. He received M.Tech in Computer Science Engineering from JNTU Kakinada University.He gained 12years Experience on Teaching . He is a good Researcher in Image Processing, Algorithms, Computer Networks. He attended Various National and International Workshops and Conferences.

Lakshmi Tulasi.R is a Professor, H.O.D of QIS Institute of Technology, Ongole. She received M.Tech from JNTUCEA. she is pursuing Ph.D at JNTUH.She is a good Researcher in semantic web, Computer Networks. She attended Various National and International Workshops and Conferences.