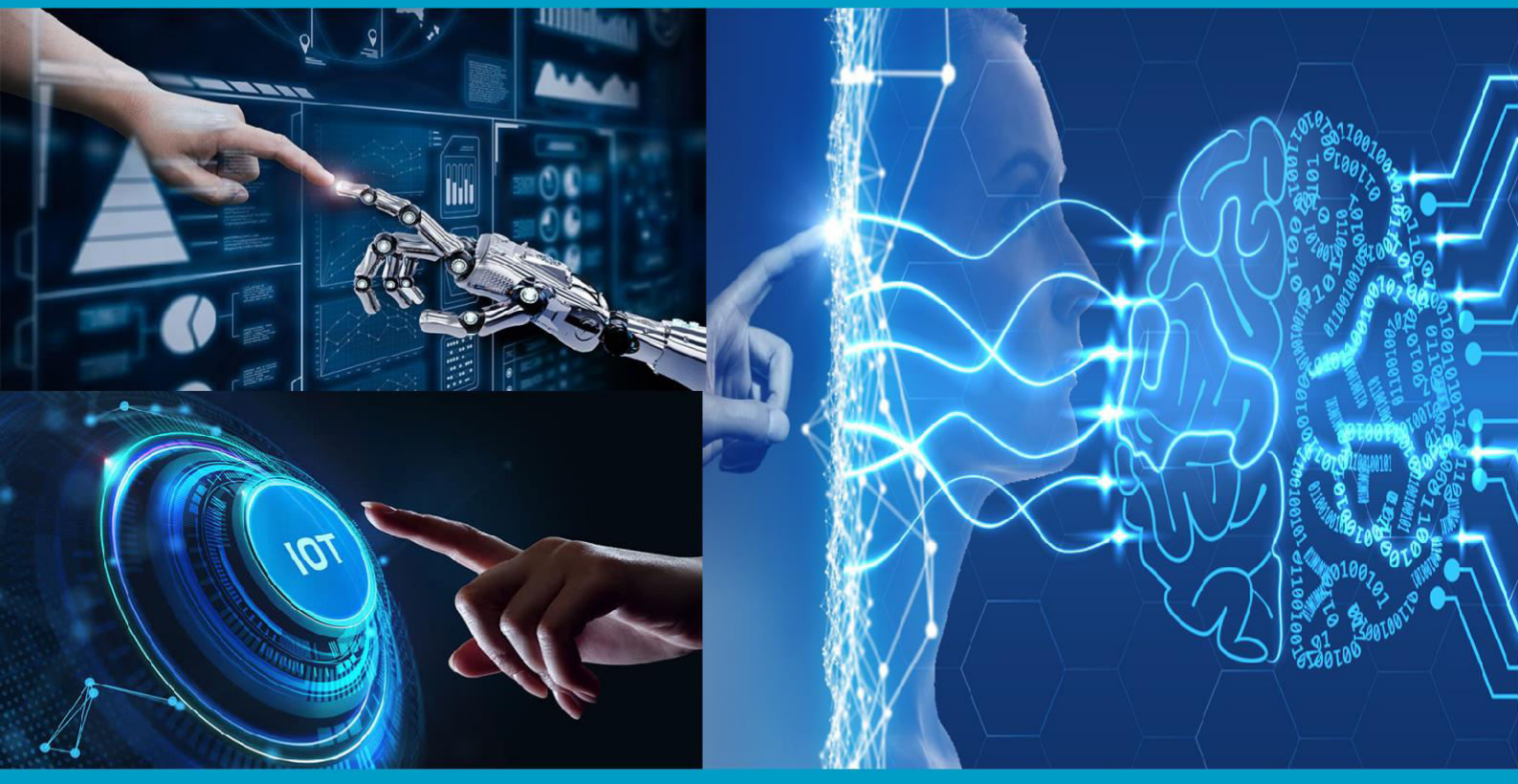
IJMRSETM

e-ISSN: 2395 - 7639



INTERNATIONAL JOURNAL OF MULTIDISCIPLINARY RESEARCH

IN SCIENCE, ENGINEERING, TECHNOLOGY AND MANAGEMENT

Volume 11, Issue 4, April 2024

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

Impact Factor: 7.802

International Journal of Multidisciplinary Research in Science, Engineering, Technology & Management (IJMRSETM)

| ISSN: 2395-7639 | www.ijmrsetm.com | Impact Factor: 7.802 | A Monthly Double-Blind Peer Reviewed Journal |

| Volume 11, Issue 4, April 2024 |

Real-Time Detection Schemes for Memory Dos M-Dos Attacks on Cloud Computing Applications

Dr.T.Aravind, Mohanraj V, Ilayasuriya A, Mukesh B

Assistant Professor, Department of Computer Science, Muthayammal Engineering College (Autonomous), Kakkaveri, Rasipuram, Tamil Nadu, India

Department of Computer Science, Muthayammal Engineering College (Autonomous), Kakkaveri, Rasipuram, Tamil Nadu, India

ABSTRACT: Cloud computing is emerging as a new paradigm of large scale distributed computing. Load balancing is one of the main Challenges in Cloud computing which is required to distribute the dynamic workload evenly across all the nodes. In the cloud storage, Load balancing is a key issue. The Map reducing task can be performed parallel over the nodes. The file chunks are not distributed uniformly as possible among the nodes. Emerging distributed systems in production system strongly depends on a central node for chunk reallocation. It would consume a lot of cost to maintain load information. Proper load balancing aids in minimizing resource consumption. This concludes that all the existing techniques mainly focus on reducing overhead, service response time and improving performance etc. various parameters are also identified, and these are used to compare the existing techniques. This paper proposed for centralized server is change in to the decentralized server using Map reducing task.

KEYWORDS: load balancing algorithm, load balancing challenges, cloud computing, distributed computing

I. INTRODUCTION

During the last several decades, dramatic advances in computing power, storage, and networking technology have allowed the human race to generate, process, and share increasing amounts of information in dramatically new ways. As new applications of computing technology are developed and introduced, these applications are often used in ways that their designers never envisioned. New applications, in turn, lead to new demands for even more powerful computing infrastructure. It is now possible to assemble very large, powerful systems consisting of many small, inexpensive commodity components because computers have become smaller and less expensive, disk drive capacity continues to increase, and networks have gotten faster. Such systems tend to be much less costly than a single, faster machine with comparable capabilities. Software challenges also arise in this environment because writing software that can take full advantage of the aggregate computing power of many machines is far more difficult than writing software for a single, faster machine Regardless of the exact definition used, numerous companies and research organizations are applying cloud-computing concepts to their business or research problems including Google, Amazon, Yahoo, and numerous universities.

A Cloud computing is emerging as a new paradigm of large scale distributed computing. It has moved computing and data away from desktop and portable PCs, into large data centre's [1]. It provides the scalable IT resources such as applications and services, as well as the infrastructure on which they operate, over the Internet, on pay-per-use basis to adjust the capacity quickly and easily. It helps to accommodate changes in demand and helps any organization in avoiding the capital costs of software and hardware [2] [3]. Thus, Cloud Computing is a framework for enabling a suitable, on-demand network access to a shared pool of computing resources (e.g. networks, servers, storage, applications, and services).These resources can be provisioned and de-provisioned quickly with minimal management effort or service provider interaction. This further helps in promoting availability [4]. Due to the exponential growth of cloud computing, it has been widely adopted by the industry and there is a rapid expansion in data centres.

Load balancing in computer networks is a technique used to spread workload across multiple network links of computers [2]. It facilitates networks and resources by providing a maximum throughput with minimum time, thus it helps to improve performance by optimally using available resources and helps in minimizing latency and response time. Load balancing is achieved by using multiple resources that is, multiple servers that are able to fulfill a request or by having multiple paths to a resource. Load balancing helps to achieve a high user satisfaction and resource utilization. When one or more components of any service fail, load balancing facilitates continuation of the service by implementing fair-over, that is, it helps in provisioning and deprovisioning of instances of applications without fail. It also ensures that every computing resource is distributed efficiently and fairly [5]

| ISSN: 2395-7639 | www.ijmrsetm.com | Impact Factor: 7.802 | A Monthly Double-Blind Peer Reviewed Journal |

| Volume 11, Issue 4, April 2024 |

Fig.1 Cloud Computing

Consumption of resources and conservation of energy is not always a prime focus of discussion in cloud computing. However, resource consumption can be kept to minimum with proper load balancing which not only helps in reducing costs but making enterprise greener. Scalability, one of the very important features of cloud computing, is also enabled by load balancing. Hence, improving resource utility and the performance of a distributed system in such a way will reduce the energy consumption and carbon footprints to achieve Green computing [1]. The objective and motivation of this survey is to provide a analytic survey of existing load balancing techniques in cloud computing. In this paper, we are interested in studying the load rebalancing problem in distributed file systems specialized for large-scale, dynamic and data-intensive clouds. (The terms “rebalance” and “balance” are interchangeable in this paper.) Such a large-scale cloud has hundreds or thousands of nodes (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 reduce network traffic (movement cost) reduce network traffic (or movement cost) caused by rebalancing the loads of nodes as much as possible to maximize the network bandwidth available to normal application.

Applications need information on both when and how to rebalance;
The three load balancing steps are:
1. Evaluate the imbalance;
2. Decide how to balance if needed;
3. Redistribute work to correct the imbalance.

We address the first two requirements and derive complete information on how to perform the third; the application must be able to redistribute its work units as instructed by our framework (a requirement also imposed by partitioners. Our load model couples abstract application information with scalable load measurements. We derive actionable load metrics to evaluate the accuracy of the information. Our load model evaluates the cost of correcting load imbalance with specific load balancing algorithms. We use it to select the method that most efficiently balances a particular scenario. We demonstrate this methodology on two large-scale production applications that simulate molecular dynamics and dislocation dynamics. Overall, we make the following contributions.

Load rebalances in the distributed file system carried out using the map reducing task in cloud which helps in arranging files in nodes of every chunk i.e. stores the files in related nodes of the chunks. Objective of this project is to allocate the chunks of files as uniformly as possible among the nodes such that no node manages an excessive number of chunks and also to reduce network traffic and maximize the network bandwidth available to normal applications. Using the distributed file system, arranging the file system in a cloud: that is the file chunks are no distributed uniformly as possible among the nodes because the load is put under workload that is linearly scaled with the system and to increase the performance of the transformation of the file. This performance of the proposal is implemented to be used in the clustered environment.

II. SYSTEM OVERVIEW

The load rebalancing problem in distributed file systems specialized for large-scale, dynamic and data-intensive clouds. Suggest offloading the load rebalancing task to storage nodes by having the storage nodes balance

| ISSN: 2395-7639 | www.ijmrsetm.com | Impact Factor: 7.802 | A Monthly Double-Blind Peer Reviewed Journal |

| Volume 11, Issue 4, April 2024 |

their loads spontaneously. The storage nodes are structured as a network based on distributed hash tables (DHTs) discovering a file chunk can simply refer to rapid key lookup in DHTs, given that a unique handle is assigned to each file. DHTs enable nodes to self-organize and -repair while constantly offering lookup functionality in node dynamism, simplifying the system provision and management. Present a load rebalancing algorithm for distributing file chunks as uniformly as possible and minimizing the movement cost as much as possible. Nodes perform their load-balancing tasks independently without synchronization or global knowledge regarding the system. This project not only takes advantage of physical network locality in the reallocation of file chunks to reduce the movement
Cost but also exploits capable nodes to improve the overall system performance. Algorithm reduces overhead introduced to the DHTs as much as possible. Additionally, our load-balancing algorithm exhibits a fast convergence rate. The Architecture can be shown in figure 2.

A. Storage Node Creation

In cloud server simultaneously create node, serve computing and storage functions; a file is partitioned into a number of chunks allocated in distinct nodes. In this module, 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 then calculates the frequency of each unique word by scanning and parsing its local file chunks. User creates a storage node after successful register our account.

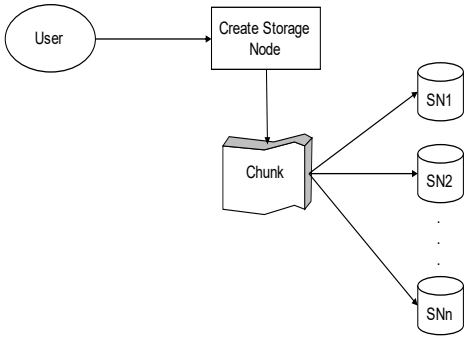


Fig.2 Storage Node Creation

B. Distributed Hash Table

DHTs guarantee that if a node leaves, then its locally hosted chunks are reliably migrated to its successor; if a node joins, then it allocates the chunks whose IDs immediately precede the joining node from its successor to manage. Our proposal heavily depends on the node arrival and departure operations to migrate file chunks among nodes. Interested readers are referred to for the details of the self-management technique in DHTs. While lookups take a modest delay by visiting n nodes in a typical DHT, the lookup latency can be reduced because discovering the l chunks of a file can be performed in parallel. Proposal is independent of the DHT protocols. To further reduce the lookup latency, can adopt state-of-the-art DHTs such as Amazon's Dynamo in that offer one-hop lookup delay.

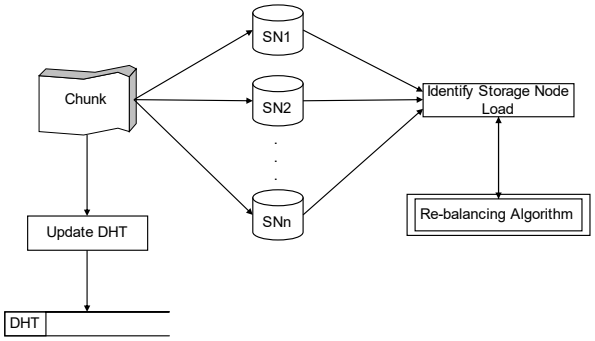


Fig.3 Identify Storage Node

| ISSN: 2395-7639 | www.ijmrsetm.com | Impact Factor: 7.802 | A Monthly Double-Blind Peer Reviewed Journal |

| Volume 11, Issue 4, April 2024 |

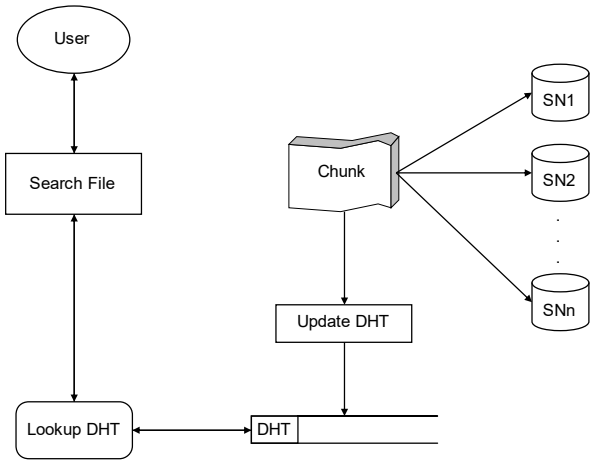


Fig.4 Distributed Hash Table

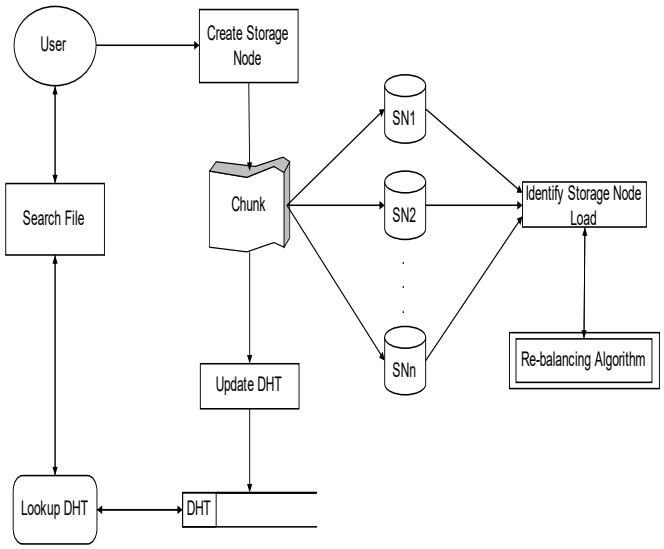


Fig.5 Overall Data Flow Diagram

C. Distributed Load Balancing

A large-scale distributed file system is in a load-balanced state if each chunk server hosts no more than A chunks. In our proposed algorithm, each chunk server node i first estimate whether it is under loaded (light) or overloaded (heavy) without global knowledge. A node is light if the number of chunks it hosts is smaller than the threshold. In contrast, a heavy node manages the number of chunks. In the following discussion, if a node i departs and rejoins as a successor of another node j, then represent node i as node j+ 1, node j's original successor as node j + 2, the successor of node j's original successor as node j + 3, and so on. For each node $I \in V$, if node i is light, then it seeks a heavy node and takes over at most A chunks from the heavy node.

| ISSN: 2395-7639 | www.ijmrsetm.com | Impact Factor: 7.802 | A Monthly Double-Blind Peer Reviewed Journal |

| Volume 11, Issue 4, April 2024 |

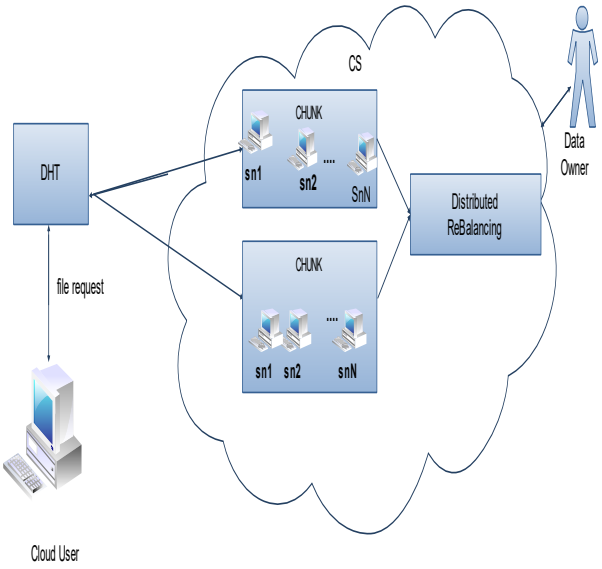


Fig.6 Load Rebalancing Architecture Diagram

D. File Distribution

A DHT node is an overlay on the application level. The logical proximity abstraction derived from the DHT does not necessarily match the physical proximity information in reality. That means a message traveling between two neighbors in a DHT overlay may travel a long physical distance through several physical network links. In the load-balancing algorithm, a light node i may rejoin as a successor of a remote heavy node j. Then, the requested chunks migrated from j to i need to traverse several physical network links, thus generating considerable network traffic and consuming significant network resources (i.e., the buffers in the switches on a Communication path for transmitting a file chunk from a source node to a destination node) and distribute the files for requesting users efficiently and effectively.

III. PRELIMINARY RESULTS

Some preliminary results on load rebalancing are presented. In the following subsections contains DHT formulation, File chunks, and then map reducing task.

A. DHT formulation

Distributed hash table is given unique identity of each every file. So files are stored in one hash table. The hash table performed given results The storage nodes are structured as a network based on distributed hash tables (DHTs), DHTs enable nodes to self-organize and repair while constantly offering lookup functionality in node dynamism, simplifying the system provision and management.. For each entity, it provides many web pages. File chunks size can be taken as example.

B. File Chunks

The distribution of chunks after performing the HDFS loads balancer.

File chunks =500
Data nodes =20
=500/20=25.0

C. Map reducing Task

The map reducing task is separated for all data's. This result is shown in figure7.all chunks are stored in goggle apps engine. This paper using map reducing task evaualate the balance and redistribute the balancing nodes are solving.

| ISSN: 2395-7639 | www.ijmrsetm.com | Impact Factor: 7.802 | A Monthly Double-Blind Peer Reviewed Journal |

| Volume 11, Issue 4, April 2024 |

Table 1
Comparison of file chunk size

File Chunks Size	Data nodes
250	25
300	30
400	40
450	20
500	20

IV. RESULTS

The entire system is implemented in .Net using eclipse Platform. In computer programming, eclipse is an integrated development environment comprising a base work phase and an extensible plug in system for customizing the environment. It is written mostly in java. It can be used to develop application in java, and by means of various plug-ins other programming languages including Ada, C, C++, COBOL, FORTRAN, Haskell, JavaScript, lasso, Perl and Erlang.It can also be used to develop packages for the software mathematics. Development environment includes the eclipse data development tools for java and scala .Eclipse CDT for C/C++ and Eclipse PDT for PHP among others. The initial codebase originated from IBM VisualAge.The Eclipse software development tool is mean for java developers. User can extend its abilities by installing plug-ins written for the Eclipse Platform. Such as development toolkit for other programming languages, other programming languages, and can write contribute their own plug-in modules. Java contains many JAR file. JAR files help to extract the plain text from the web pages. The result is shown in fig. 7.

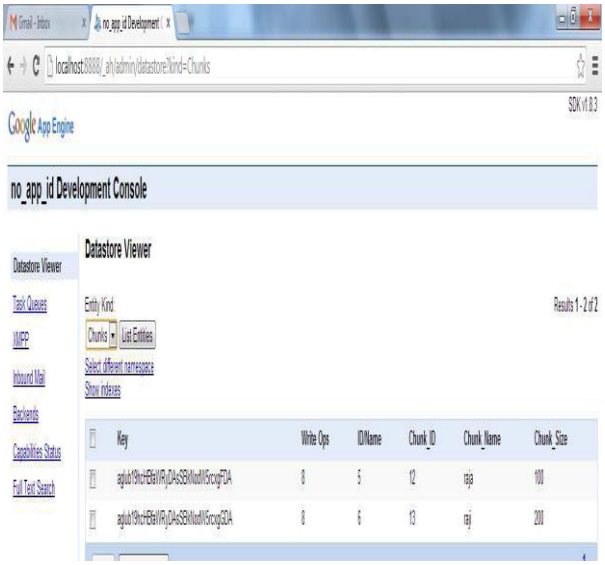


Fig. 7 Chunks Stored in Google AppsEngine

V. CONCLUSION

A load-balancing algorithm to deal with the load rebalancing problem in large-scale, dynamic, and distributed file systems in clouds has been presented in this project. Proposal strives to balance the loads of nodes and reduce the demanded movement cost as much as possible, while taking advantage of physical network locality and node heterogeneity. In the absence of representative real workloads (i.e., the distributions of file chunks in a large-scale storage system) in the public domain, To have investigated the performance of our proposal and compared it against competing algorithms through synthesized probabilistic distributions of file chunks. The synthesis workloads stress test the load-balancing algorithms by creating a few storage nodes that are heavily loaded.

Proposal is comparable to the centralized algorithm in the Hadoop HDFS production system and dramatically outperforms the competing distributed algorithm in terms of load imbalance factor, movement cost, and algorithmic overhead. Particularly, our load-balancing algorithm exhibits a fast convergence rate. The efficiency and effectiveness

| ISSN: 2395-7639 | www.ijmrsetm.com | Impact Factor: 7.802 | A Monthly Double-Blind Peer Reviewed Journal |

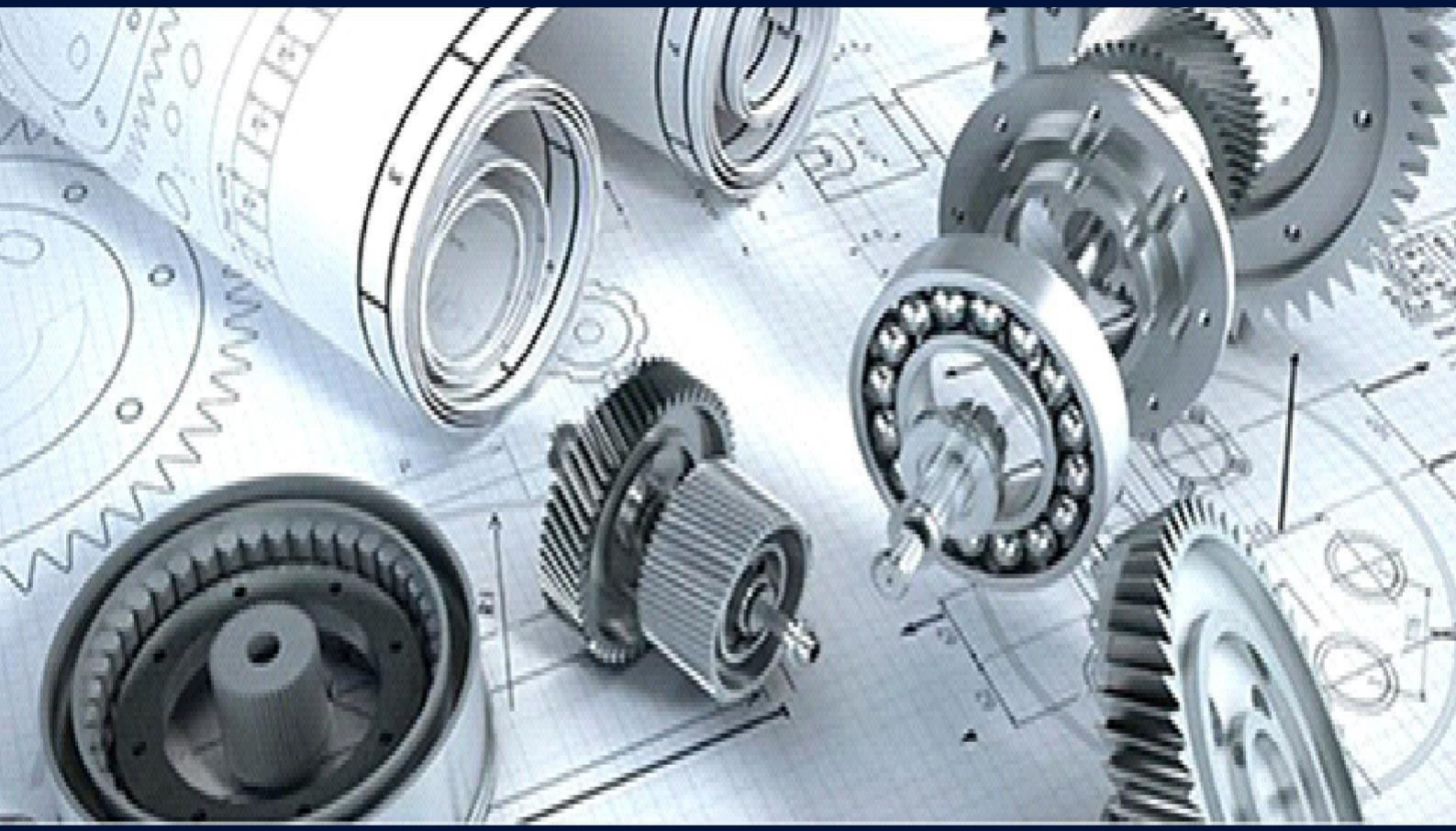
| Volume 11, Issue 4, April 2024 |

of our design are further validated by analytical models and a real implementation with a small-scale cluster environment. Consider a DHT with an ordered id space I with size N =|j| and a branching factor B such that log N is integral. The branching factor is used by each chunk to construct its routing table.

To provide consistency with previous work, reconsider Chord as a tree-based routing DHT. It is straightforward to show that Chord finger tables are constructed like tree-based routing tables.

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] A.W. M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain, "Content-based image retrieval at the end of the early years", *IEEE Transaction Pattern Analysis Machine Intelligence*, Antony Rowstron and Peter Druschel, "Pastry: Scalable, Distributed Object Location and Routing for Large-scale Peer-to-Peer Systems," in *Proc. Middleware*, 2001.
[3] John Byers, Jeffrey Considine, and Michael Mitzenmacher,
[4] "Simple Load Balancing for Distributed Hash Tables," in *Proc. IPTPS*, Feb. 2003
[5] David Karger and Matthias Ruhl, "New Algorithms for Load
[6] Balancing in Peer-to-Peer Systems," Tech. Rep. MIT-LCS-TR-911, MIT LCS, July 2003.
[7] J. Westbrook, "Load balancing for response time," in *EuropeanSymposium on Algorithms*, 1995, pp. 355–368.
[8] Micah Adler, Eran Halperin, Richard M. Karp,and Vijay V. Vazirani. A Stochastic Process on the Hypercube with Applications to Peer-to-Peer Networks. In *Proceedings STOC*, pages 575–584, 2003.
[9] Tanveer Ahmed, Yogendra Singh, Analytic study of load balancing techniques using tool cloud analyst.
[10] Zenon Chaczko, Venkatesh Mahadevan, Shahrzad Aslanzadeh and Christopher Mcdermid, Availabilty and load balancing in cloud computing, 2011 InternationalConference on Computer and Software Modeling, IPCSIT vol.14 (2011) ACSIT Press, Singapore
[11] Giuseppe Valetto, Paul Snyder, Daniel J. Dubois, Elisabetta DiNitto and Nicolo M. Calcavecchia, A self-organized load balancing algorithm for overlay based decentralized service networks
[12] Nidhi Jain Kansal, Inderveer Chana, Cloud Load balancing techniques: A step towards green computing, IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 1, No 1, January 2012.
[13] 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," in Proc. 21st ACM Symp.
[14] Hadoop Distributed File System, "Rebalancing Blocks," http://developer.yahoo.com/hadoop/tutorial/module2.html#rebalancing.
[15] HDFSFederation,http://hadoop.apache.org/common/docs/r0.23.0/hadoop-yarn/hadoop-yarn-site/Federation.htm
[16] D. Karger and M. Ruhl, "Simple Efficient Load Balancing Algorithms for Peer-to-Peer Systems," in Proc. 16th ACM Symp. Parallel Algorithms and Architectures (SPAA'04), June 2004, pp. 36–43.



INTERNATIONAL STANDARD SERIAL NUMBER INDIA

INTERNATIONAL JOURNAL OF MULTIDISCIPLINARY RESEARCH

IN SCIENCE, ENGINEERING, TECHNOLOGY AND MANAGEMENT

+91 99405 72462 +91 63819 07438 ijmrsetm@gmail.com