ISSN (Online) 2278-1021
ISSN (Print) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 4, Issue 7, July 2015

Fault Tolerance and Load Balancing algorithm in Cloud Computing: A survey

Sushil Kumar[1], Deepak Singh Rana[2], Sushil Chandra Dimri[3]

Assistant Professor, Department of Computer Application, Graphic Era University, Dehradun, India[1]

Assistant Professor, Department of Computer Science, Graphic Era Hill University, Dehradun, India[2]

Professor, Department of Computer Application, Graphic Era University, Dehradun, India[3]

Abstract: Cloud computing is developing as a new standard for deploying, organizing, and accessing large scale distributed computing applications over the network. In cloud computing, fault tolerance is a major problem and one of the metric which consider being most important since the resource failure affects job execution, throughput, response time and performance of system and network. Fault tolerance in load balancing is one of the main challenges in cloud computing, which is required to distribute the workload equally across all the nodes, detect the fault and remove fault from the network and share workload to all the nodes to increase the performance of cloud network. The load is an amount of work that a computation system performs, which can be classified as network load, storage capacity, memory capacity and CPU load. This paper describes a survey on fault tolerance, fault tolerance techniques, load balancing algorithm and load balancing schemes including fault tolerance in a cloud environment.

Keywords: Cloud computing, Load Balancing, Fault Tolerance, Load balancing, Static load balancing, Dynamic load balancing algorithm.

I. INTRODUCTION

"A cloud computing is a set of network enabled services, providing scalable, QoS guaranteed, normally personalized, inexpensive computing platforms on demand, which could be accessed in a simple and pervasive way"[1].
The US National Institute of Standards and Technology (NIST) has published a working definition [2] as "a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., Networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction".

This definition describes Cloud Computing using [3]:

- **Five characteristics:** on-demand self-service, broad network access, resource pooling, rapid elasticity, and measured service.
- **Four deployment models:** private Clouds, community Clouds, public Clouds, and hybrid Clouds.
- **Three service models:** Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS).

Figure 1 shows the architecture of cloud which contains cloud services, resources with examples.

The remainder of the paper is organized as follows: Section II contains the introduction of fault tolerance and its techniques. Section III describes the load balancing and some existing load balancing algorithms.

Some existing load balancing schemes, including fault tolerance algorithms are analyzed in section IV and the final conclusion is given in section V.

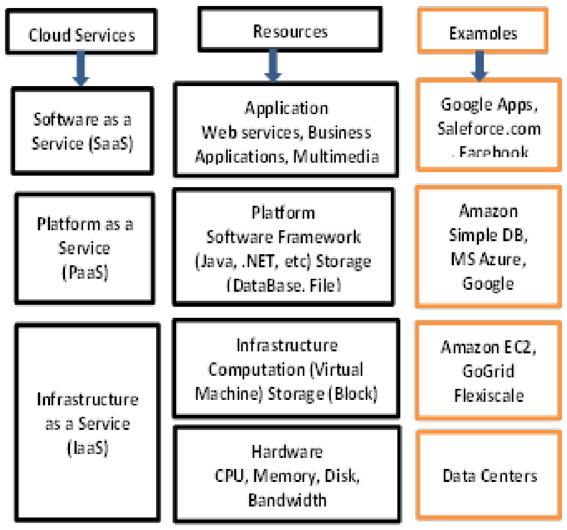
Cloud Services
Resources
Examples
Software as a Service (SaaS)
Application Web services, Business Applications, Multimedia
Google Apps, Salesforce.com, Facebook
Platform as a Service (PaaS)
Platform Software Framework (Java, .NET, etc) Storage (DataBase, File)
Amazon Simple DB, MS Azure, Google
Infrastructure as a Service (IaaS)
Infrastructure Computation (Virtual Machine) Storage (Block)
Amazon EC2, GoGrid Flexiscale
Hardware CPU, Memory, Disk, Bandwidth
Data Centers

Fig 1. Cloud Computing Architecture

II. FAULT TOLERANCE

Fault tolerance is one of the metric which is considered to be most important since the resource failure affects job execution, throughput, response time and performance of the system. Thus a fault tolerance policy is required to detect failures, resolve these failures, thus improving performance metric [4]. A load balancing algorithm should have the fault tolerance ability, means it should perform uniform load balancing in spite of failure of arbitrary node or link. Fault tolerance is a major concern to guarantee availability of critical services as well as application execution [4][5].

Reactive fault tolerance: Reactive fault tolerance policies reduce the effect of failures on application execution when failure effectively occurs [5].

Proactive fault tolerance: Proactive fault tolerance policies are to avoid recovery from faults, errors and failures by predicting them and proactively replace the suspected components with other working components [5].

A. Existing Fault Tolerance Techniques in Cloud Computing:

In cloud, fault can be categorized on the basis of computing resources, time and on various issues like: Network fault, Physical faults, Processor faults, Process faults, Service expiry fault.

In cloud, there are various fault tolerance techniques available some of these are [6]:

- **Check pointing:** A Checkpoint is an efficient task level fault tolerance technique for long running and big applications. In this technique after doing every change in system a check pointing is done. When a task fails, rather than from the beginning, it is allowed to be restarted that job from the recently checked pointed stare.
- **Self- Healing:** For better performance, a big task can divided into various parts. When several instances of an application are running on several virtual machines, it automatically handles failure of application instances.
- **Job Migration:** During job migration, failure of any task at any time, the task can be migrated to another machine.
- **Replication:** In replication, various tasks are replicated and they are running on different resources, for the successful execution and for getting the desired result.
- **Task Resubmission:** A job may fail now whenever a failed task is detected, In this case at runtime the task is resubmitted either to the same or to a different resource for execution.
- **Masking:** After the employment of error recovery the new state needs to be identified as a transformed state. Now if this process applied systematically even in the absence of effective error provide the user error masking.
- **Timing check:** Timing check is a supervision technique with time of critical function and this is done by watch dog.
- **S-Guard:** S-Guard is based on rollback recovery and it is less turbulent to normal stream processing.
- **Safety-bag checks:** In the safety - bag checks the commands which are not meeting the safety properties and block those commands.
- **Pre-emptive Migration:** Pre-emptive Migration is techniques in which the application is constantly monitored, analyzed and count on a feedback-loop control mechanism.
- **Retry:** Retry is the simplest technique that retries the failed task on the same resource and implement a task again and gain.
- **Rescue workflow:** Rescue workflow is a technique that allows the workflow to persist until it becomes unimaginable to move forward without catering the failed task.
- **Reconfiguration:** In reconfiguration technique the faulty component eliminated from the system.
- **Software Rejuvenation:** It is a technique that designs the system for periodic reboots and it restarts the system with clean state with a fresh start.

III. LOAD BALANCING

Load balancing is all about availability, scalability and performance of resources for critical web-based applications. Load balancing is a process of reassigning the total load to the individual nodes of the collective system of the facilitate networks and resources to improve the response time of the job with maximum throughput in the system [7]. The important things which said about load balancing are estimation of load, comparison of load, stability of different system, performance of system, interaction between the nodes, nature of work to be transferred, selecting of nodes and many other ones to consider while developing such algorithm [8]. Initially Load balancing techniques are classified as:

Static algorithms: Static algorithms are those which divide the traffic equivalently between servers.

Dynamic algorithm: Dynamic algorithms are which search for the lightest server in the network and then designated appropriate weights on it.

A. Existing Load Balancing Algorithm:

Some existing l*oad balancing algorithms* are discussed as follows:

Biased Random Sampling [9] approach the load on a server is represented as a virtual graph having connectivity with each node. Each server is symbolized as a node in the graph, with each in degree directed to the free resources of the server. Whenever a node executes a job, it deletes an incoming edge, which indicates a reduction in the availability of free resource. After completion of a job, the node adds on an incoming edge, indicating an increase in the availability of free resource. Random sampling is used for the increment and decrement processes. The last node in the walk is selected for allocation of load; instead, any other node based on certain criteria could also be preferred. A node on receiving a job, will execute it only if its current walk length is equal to or greater than the threshold value. Else, the walk length of the job under consideration is incremented and another neighbor node is selected randomly. Again a new directed graph is formed and load balancing is achieved in a fully decentralized manner, thus making it suitable for large network systems like a cloud.

Active Clustering [10] is considered as a self-aggregation algorithm, works on the principle of grouping the similar nodes and work together on these available groups. A set of processes is iteratively executed by each node on the network. Initially any node can become an initiator and selects another node from its neighbors to be the matchmaker node satisfying the criteria of being a different type than the former one. The matchmaker node then forms a connection between neighbors of it which are similar to the initiator. The matchmaker node, then removes the connection between itself and the initiator.

Honey Bee Foraging [11] algorithm is derived from the behaviour of honey bees for finding and reaping food. In order to check for fluctuation in demand of services, servers are grouped under virtual servers (VS), having its own virtual service queues. Each server processing a request from its queue calculates a profit or reward on basis of CPU utilization, which is corresponds to the quality that the bees show in their waggle dance and advertise on the advert board. Each of the servers takes the role of either a forager or a scout. A server serving a request, calculates its profit and compare it with the colony profit, if profit was high, then the server stays at the current virtual server and if it was low, then the server returns to the forager or scout behaviour, thus balancing the load with the server.

Join Idle Queue [12] load balancing algorithm is applicable for dynamically scalable web services. This technique involves a dispatcher to whom processors informs at the time of their idleness, without interfering with job arrivals. Thus removing the load balancing work from the critical path of request processing, system load is reduced; no communication overhead at job arrivals and no increment in actual response time.

Divisible Load/Task Theory (DLT) [13] is inspired by level-balancing property of liquid. In DLT, a load can be arbitrarily partitioned into chunks for a group of processors with no priority relationship among obtained chunks. This approach assumes that the nodes in the grid to be homogeneous and during load balancing no task can neither insert the queue nor leave. A node can only exchange and collect information with their nearest neighbours within one hop to make grids converge to the load balanced equilibrium state.

Load Balance Min-Min (LBMM) scheduling algorithm [14] and new optimized *Load Balancing Max-Min-Max (LB3M)* [15] had main objective to minimize execution time of each task, also avoid unnecessary replication of task on the node thereby minimizing overall completion time. Opportunistic Load balancing algorithm when combined with LBMM (OLB + LBMM) [14] keeps every node in working state to achieve load balance. Similar to LBMM, LB3M [15] also calculate average completion time for each task for all nodes. Then mark the task with maximum average completion time. After that it dispatches the task of marked node to the unassigned node with minimum completion task, thus balancing the workload evenly among all nodes.

IV. LOAD BALANCING SCHEMES INCLUDING FAULT TOLERANCE

In this section we analyse load balancing schemes including fault tolerance and some of them are as follows:

Ant Colony Optimization (ACO) [16] is an improved version of load balancing mechanism based on Ant Colony and Complex Network Theory (ACCLB) [17] in an open cloud computing federation. Both algorithms make use of ants' pheromone to gather and update information about the cloud thus selecting a specific node in order to assign the task however evenly distributing work among nodes. The ants in proposed algorithm continuously originate from the Head node and traverse all around the network making forward and backward movement to find the under loaded and overloaded nodes. In ACO [17] two types of pheromones are used *Foraging Pheromone* (FP) used to explore overloaded node by forward movement of ants while *Trailing Pheromone* (TP) used to discover its path back to the under loaded node. In order to limit the number of ants in the network, they would commit suicide once it finds the target node.

ESWLC (Exponential Smooth Forecast based on Weighted Least Connection) [18] improved form of Weighted Least-Connection (WLC) along with its features, it also take into account time series and trials. However WLC counts the connections of each server and reports the appropriate server based on the multiplication of a server weight and its count of connections, ESWLC algorithm concludes assigning a certain task to a node only after getting to know about the node capabilities. ESWLC builds the decision based on the experience of the node's CPU power, memory, number of connections and the amount of disk space currently being used. ESWLC then predicts which node is to be selected based on exponential smoothing [18].

Map Reduce [19] In *Map Reduce Fault Tolerance*, the master first attempts to assign a map task (in the queue) whose data is on that machine (*data locality*) provided that the machine is free to process the request. In case of failure in the execution, it attempts to assign a map task whose data is located (on a machine) on the same network switch with that machine (*rack locality*). Therefore, on occurrence of failure complete map tasks need to be re-executed, but completed reduce tasks does not. To ensure that a failed job can be recovered and is being scheduled with a guaranteed time period, a threshold value is used whereby beyond it, the failed job will be scheduled on the next available machine in spite of data locality [19].

Virtual Machine Mapping (VM Mapping) [20] is based on multi-dimensional resources to achieve overall load balance. This algorithm adopts the centralized control architecture comprises of scheduling controller and resource monitor as core elements of the system. The scheduling controller is responsible for VM lifecycle management and fulfilling allocation policy while the resource monitor collects the information about resources from physical hosts [20].The processes involved in VM mapping policy goes through following four phases: firstly; accepting the request for virtual machine on FCFS principle, secondly; obtaining resource information which in turn is maintain by resource monitor, thirdly; VM initial placement by scheduling controller, finally; user can remotely access the application on cloud.

Dual Direction FTP (DDFTP) [21] is a *dual-direction* downloading algorithm from FTP servers and its modified version is in [22], introducing efficient fault-tolerance and load balancing with minimal communication and coordination overhead while executing services in parallel over shared and dynamic heterogeneous distributed cloud infrastructure. The main idea of this algorithm is to splits the file into two half and task is being executed on two servers, such that each server starts processing the task in an opposite direction from the other, one server starts processing from the beginning in an incremental order while other starts the file downloading from the last block in decrement order. The task is considered to be finished when the two servers download two consecutive blocks meeting at consent. As a result, both servers will work independently, but will end up downloading the whole file to the client in the best possible time given the performance and properties of both servers [22]. Moreover, attributes such as network load, node load, network speed are automatically taken into consideration, while no run-time monitoring of the nodes is required, yet it maintain good load balancing among all participating server. In addition, if one of the servers fails before completion of task, second one continues the task till it reaches to the point where the other gets fail.

Fault Tolerance Policy on Dynamic Load Balancing (FTDLB) [23] is a fault tolerance policy that could tolerate the node's permanent failures while balancing load of real-time applications on P2P grids. For improving the system reliability, FTDLB duplicates jobs into different sites and adaptively adjust the load of real-time applications to achieve the job's minimal turnaround time. FTDLB algorithm works as follows; each site regularly sends "heartbeat" messages to its neighbour site which includes the CPU utilization, memory usage, job status, etc. When receiving of "heartbeat" messages stops, within a fixed period, it indicates failure of neighbour site thus triggering fault tolerance policy.

O-Ring (Overlapped Ring) [24] is a novel architecture that provides fault tolerance and load balancing for distributed and dynamic scenario. O-Ring use the approach of data replication (mirroring) and data distribution in order to provide both fault tolerance and load balancing in well- organized manner. In the initial phase data items are replicated on the neighbouring peers on the ring in order to achieve fault tolerance and each peer also stores the address of its predecessor and successor. Every ring had a Directory Service which is responsible for routing of requests like; data retrieval, updates, insertions and deletions on appropriate peers. As copy of data is already being replicated for backup on another peer short-term fluctuations are addressed by moving the boundaries of responsibility between peers without the need to move the data itself. Thus, redistributing the load in forward and backward direction in order to balance the load faster, and minimizing interferes with concurrent query processing. Any types of fluctuations, that require the movement of data, are addressed by moving the backup copies of the data in the background, without disturbing the primary copy of the data that is being used to handle requests for the data. Along with less expensive load balancing of O-Ring also achieves higher throughput, as it can balance the load with lower overheads and can respond rapidly to load imbalances.

Honey Bee Behaviour inspired Load Balancing [HBB-LB] [25] a technique which helps to achieve even load balancing across virtual machine to maximize throughput. It considers the priority of task waiting in queue for execution in virtual machines. After that work load on VM calculated decides whether the system is overloaded, under loaded or balanced. And based on this VMs are grouped. New according to load on VM the task is scheduled on VMs. Task which is removed earlier. To find the correct low loaded VM for current task, tasks which are removed earlier from over loaded VM are helpful. Forager bee is used as a Scout bee in the next steps.

V. CONCLUSION

In the cloud computing environment, fault tolerance with load balancing is one of the main issues that is required to improve the performance of it. This paper primarily focuses on fault tolerance and load balancing algorithms in a cloud environment.

For this, various existing fault tolerance techniques, load balancing algorithms and load balancing algorithm including fault tolerance are surveyed. From the survey, we have identified that there is a need to implement the autonomic fault tolerance technique for multiple instances of an application running on several virtual machines, a new approach needs to be developed that integrate fault tolerance techniques with load balancing algorithm or with existing workflow scheduling algorithms.

REFERENCES

[1] L. Wang, G. Laszewski, "Scientific cloud computing: Early definition and experience", in Proceedings of 10th IEEE International Conference on High Performance Computing and Communications (Dalian, China, 2008), pp. 825–830.
[2] Mell, Peter, and Tim Grance. "Draft NIST working definition of cloud computing."*Referenced on June. 3rd* 15 (2009).
[3] Rimal, Bhaskar Prasad, Eunmi Choi, and Ian Lumb. "A taxonomy and survey of cloud computing systems." *INC, IMS and IDC, 2009. NCM'09. Fifth International Joint Conference on*. IEEE, 2009.
[4] Geoffroy Vall´ee , Kulathep Charoenpornwattana, Christian Engelmann Anand Tikotekar , Chokchai Leangsuksun , Thomas Naughton , Stephen L. Scott ,"A Framework For Proactive Fault Tolerance", Availability, Reliability and Security, 2008. ARES 08. Third International Conference pp:659-664.
[5] Akanksha Chandola Anthwal, Nipur, "Survey of Fault Tolerance Policy for Load Balancing Scheme in Distributed Computing", International Journal of Computer Applications (0975 – 8887) Volume 74– No.15, July 2013.
[6] Saikia, Lakshmi Prasad, and Yumnam Langlen Devi. "FAULT TOLEREANE TECHNIQUES AND ALGORITHMS IN CLOUD COMPUTING." *International Journal of Computer Science & Communication Networks* 4.1 (2014):1
[7] R. Shimonski. "Windows 2000 & Windows Server 2003 Clustering and Load Balancing", Emeryville. McGraw-Hill Professional Publishing, CA, USA (2003), p 2, 2003.

[8] Ali M. Alakeel, “A Guide to Dynamic Load Balancing in Distributed Computer Systems”, IJCSNS International Journal of Computer Science and Network Security, VOL.10 No.6, June 2010.

[9] O. Abu- Rahmeh, P. Johnson and A. Taleb-Bendiab, “A Dynamic Biased Random Sampling Scheme for Scalable and Reliable Grid Networks”, INFOCOMP - Journal of Computer Science, ISSN 1807-4545, 2008, VOL.7, N.4, December, 2008, pp. 01-10.

[10] F. Saffre, R. Tateson, J. Halloy, M. Shackleton and J.L. Deneubourg, “Aggregation Dynamics in Overlay Networks and Their Implications for Self-Organized Distributed Applications.” The Computer Journal, March 31st, 2008.

[11] Randles, M., D. Lamb and A. Taleb-Bendiab, “A Comparative Study into Distributed Load Balancing Algorithms for Cloud Computing,” in Proc. IEEE 24th International Conference on Advanced Information Networking and Applications Workshops (WAINA), Perth, Australia, April 2010.

[12] Yi Lua, Qiaomin Xiea, Gabriel Kliotb, Alan Gellerb, James R. Larusb, Albert Greenbergc, “ Join-Idle-Queue: A Novel Load Balancing Algorithm for Dynamically Scalable Web Services” Volume 68 Issue 11, November, 2011, pp:1056-1071, Elsevier Science Publishers, 2011.

[13] V. M. B. Veeravalli, D. Ghose and T. Robertazzi, “Scheduling Divisible Loads in Parallel and Distributed Systems,” *IEEE CS Press*, 1996.

[14] S. Wang, K. Yan, W. Liao, and S. Wang, “Towards a Load Balancing in a Three-level Cloud Computing Network”, Proceedings of the 3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT), Chengdu, China, September 2010, pages 108-113.

[15] Che-Lun Hung, Hsiao-hsi Wang and Yu-Chen Hu “Efficient Load Balancing Algorithm for Cloud Computing Network”, International Conference on Information Science and Technology (IST 2012), April 28-30, pp; 251-253.

[16] Nishant, K. P. Sharma, V. Krishna, C. Gupta, KP. Singh, N. Nitin and R. Rastogi, "Load Balancing of Nodes in Cloud Using Ant Colony Optimization." In proc. 14th International Conference on Computer Modelling and Simulation (UKSim), IEEE, pp: 3-8, March 2012.

[17] Zhang, Z. and X. Zhang, "A load balancing mechanism based on Ant Colony and Complex Network Theory in Open Cloud Computing federation." In proc. 2nd International Conference on. Industrial Mechatronics and Automation (ICIMA), IEEE, Vol. 2, pp:240-243, May 2010.

[18] Ren, X., R. Lin and H. Zou, "A dynamic load balancing strategy for cloud computing platform based on exponential smoothing forecast" in proc. International Conference on. Cloud Computing and Intelligent Systems (CCIS), IEEE, pp: 220-224, September 2011.

[19] Qin Zheng, “Improving MapReduce Fault Tolerance in the Cloud”, Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), 2010 IEEE International Symposium, April 2010.

[20] Ni, J., Y. Huang, Z. Luan, J. Zhang and D. Qian, "Virtual machine mapping policy based on load balancing in private cloud environment," in proc. International Conference on Cloud and Service Computing (CSC), IEEE, pp: 292-295, December 2011.

[21] Anju Bala, Inderveer Chana, “Fault Tolerance- Challenges, Techniques and Implementation in Cloud Computing”, IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 1, No 1, January 2012.

[22] Jameela Al-Jaroodi, Nader Mohamed, and Klaithem Al Nuaimi, “An Efficient Fault-Tolerant Algorithm for Distributed Cloud Services,” in proc. 2012 IEEE Second Symposium on Network Cloud Computing and Applications, pp:1-8.

[23] Tian-Liang Huang, Tian-An Hsieh, Kuan-Chou Lai, Kuan-Ching Li, Ching-Hsien Hsu, and Hsi-Ya Chang, “Fault Tolerance Policy on Dynamic Load Balancing in P2P Grids”, in proc. International Joint Conference of IEEE TrustCom-11/IEEE ICESS-11/FCST-11, 2011, pp:1413-1419.

[24] P. M. Melliar-Smith and Louise E. Moser, “O-Ring: A Fault Tolerance and Load Balancing Architecture for Peer-to-Peer Systems”, Proceedings of International Conference of the Chilean Computer Science Society 2009, IEEE Computer Society, pp:25-33.

[25] Dhinesh B. L.D , P. V. Krishna, “Honey bee behavior inspired load balancing of tasks in cloud computing environments”, in proc. Applied Soft Computing, volume 13, Issue 5, May 2013.