Grenze International Journal of Engineering and Technology, Jan Issue

Research Summary of Load Balancing and Interoperability in Distributed IoT

Suvarna N Aradhya[1] and Dr. Rashmi Priya[2]
[1]JSS Academy of Technical Education, Department of ECE, Noida, Uttar Pradesh, India
Email: suvarna@jssaten.ac.in
[2]GD Goenka University, Department of MCA, Gurgaon, Haryana, India
Email: Rashmi.priya@gdgu.org

***Abstract*—With the evolutionary era of IoT and its implied changes in area of computing, Distributed systems have resurfaced as Cloud Computing, Grid Computing, Utility Computing, Cluster Computing, Edge Computing and Fog Computing. Load balancing is very vital in distributed systems for optimum usage of resources: computing power, storage, bandwidth, latency while maintaining the required levels of security and fault tolerance. The research on this topic is very extensive with varying algorithms, architectures, programming models, communication protocols. This research survey paper makes a comparison of the research works, research gaps and related future scope for improvement through bridging the gap.**

***Index Terms*— Load Balancing, Distributed Systems, Edge Computing, IoT, Algorithms, Task Scheduling, Resource Allocation, Fault Tolerance.**

I. Introduction

Load balancing refers to efficiently distributing the work among a group of nodes with minimum overheads of communication and maximum speed of completion. The paper is organized in sections as follows:

- Evolving Technologies around Distributed Architectures
- Load Balancing in distributed IoT systems and related algorithms
- Implementation of Load Balancers
- Research questions and answers
- IoT Edge
- Comparison of selected algorithms
- Conclusion and Future Scope

II. Evolving Technologies around Distributed Architectures

The various architectures supporting distributed computing to suit the challenges posed by IoT are mentioned in [3] as:

A. Edge Computing

After a decade of extensive progress in Cloud Computing, the focus is now on Edge/Fog Computing. The reason has been the increasing number of devices (IoT), data and consistent demand for more bandwidth and improvement in latency for time critical services. Edge Computing has the following characteristics:

Grenze ID: 01.GIJET.8.1.4

- Lower demands on network bandwidth and greatly reduced latency.
- Increased privacy of sensitive information.
- Enable operations even when networks are disrupted.

B. Fog Computing

It brings the computation and control closer to the sensors, actuators and users. Fog is one in which the computation is moved near to the edge of the network.

C. Mobile Cloud Computing

In order to provide a large number of high-end applications to even the mobile devices with minimum capabilities (computing, storage), the processing and storage is moved to the cloud.

D. Cloudlet Computing

It is an architecture that resembles the fog computing. It is the most common implementation of MCC. Basically, CC consists of using cloudlets to perform data processing and storage close to end devices. A cloudlet is a trusted, small cloud infrastructure, located at the edge of the network and available for nearby mobile devices, that collaborates with the cloud to compute the results and then sends them back to mobile devices.

E. Software Defined Network

The new buzzword in the field of networked systems is SDN(Software Defined Network). SDN architecture[21] decouples network control functions like routing decisions and network controlling operations from data forwarding functions such as encapsulation, decapsulation, management of data packets, etc. This enables the network control to become directly programmable, irrespective of the technology and manufacturer of the connecting devices. As a result, the underlying network infrastructure can be abstracted from applications and network services.

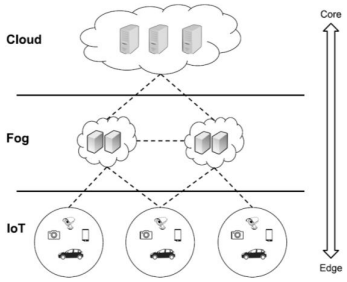

Figure 1. Three Layer Architecture of Fog Computing[2]

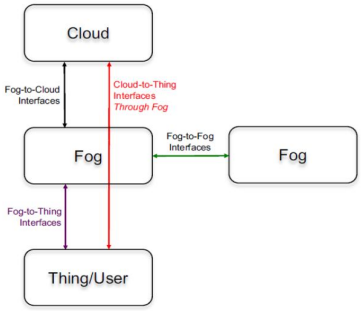

Figure 2. Fog Interfaces[2]

The architecture of FECIoT[4,5] has three layers(Sensors and Actuators, Network, Application) as in Fig. 1, and can be extended to five layers (Service layer and Business layer)

III. Load Balancing in Distributed IOT Systems and Related Algorithms

Load Balancing in reference[23] is a computer networking method to distribute workload across multiple computers or a computer cluster, network links, central processing units, disk drives, or other resources, to achieve optimal resource utilization, maximize throughput, minimize response time, and avoid overload. It is a mechanism that distributes the dynamic local workload evenly across all the nodes in the whole cloud to achieve a high user satisfaction and resource utilization, hence improving the overall performance and resource utility of the system. It also ensures that every computing resource is distributed efficiently and fairly, preventing bottlenecks and fail-over.

Load Balancing Algorithms

The task allocation and load sharing algorithms[39] are broadly classified as

- Static Algorithms
- Dynamic Algorithms
- Adaptive Algorithms

Static algorithms assume a known behaviour of the nodes and share the load based on graph theory models. In dynamic algorithms, the node behaviour is assumed unpredictable and hence periodically or upon arrival of new task, the status of load is assessed and reallocated.
Adaptive algorithms take care of the dynamic nature of the systems and tasks and changes the redistribution policies according to the existing situation. Thus, it exhibits more flexibility than dynamic algorithms.
Few more load balancing algorithms [43] are:

A. Least Connection Method

directs traffic to the server with the fewest active connections. Most useful when there are a large number of persistent connections in the traffic unevenly distributed between the servers.

B. Least Response Time Method

directs traffic to the server with the fewest active connections and the lowest average response time.

C. Round Robin Method

rotates servers by directing traffic to the first available server and then moves that server to the bottom of the queue. Most useful when servers are of equal specification and there are not many persistent connections.

D. IP Hash

the IP address of the client determines which server receives the request.
Algorithms classified on certain parameters[39] exist as:

- pre-emptive or non-pre-emptive.
- Sender-Initiated, Receiver-Initiated.
- Periodic polling or Event driven or Threshold level triggering.
- Homogeneous or Heterogeneous nodes
- Disk-based or Disk-less nodes
- Shared File System or Local File System.
- Computationally Intensive or Data Intensive

E. Resource-Aware Load Balancing Algorithm (RALBA)

[40] assigns jobs to machines utilizing two sub-schedulers: Fill-scheduler and spill-scheduler. First, the Fill-scheduler assigns jobs to machines according to their computing share (computing capabilities in terms of MIPs). The computing share is determined as a ratio of total requisite computing requirements and total computing power of all the available machines. On contrary, spill-scheduler assigns the remaining jobs (to be scheduled) to machines relying on the earliest completion time (in descending order of job sizes) .

F. Sufferage heuristic

allocates job to the machines in descending order of suffrage value of the jobs [40]. The sufferage value is the difference between MCT and the second MCT for a specific job produced by different machines in cloud. The job with larger sufferage value is assigned to a machine with MCT for that job in each scheduling decision.

G. Minimum Completion Time (MCT)

considers an arbitrary order of jobs and assigns machine to jobs having earliest completion time [40]. MCT results in the assignment of some jobs to machines that do not produce the minimum execution time.

H. Min-Min [40] heuristic

calculates the MCT of each un-allocated jobs in scheduling decision. Next, the job with overall MCT is allocated to the corresponding machine and removed from the list of the unallocated jobs. Min-Min differs from the MCT in such a way that MCT only considers one job at a time and the Min-Min considers all unallocated jobs in each scheduling decision.

I. Max-Min

[40] Similar to Min-Min heuristic, the also calculates the MCT of each un-allocated job during every scheduling decision. Afterwards, the job with the overall maximum MCT (calculated in the first step) is allocated to the corresponding machine and removed from the list of unallocated jobs favouring large sized jobs.

IV. IMPLEMENTATION OF LOAD BALANCERS[42]

L4 — Directs traffic based on data from network and transport layer protocols, such as IP address and TCP port.

L7 — Adds content switching to load balancing. This allows routing decisions based on attributes like HTTP header, uniform resource identifier, SSL session ID and HTML form data.

GSLB — Global Server Load Balancing extends L4 and L7 capabilities to servers in different geographic locations.

SDN **—** Load balancing using SDN (software-defined networking) separates the control plane from the data plane for application delivery. This allows the control of multiple load balancing. It also helps the network to function like the virtualized versions of compute and storage. With the centralized control, networking policies and parameters can be programmed directly for more responsive and efficient application services. This is how networks can become more agile.

UDP **—** A UDP load balancer utilizes User Datagram Protocol (UDP). UDP load balancing is often used for live broadcasts and online games when speed is important and there is little need for error correction. UDP has low latency because it does not provide time-consuming health checks.

TCP **—** A TCP load balancer uses transmission control protocol (TCP). TCP load balancing provides a reliable and error-checked stream of packets to IP addresses, which can otherwise easily be lost or corrupted.

*SLB***—** Server Load Balancing (SLB) provides network services and content delivery using a series of load balancing algorithms. It prioritizes responses to the specific requests from clients over the network. Server load balancing distributes client traffic to servers to ensure consistent, high-performance application delivery.

Virtual **—** Virtual load balancing aims to mimic software-driven infrastructure through virtualization. It runs the software of a physical load balancing appliance on a virtual machine. Virtual load balancers, however, do not avoid the architectural challenges of traditional hardware appliances which include limited scalability and automation, and lack of central management.

Elastic - Elastic Load Balancing scales traffic to an application as demand changes over time. It uses system health checks to learn the status of application pool members (application servers) and routes traffic appropriately to available servers, manages fail-over to high availability targets, or automatically spins-up additional capacity.

Geographic **–** Geographic load balancing redistributes application traffic across data centers in different locations for maximum efficiency and security. While local load balancing happens within a single data center, geographic load balancing uses multiple data centers in many locations.

Multi-site -Multi-site load balancing, also known as global server load balancing (GSLB), distributes traffic across servers located in multiple sites or locations around the world. The servers can be on-premises or hosted in a public or private cloud. Multi-site load balancing is important for quick disaster recovery and business continuity after a disaster in one location renders a server inoperable.

Load Balancer as a Service (LBaaS) **—** Load Balancer as a Service (LBaaS) uses advances in load balancing technology to meet the agility and application traffic demands of organizations implementing private cloud infrastructure. Using an as-a-service model, LBaaS creates a simple model for application teams to spin up load balancers.

Per App Load Balancing-A per-app approach to load balancing equips an application with a dedicated set of application services to scale, accelerate, and secure the application. Per app load balancing provides a high degree of application isolation, avoids over-provisioning of load balancers, and eliminates the constraints of supporting numerous applications on one load balancer.

Weighted Load Balancing-Weighted load balancing is the process of permitting users to set a respective weight for each origin server in a pool. It's important to consider weighted load balancing because of its ability to rebalance traffic when an origin becomes unhealthily crowded. Depending on their respective weights and the load balancing weight priority, traffic will be rebalanced to the remaining accessible origins.

V. Research Questions

Some of the research questions which prompted for answers while delving into load balancing are listed below along with answers that were found through literature study.

A. What factors affect the speed of execution of distributed task?

When job is distributed to number of heterogeneous systems, the time of completion of the job depends on the slowest processor. Also, some of the threads, which require reallocation due to non-availability of systems at runtime affect the time of completion.

B. Is Edge Computing alone able to reduce computing time?

Edge Computing reduces the network latency by doing the computation at the place of data source. But, there are certain type of tasks such as up-streaming or down-streaming from cloud, which cannot be benefitted by Edge Computing. Therefore, architecture to be selected is specific to the type of task.

C. What are the different types of tasks that require specific architecture and allocation strategies?

The different types of tasks are

- Content Delivery Networks (CDN)
- Up-streaming or Down-streaming data from client-to-cloud or cloud-to-client
- Large computation on small sets of data
- Small computation on large sets of data
- Tasks requiring specialized software or hardware.

Each of the above types of tasks requires different setups for optimizing the speed and cost.

D. What type of communication is required between processing nodes?

- Inter-Process Communication or Intra-Process Communication on a single machine.
- Inter-Process Communication between systems on a same LAN.
- Inter-Process Communication between systems on different LANs.
- Remote method invocation
- Message transfer through MQTT protocol

E. What are the different types of Load Balancers?

- HTTPs Load Balancer
- SSL Proxy Load Balancer
- TCP Proxy Load Balancer
- Network Load Balancer
- Internal Load Balancer

F. What are the ways to implement Load Balancers?

- **Hardware Load Balancer:** A hardware load balancer, as the name implies, relies on physical, on-premises hardware to distribute application and network traffic. These devices can handle a large volume of traffic but often carry a hefty price tag and are fairly limited in terms of flexibility.
- **Software Load Balancer:** A software load balancer comes in two forms—commercial or open-source—and must be installed prior to use. Like cloud-based balancers, these tend to be more affordable than hardware solutions.
- **Virtual Load Balancer:** A virtual load balancer differs from software load balancers because it deploys the software of a hardware load balancing device on a virtual machine.

G. What problems and solutions can be specified regarding the load balancing in the coming years?

This question seeks to clarify the role of load balancing in the SDN, and identify the challenges and the techniques applied to guarantee the QoS.

H. What are the challenges posed by Internet of Things?

The emerging IoT poses the following challenges that might not be addressed by the cloud architecture.

- Latency
- Network Bandwidth
- Resource Constrained Devices
- Cyber-Physical Systems

- Uninterrupted service with intermittent connectivity to cloud
- New Security Challenges.

I. What are the Programming Models for parallel and distributed computing?

There are four types of parallel programming models:

- Shared memory model
- Message passing model
- Threaded model
- Data parallel model

There are several distributed programming architectures:

- Client-Server
- Three-Tier
- N-Tier
- Peer-To-Peer

Load balancing is also specific to the type of computing model in addition to the above mentioned programming models and hardware frameworks used such as:

- Cloud Computing
- Grid Computing
- Edge Computing
- Cluster Computing

VI. IOT EDGE

Load Balancing in distributed systems play key role in improving performance. Research in this area has led to development of Specialized Algorithms. But**,** Algorithms cannot be based only on the basic attributes of a distributed system such as - Heterogeneity, Communication Protocol, Latency, etc. It also depends on higher level QoS parameters like:

- Resource Utilization
- Response Time
- Processing Time
- Scalability
- Throughput
- System Stability
- Power Consumption

Effect of Load Balancing is to be measured with parameters such as: Execution Time, Resource Allocation, Resource Utilization, Processing Time, Migration Time, Migration Cost, Overhead Communication Involved, Service Level Violations, Degree of Balance, Task Rejection Ratio, Fault Tolerance, Latency.

The most meaningful challenges created by the proliferation of IoT devices, particularly in enterprise and industrial applications, include concerns regarding interoperability, security issues, and data management challenges – all of which can be effectively addressed with Distributed Computing.

Edge computing follows a three tier architecture in general[10], comprised of end device, edge layer (fog, cloudlet, MEC, MDC), and cloud data center. Various types of applications and SLA parameters are considered in the research papers studied on IoT Edge.

In the research area of Load Balancing and Interoperability in Distributed IoT, after knowing about load balancing algorithms, distributed architectures and their computing models, IoT and its challenges, the research narrowed down towards “Distributed Computing at IoT Edge”.

IoT Edge is a fully managed service built on IoT Hub. One can deploy cloud workloads—artificial intelligence and third-party services or business logic—to run on Internet of Things (IoT) edge devices via standard containers. By moving certain workloads to the edge of the network, the devices spend less time communicating with the cloud, react more quickly to local changes and operate reliably even in extended offline periods.

Under IoT Edge and load balancing, challenges are: resource allocation, task scheduling algorithms, security, cognition using AI or ML and energy efficiency.

VII. Comparison of Selected Algorithms

The simplest load balancing algorithm[26] works in the following manner.
Every node will make comparison of available units with the received amount of units from the overloaded node. If the node doesn't have enough capacity, it will share information with its neighbours and increase the hop-count by 1.

- If the node has enough capacity, the information will be aggregated, based on available capacities and hop-count, and send it to the parent node.
- After that, the overloaded node has information about a list of nodes that have enough capacity and are ready to accept an extra load.
- Overloaded node migrate extra load to the node with smaller hop-count and node that has more than enough available capacity.

The algorithm only considers the capacity of the nodes jobs in queue for distribution of tasks. Firstly, there is maintenance of a table of node capacities, which is to be regularly revised after completion of every task. This might consume more time than the time required for the task. Secondly, the task distribution cannot be done without the associated required data. Thirdly, fault tolerance is not accounted for after building the table. Lastly, latency is not considered in the algorithm, which is a vital part in networked systems.
Reference [6], a dynamic load balancing algorithm is proposed for a hierarchical control involving controllers and switches[6]. It transforms the load balancing problem into a network latency problem, which avoids load imbalance by reducing the queuing delay and transmission delay. CPU utilization shown through experimental results is more balanced.
Firstly, the algorithm considered is only for electing the leader (switch) by the controllers, for an SDN based on the minimum distance from the controller. But the speed of communication is not just dependent on distance alone but relies also on the processing device. Secondly, the task allocation and reallocation along with related data distribution required is not discussed. Resources can be shared through a common controller only at a specific level in the hierarchy. Vertically, the resource allocation plays a major role in controlling latency. It is implied that resources are replicated on all controllers.
Reference [37] Portrays an Improved Throttled Algorithm. In the traditional throttled approach, only one task can be assigned to a VM and there is no selection criterion of VM. Any VM which is idle can be assigned the task. It does not consider the task size and VM capacity. But in the proposed approach, i.e. Improved Throttled approach, priority is assigned to each VM. Priority is calculated based on the capacity of VM and active allocated task count and size.

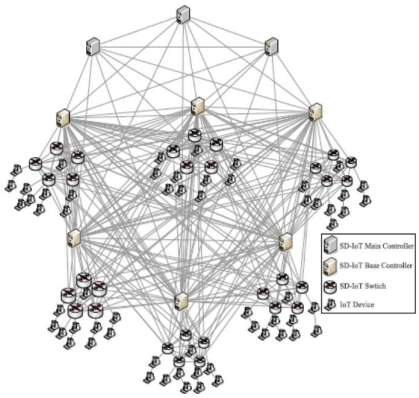

Figure 3. Vertical Control involving controllers and switches[6]

Although, it has yielded better performance in terms of the response time in comparison with round robin, equally spread or throttled, it does not involve the migration of the task or computing power optimization. The communication overhead to report the task completion seems to be higher.
Resource allocation[10] using CoAP (Constrained Application Protocol) that specifically targets resource constrained IoT devices and alleviates various overheads imposed by the HTTP protocol.
Architecture, Protocols, Simulators[4] for Fog Computing discussing technologies or protocols (LoRaWAN: Long Range Wide Area Network for power constrained devices, IPv6: Addressing scheme for for identifying large number of IoT devices on Internet), 6LowPAN (IPv6 over Low Power devices on Personal Area Network),

RPL (Ripple Routing Protocol), SOAP (Simple Object Access Protocol. SOAP is a protocol which is used to interchange data between applications which are built on different programming languages. SOAP is built upon the XML specification and works with the HTTP protocol. This makes it a perfect for usage within web applications.), MQTT (Message Queue Telemetry Transport Protocol for Latency constrained devices using publish/subscribe model for message transport), Simulators for implementing IoT Edge and their comparison.
Task allocation using three different control models[17] such as centralized, distributed and hybrid models is discussed. Practically, task allocation often means allocating tasks to resources. Therefore, along with task allocation, resource accessibility optimization is also considered. Two approaches to resource allocation: self-owned resource-based approach implemented based on nodes' self-owned resource statuses, i.e. a node's probability of being allocated tasks is determined only by its self-owned resources and a contextual
resource-based approach implemented based on not only nodes' self-owned resource statuses but also their contextual resource statuses because nodes may cooperate with others within their contexts.
Resource allocation and scheduling[36] is divided into three main categories.

- Dynamic Scheduling Algorithms.
- Agent Based Scheduling Algorithms.
- Cost Optimization Based Scheduling Algorithms.

Comparison of existing resource scheduling algorithms is being made on the basis of three parameters significant in resource scheduling: Resource utilization, cost of computing and load balancing. From this analysis it has been observed that there is no one algorithm satisfying all three basic parameters chosen for analysis. Thus improvement in resource scheduling algorithms is required. There is need for a single, scalable, adaptive and optimized technique for resource allocation which could satisfy diverse user demands.
A thorough comparison of two distributed system models - grid and cloud[32] in terms of the resource allocation problem is presented, as the two computing models often get confused due to their similar conceptual properties. Resource Allocation plays a crucial role in both models as it directly affects their performance in respect of resources utilisation, energy consumption, and/or load balancing. The main findings of this paper are:
Firstly, the resource allocation problem is completely different in both models in terms of
associated challenges, scheduling algorithms, and deploying strategies. In Cloud computing, the applications are hosted indirectly, i.e. first the applications are allocated to VMs and then the VMs get deployed into physical resources; this strategy enables simultaneous time and space sharing of the underlying resources. This feature allows latency sensitive applications to operate efficiently on Cloud. Whereas, Grid Computing resources are directly and independently deployed to the coming applications, but the deployment decision is also restricted by time and space requirements.
Secondly, the comparison shows that Cloud resources are rented under full virtualisation concept which leads to improve the overall resource utilization but it adds extra challenges in making the right allocation decision. It also indicates that both resources consolidation and load balancing are major challenges related to single Cloud resources allocation and not found in Grid resources allocation. On the other hand, Grid resources allocation suffers from the multi-site administration problem; wrong resources may be allocated with high energy consumption or low utilisation.
Thirdly, there is notable lack of performance analysis and fault tolerance research in Cloud resources allocation; ensuring that a good level of QoS is delivering to the end users is likely to be one of the major challenges in Cloud Computing since the Cloud is applicable to scalable user demands and dynamic providing scheme.
Fourthly, Clouds come in two types namely Single Cloud and Federated Cloud.
Research on resource allocation and scheduling in single Cloud environment is different from that in Federated Cloud. In the Federated Cloud the major resources allocation challenge is multi-site administration.

VIII. CONCLUSION

Load Balancing is an NP Hard Problem. There are a large number of parameters that are used to measure the effect of load balancing. These parameters can be grouped under attributes of Quality, Cost and Performance. There are several types of architectures: peer-to-peer, client-server, three-layer, cloud or grid. Programming models such as: Java RMI, CORBA, DCOM, Remoting. Communication protocols such as CoAP, REST, MQTT, TCPIP, HTTP, etc. Problem of load balancing can be specific to application, network or data. Hence, specific to the problem, only few quality of service parameters can be aimed for improvement using Load Balancing. But, associated with Load balancing is the problem of task migration, resource allocation and interoperability that can not be segregated. An important parameter for dynamic situations like that of IoT is: Fault Tolerance. This feature is not included in almost all the simulators being used by researchers. Dynamically

TABLE I. SUMMARY OF THE RESEARCH

Reference	Architecture	Algorithm	Implementation	Parameters	Simulator	Research Gaps
6	Fog Computing	Server with Minimum Ditance	SDIoT	Network Latency and Queuing Delay	OpenDayLight/ MiniNet	Processor Speed is not considered. Resource Allocation along with Task Redistribution is not discussed. Fault Tolerance not considered.
28	Edge Computing	Server with Minimum Ditance	Content Delivery Network (CDN)	Transmission Delay and Queuing Delay	Not Experimental Research	Algorithm is not Dynamic. Fault Tolerance not considered. Information maintenance overhead regarding the node capacities and loads is large. Resource Allocation along with Task Redistribution is not discussed.
17	Distributed & Centralized	MultiAgent based Algorithm	General Distributed System	Task Allocation Resource Accessibility	Not Experimental Research	Interoperability between nodes is not considered while making the task migration. Dynamic nature of the network is not thought of to include fault tolerance.
37	Cloud Computing	Improved Throttled Algorithm	Client-Server	Response Time, Cost, Throughput	CloudAnalyst	Communication overhead involved is high. Fault Tolerance is not measured as the simulator does not support it. Task Migration is not discussed
42	Cloud Computing	Round Robin with Server Affinity	Cloud Server	Response Time and Resource Utilization	CloudAnalyst	Overhead involved in maintaining a database of earlier VM allocations. Current active load is not considered and hence not dynamic.
26	WSN	Criticality Aware Load Balancing	IoT	Transmission Delay and Energy Conservation	OMNET++	Fault Tolerance not included in part. Energy constraint is taken care of. Non-communicating nodes and related task migration is not considered.

a node may become dead or alive. Both these conditions need to be tracked while distributing load. But the tricky nature of this parameter is that, if a node is suddenly dead, it is not able to even communicate its status as dead. The only way to trace this condition is through polling from a central scheduler.

RESEARCH GAP AND FUTURE SCOPE

Time of completion of a distributed task mainly depends on the slowest processor or the non-executing node due to which the task is to be reallocated to other node along with required resources. Identifying the non-performing node dynamically plays a vital role in time critical systems, which is named fault tolerance. None of the algorithms investigated in the research study has addressed fault tolerance. Also, every simulator being used to implement IoT or cloud lacks the feature of fault tolerance embedded in it.

Time critical applications like inter-vehicular communications, health care IoT devices for monitoring require latency to be within few milliseconds. Such applications would be greatly benefitted by incorporating fault tolerance in them.

Easily said than done is the fact that the central scheduler be informed about the failure of a processing node. It is the node which even fails to inform about its failed status. The only way left to find the status of failure is by periodically polling from the central scheduler. This is a communication overhead, that needs to be kept to minimum. So, lies the challenge to implement fault tolerance in load balancing algorithms.

REFERENCES

[1] Mung Chiang, *Fellow, IEEE,* and Tao Zhang, *Fellow, IEEE,* "Fog and IoT: An Overview of Research Opportunities", December 2016, IEEE Internet of Things Journal.

[2] Michele De Donno, Koen Tange and Nicola Dragoni, "Foundations and Evolution of Modern Computing Paradigms: Cloud, IoT, Edge, and Fog", October 2019, IEEE Access Open Journal.

[3] Ashkan Yousefpour, Caleb Fung, Tam Nguyen, Krishna Kadiyala, Fatemeh Jalali, Amirreza Niakanlahiji, Jian Kong, Jason P. Jue, " Fog and Edge Computing Paradigms: A complete survey", 2019, Journal of Systems Architecture, ScienceDirect.

[4] Babatunji Omoniwa, Riaz Hussain, Muhammad Awais Javed, Safdar Hussain Bouk , *Senior Member, IEEE,* and Shahzad A. Malik, "Fog/Edge Computing-Based IoT (FECIoT):Architecture, Applications, and Research Issues", June 2019, IEEE Internet of Things Journal

[5] Carla Mouradian, Member, IEEE, Diala Naboulsi, Sami Yangui, Roch H. Glitho, Member, IEEE, Monique J. Morrow, Senior Member, IEEE, and Paul A. Polakos, Senior Member, IEEE, "A Comprehensive Survey on Fog Computing: State-of-the-Art and Research Challenges", IEEE Communications Surveys and Tutorials, 2018

[6] Xiaoxun zhong, lianming zhang , and Yehua Wei, "Dynamic Load-Balancing Vertical Control for a Large-Scale Software-Defined Internet of Things.", September 23, 2019, IEEE Access Journal, DOI:10.1109/ACCESS.2019.2943173.

[7] Majid Ashouri, Fabian Lorig , Paul Davidsson and Romina Spalazzese, "Edge Computing Simulators for IoT System Design: An Analysis of Qualities and Metrics", November 2019, Future Internet-MDPI

[8] Hesham el-sayed, Sharmi Sankar, Mukesh Prasad, Deepak Puthal, Akshansh Gupta, Manoranjan Mohanty and Chin-Teng Lin, (fellow, IEEE), "Edge of Things: The Big Picture on the Integration of Edge, IoT and the Cloud in a Distributed Computing Environment", 6 December 2017,IEEE Access

[9] Mahadev Satyanarayanan, Weisong Shi, "Overview of Edge Computing", https://ea.iln.ieee.org/Kview/CustomCodeBehind/base/courseware/scorm/scorm12courseframe.aspx

[10] Kashif Bilal, Osman Khalid, Aiman Erbad, and Samee U. Khan Qatar University, Doha, Qatar, "Potentials, Trends, and Prospects in Edge Technologies: Fog, Cloudlet, Mobile Edge, and Micro Data Centers", October 2017, Computer Networks

[11] Rafael Moreno-Vozmediano, Eduardo Huedo, Ruben S. Montero, Ignacio M. Llorente, "A Disaggregated Cloud Architecture for Edge Computing", June 2019, IEEE Internet Computing.

[12] Maxim Chernyshev, Zubair Baig, Oladayo Bello, and Sherali Zeadally, "Internet of Things (IoT): Research, Simulators and Testbeds", June 2018, IEEE Internet of Things Journal.

[13] Wei Yu, Fan Liang, Xiaofei He, William Grant Hatcher, Chao Lu, Jie Lin and Xinyu Yang2, " A Survey on the Edge Computing for the Internet of Things", November 2017, IEEE Access

[14] Michele De Donno, Koen Tange, and Nicola Dragoni, "Foundations and Evolution of Modern Computing Paradigms: Cloud, IoT, Edge, and Fog" , October 2019, IEEE Access

[15] N. A. Suvarna , Dinesh Chandra "Evaluation of Improvement Algorithms for Dynamic Co-allocation with respect to Parallel Downloading in Grid Computing", August 2010, International Conference on Integrated Intelligent Computing.

[16] Xuezhi Zeng , Saurabh Kumar Garg, Peter Strazdins, Prem Prakash Jayaraman, Dimitrios Georgakopoulos, Rajiv Ranjan, " IOTSim: A simulator for analysing IoT applications", July 2016, Journal of Systems Architecture-Elsevier

[17] Yichuan Jiang, Senior Member, IEEE, " A Survey of Task Allocation and Load Balancing in Distributed Systems", February 2016, IEEE Transactions on parallel and distributed systems

[18] IBM Redbook on Grid Computing

[19] "Distributed Algorithms" , Nancy A Lynch, Google books

[20] Shahbaz Afzal, G Kavitha, "Load balancing in cloud computing – A hierarchical taxonomical classification", December 2019, Springer Journal of Cloud Computing, https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-019-0146-7

[21] Ali Akbar Neghabi, Nima Jafari Navimipour, Mehdi Hosseinzadeh, Ali Rezaee1, "Load Balancing Mechanisms in the Software Defined Networks: A Systematic and Comprehensive Review of the Literature", March 5, 2018, IEEE Access

[22] Pooja Kathalkar, A. V. Deorankar, "Challenges and Issues in Load Balancing in Cloud Computing", April 2018, International Journal for Research in Applied Science & Engineering Technology (IJRASET).

[23] Amal zaouch, Faouzia benabbou, "Load Balancing for Improved QoS in Cloud Computing", 2015, (IJACSA) International Journal of Advanced Computer Science and Applications.

[24] Veeramanikandan M. , Suresh Sankaranarayanan, "Publish/subscribe based multi-tier edge computational model in Internet of Things for latency reduction", Journal of Parallel and Distributed Computing 127 (2019), Pages:18-27.

[25] Mosab Hamdan, Entisar Hassan , Ahmed Abdelaziz, Abdallah Elhigazi, Bushra Mohammed, Suleman Khan, Athanasios V. Vasilakos, M.N. Marsono, " A comprehensive survey of load balancing techniques in software-defined network", Journal of Network and Computer Applications 174(2021) 102856.

[26] Rahim Khan, "An efficient load balancing and performance optimization scheme for constraint oriented networks", Simulation Modelling Practice and Theory 96 (2019) 101930

[27] Diana Josephine.D, Rajeswari.A, " Simulation and Performance Analysis of CIT College Campus Network for Realistic Traffic Scenarios using NETSIM", Procedia Computer Science 171 (2020) 2635–2644.

[28] Marian Kyryk, Nazar Pleskanka, Mariana Pleskanka, Petro Nykonchuk, " Load Balancing Method in Edge Computing", 15th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering, 2020.
[29] Altaf Hussain, Muhammad Aleem, Muhammad Azhar Iqbal, Muhammad Arshad Islam, "SLA‐RALBA: cost‐efficient and resource‐aware load balancing algorithm for cloud computing", The Journal of Supercomputing (2019) 75:6777–6803.
[30] Altaf Hussain, Muhammad Aleem, Muhammad Azhar Iqbal, Muhammad Arshad Islam, "A Rigorous Evaluation of State-of-the-Art Scheduling Algorithms for Cloud Computing", IEEE Access, DOI: 10.1109/ACCESS.2018.2884480.
[31] Jagdish Chandra Patni, Dr. M.S.Aswal, Om Prakash Pal, Ashish Gupta, "Load balancing Strategies for Grid Computing", 978-1-4244-8679-3/11/$26.00 ©2011 IEEE.
[32] Huda Hallawi, Jörn Mehnen and Hongmei He, "A comparison of resource allocation process in grid and cloud technologies", The Sixth Scientific Conference "Renewable Energy and its Applications" , IOP Publishing, 10.1088/1742-6596/1032/1/012048, 2018.
[33] https://www.researchgate.net/publication/ 331772303
[34] James Larus, "Programming Clouds", Microsoft Research white paper, 2010.
[35] Z. Tang, J. D. Birdwell, J. Chiasson, C. T. Abdallah, and M. Hayat, "Resource-Constrained Load Balancing Controller for a Parallel Database", IEEE transactions on control systems technology, vol. 16, no. 4, July 2008.
[36] Aarti Singh1, Manisha Malhotra, "A Comparative Analysis of Resource Scheduling Algorithms in Cloud Computing", American Journal of Computer Scie[nce and Engineering Survey, a Pubicon Open Journal,2013.
[37] Er. Imtiyaz Ahmad, Er. Shakeel Ahmad, Er. Sourav Mirdha, "An Enhanced Throttled Load Balancing Approach for Cloud Environment", International Research Journal of Engineering and Technology (IRJET) Volume: 04 Issue: 06 , June -2017, e-ISSN: 2395 -0056
[38] Shu-Ching Wang, Kuo-Qin Yan (Corresponding author), Wen-Pin Liao and Shun-Sheng Wang, "Towards a Load Balancing in a Three-level Cloud Computing Network",978-1-4244-5540-9/10/$26.00 ©2010 IEEE
[39] Kouider Ben, Mohammed-Mahieddine, "An Evaluation of Load Balancing Algorithms for Distributed Systems", Ph.D Thesis, University of Leeds, School of Computer Studies, October 1991.
[40] Altaf Hussain, Muhammad Aleem, Muhammad Arshad Islam and Muhammad Azhar Iqbal, "A Rigorous Evaluation of State-of-the-Art Scheduling Algorithms for Cloud Computing", IEEE Access, December 2018.
[41] Yichuan Jiang, Senior Member, IEEE, "A Survey of Task Allocation and Load Balancing in Distributed Systems", IEEE transactions on parallel and distributed systems, vol. 27, no. 2, February 2016.
[42] Komal Mahajan, Ansuyia Makroo and Deepak Dahiya, "Round Robin with Server Affinity: A VM Load Balancing Algorithm for Cloud Based Infrastructure", Journal of Information Processing Systems, Sept 2013.