Tec Empresarial
P-ISSN: 1659-2395; E-ISSN: 1659-3359

A SURVEY ON BASIC LOAD BALANCING TECHNIQUES

Dr.Ganesh Vilas Patil

Associate Professor, DYPCET, Kolhapur

Abstract: A distributed system offers a practical method for sharing resources. Using a distributed system will produce different effects depending on the load balancing technique employed. Load balancing algorithm provides decision making with efficient resource utilization and maximum throughput. The decision-making process is absent in work distribution or load sharing. Response time is used as the key attribute to measure the effectiveness of load balancing. There are various categories in which load balancing can be placed. This division is based on a number of characteristics. The aim of presented research work is to emphasize the fundamental classification of load balancing strategies.

Keywords: Distributed System, Static Load Balancing**,** Dynamic Load Balancing.

1 Introduction

In 1970, advancement in computer hardware and networking technologies resulted in the invention of distributed system.

Definition 1. (Distributed System) "Distributed system is defined as a set of autonomous computing nodes interconnected by a computer network and which facilitates users by making provision of resource sharing in a controlled way".

Distributed system works as a loosely coupled system. Loosely coupled systems consist of the computational nodes with its own memory and are interconnected by communication network. The nodes are interacting with each other with the help of message passing mechanism. Each node works on the private memory associated with that node so no overhead of memory consistency is associated. Till today, the distributed system has been changing its face to fulfil the dynamic needs of the user.

Now a day, because of easy and rapid availability of an internet facility, world is experiencing very close virtual connectivity. The statistical information provided by worldwide survey of wearesocial blog in January, 2022[1], we can represent the increase in the number of Mobile, Internet and Social Media users is shown in Figure 1.

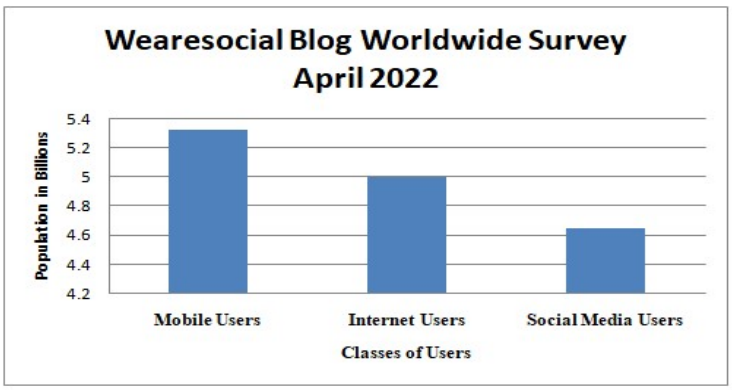

Figure 1. Wearesocial Blog Survey 2022

According to worldwide survey of wearesocial blog in 2018 [1], number of internet users in 2018 was 4.021 billion and increases by 7% every year. The exponential growth is observed in users of social media sites. The number of social media users in 2018 is 3.0196 billion and increases by 13% every year. Easily available internet facility on smart mobile devices results in increasing trend of multimedia based communication. In year 2018, the number of mobile phone users was 5.135 billion and per year it increases by 4%. Proliferation in number of internet users, results in demand of the internet based services. In order to respond to these extremely growing requests rate of multimedia communications, social site communications and many others, service providers need a powerful hardware and software infrastructure. This fact is proved by the Intel's survey of computer hardware demand. According to Intel's worldwide survey [2] the computational demand is growing 30% to 40% every year. From year 2009 to 2016, considering the global trends of computation, Intel data centre has increased the number of servers per pack from 140 to 280. This verity represents that, in seven years the need of virtualization has doubled by seven folds. The worldwide survey conducted by IDC (Interactive Data Corporation) [3] puts a light on a fact that from year 2009 to 2019, rate of x86 server's distribution is increased by 100.35%. The pragmatic surveys provided by Intel and IDC indicate the increasing rate of computational demands. The demand of computational power cannot be satisfied only with powerful hardware architectures. It also requires the applications which run on those hardware architectures and which does the efficient management of resources. The concept of grid emerged as a modified version of distributed system and provides the collaborative computational environment for execution of computational work load. It has made possible to run distributed applications in virtual environment to fulfil the users computational demands using concept of virtualization.

Cloud emerged as a virtual platform provider to provide solution to users computational problems. It is highlighted by Intel's survey that, from year 2009 to 2016 the number of virtual servers changed from 129 to 943. In terms of Grid Technology, Cloud Technology, Block Chain Management [4], Edge computing [5], distributed system continuing its journey of providing fruitful outcomes in field of computation. Distributed system provides a platform to solve the problems which includes massive computational executions. The computational intensive task is a set of instructions which includes more percentage of CPU intensive instructions than memory intensive or I/O intensive.

To fulfil the basic purpose of resource sharing, distributed system makes the provision of task execution on different nodes in a distributed manner. In distributed system, any computational problem is considered as a load. In order to execute the input load in distributed manner, it is divided into small fragments. This process of task division is called as fragmentation. The size of the fragment depends on the application characteristics and the policy adopted by the distributed system. The fragmented load is then mapped over the available set of computing nodes for execution. The policy of mapping the load to the computing nodes is called as scheduling. The basic objective of scheduling policy is to manage the resources in order to optimize resource usage, response time, network congestion and scheduling overhead.

A. Classification of scheduling paradigms

Scheduling paradigms of distributed system are fundamentally classified in three classes [6]:

1. Task assignment
2. Load sharing
3. Load Balancing

1. Task assignment

Task assignment approach consists of mapping of available tasks to the resources so as to improve performance of the system. Practical limitation of task assignment approach is that it assumes that following attributes are known in advance [6]:

1. Task characteristics
2. Computations required by each task
3. Speed of each processor
4. Cost of computation
5. Inter-process communication cost
6. Resource requirements of tasks
7. Task dependency

As it considers all mentioned attributes at static time, it does not consider the dynamically changing behavior of the system.

2. Load sharing

Load sharing approach confirms that no node in system remain idle by simple task to node mapping. Load sharing does not consider the state information of the system. Without considering the state information of resources it becomes difficult to manage the run time scheduling approach.

3. Load Balancing

In distributed system to achieve better resource utilization and to maximize the throughput; it needs to balance the workload among the available resources in such a way that each resource will have even contribution in task execution. Load from heavily loaded nodes need to be transferred to lightly loaded nodes. The performance of load balancing algorithm is measured in terms of response time achieved [6]. The task of load balancing in distributed system is performed by a special node known as load balancer.

B. Taxonomy of Load Balancing Algorithm

Figure 2 provides the classification of load balancing algorithms.

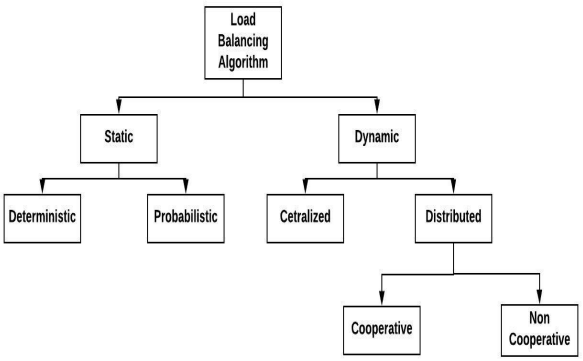

Figure 2. Classification of Load Balancing Algorithm

a. Comparison of Static and Dynamic Load Balancing Algorithms

Static load balancing [6-14] algorithms are simple and don't need to have state information of overall system for decision making. It maintains the system in steady state when the information of task characteristics and network attributes are known prior to decision making process. It does not consider the time variant load changes in a system. Whereas in dynamic load balancing algorithms [15-48] decision making is done using the current state of the system. Dynamic load balancing results in good performance benefits than static load balancing. The complexity of dynamic load balancing algorithms is more than static load balancing algorithms.

b. Comparison of Deterministic and Probabilistic Load Balancing Algorithms

Deterministic algorithms work in deterministic manner and utilize the characteristics of tasks for decision making. Probabilistic load balancing algorithms work on the static attributes of the system such as number of nodes, processing capacity of nodes and network topology. These both methods do not consider the load variation in the system with respective to time.

c. Comparison of Centralized and Distributed Load Balancing Algorithms

In centralized load balancing [17-33] approach only one node is responsible for taking scheduling decisions. State information of the system is collected at a centralized node which coordinates the decision making process. The coordinator node makes the decisions efficiently based on the available state information. Coordinator node collects the state information from all nodes involved in computation periodically. The performance of centralized load balancing system depends on the centralized decision maker.

In distributed load balancing [34-37] system more than one nodes are responsible for making load balancing decisions. It avoids the central dependency problem which exists in centralized dynamic load balancing by having more than one decision makers. The complexity of distributed load balancing is more than centralized load balancing. The communication cost incurred is also more as it involves internodes communication. Every node in a system collects the state information from all other nodes and does a decision making.

d. Comparison of Cooperative and Non cooperative Load Balancing Algorithms

In cooperative load balancing algorithms [34, 35] the nodes take the decision of load balancing with cooperation of all other nodes. This method of load balancing is more complex and involves larger communication overhead. In non-cooperative load balancing [35] each node has the autonomy to take the scheduling decisions independently.

C. General Issues in designing a load balancing Algorithms

1. Load estimation policy

It considers the problem of estimating the work load of a particular node in the system.

2. Process transfer policy

It is related with the choice of executing the process either locally or at a remote node. Solution to this issue depends on whether the load balancing approach is a static or dynamic.

3. State Information exchange policy

It decides method of exchanging the state information among the nodes involved in computation.

4. Location Policy

It takes the decision of transferring the selected task for execution.

5. Priority assignment policy

It is related with the assignment of execution priorities to local or remote tasks at a particular node.

6. Migration limiting policy

It determines the number of times a task migration is allowed in a particular load balancing system.

I. Motivation behind the work

The basic objective of presented research work is to provide the direction to the researcher who is working in the domain of dynamic load balancing. The research work includes fundamental classification of load balancing. The details of dynamic load balancing are provided in more precise way.

II. Literature Review

In modern era of computations, the demand for shared resource utilization is increasing enormously [1-5]. Modern applications demand, sharing of computational infrastructure to achieve minimization of response time and to increase throughput. Distributed systems attempts to solve this problem by providing a set of autonomous computational resources interconnected by communication network. The technique used to avoid the unbalanced utilization of computational nodes, is called as load balancing.

Decision making system in distributed system does the mapping of the available resources to the incoming load with best possible alternatives. Based on the result of decision making, node is selected for execution. In work flow of load balancing, variation in time of decision making results in two classes of load balancing.

1. Static Load Balancing
2. Dynamic Load Balancing

If the decision making is done at static time, the corresponding load balancing is called as static load balancing. If the decision of load distribution is taken at run time the corresponding load balancing is called as dynamic load balancing.

Performance of load balancing depends on a decision making policy adopted by distributed system. Based on the policy of decision making load balancing strategies are classified as [6]

1. *Global strategy*
2. *Cooperative strategy*
3. *Non-cooperative strategy*

In *global strategy*, only one decision maker exists that takes decision of resource allocation and job mapping over a distributed set of resources. In global strategy information is collected from all the nodes which are involved in computation. In *cooperative approach*, several decision makers exist; those take decisions at local level and cooperate with each other using message passing system. In cooperative approach of load balancing state information is collected from neighbouring nodes. In *non-cooperative approach*, each node is responsible for taking its own load balancing decision independently. Local level decision making causes the minimization of communication cost.

The performance of distributed system depends on the process of decision making. Accuracy of decision making can be enhanced by using appropriate load balancing algorithm. Table 1 represents the parameters which are considered as a performance indicators of load balancing algorithm [7],

Table 1 Performance indicators of load balancing algorithm

Sr.No.	Performance Indicator	Significance
1	Resource Utilization	A good load balancing algorithm is expected to give optimum resource utilization
2	Performance	Performance of load balancing algorithm is checked as an efficiency of resource utilization.
3	Scalability	It is necessary attribute to be a good load balancing algorithm to provide flexibility in increasing and decreasing the numbers of computational nodes during the computation.
4	Throughput	This attribute measures the number of tasks completed per unit time. For a good load balancing algorithm value of this attribute should be high enough
5	Response time	It is the difference in time between the job submission and the final output. A good load balancing strategy is always trying to minimize the response time of computation.
6	Associated Overhead	This attribute involves job transfer cost and inter process communication cost. It is the measure of overall communication cost incurred in computation.

1. Static Load Balancing

In static load balancing technique [6-14] the decision of task to resource mapping is taken at static time. The decision of load balancing is governed by the static information of job behavior and fixed characteristics of underlying resources. Static load-balancing algorithms may be either *deterministic* or *probabilistic* [6]. Deterministic algorithm utilizes the information of properties of nodes and characteristics of given process for load distribution. The probabilistic algorithm utilizes the static properties of the system like number of nodes and processing capacity of each node. Static load balancing exhibits fixed functional behavior under fixed conditions and inputs. It is easy to predict the behavior of the system under certain conditions with fixed inputs. Based on job properties, resource capacity and communication cost incurred, the scheduler takes the decision of job distribution at static time. Static load balancing is simple and effective when the workload can be sufficiently well characterized beforehand, but it fails to adjust to the fluctuations in system load [7].

Definition 2. (Static Load Balancing) *"Load balancing policies that react to the system before job execution with known job characteristics, resource capacities and fixed arrival pattern are termed as static policies"*.

It becomes easy for policy maker to design the static policy as it excludes the dynamic state information and run time communication overhead. The performance of static load balancing depends on the scheduling policy used for load balancing. Static load balancing can be carry out by using different scheduling algorithms.

Based on the scheduling algorithms used, static load balancing can be classified in following classes [8]:

1.1 Round Robin Scheduling

In this algorithm job assignment is done in round robin fashion. The total time quantum is divided into number of fixed length time quantum and each quantum is assigned to each processor in round robin fashion. To improve the performance of round robin algorithm designer should consider the job size. If the job size is large the quantum size should be large enough so as to achieve maximum throughput. The small length of time quantum results in starvation of jobs which requires large completion time. If the job execution is not completed in given time quantum, the job has to wait for long period. The job waiting time depends on number of jobs in a job buffer. This type of scheduling is useful for the jobs which require less amount of execution time.

1.2 Min-Min policy Scheduling

This algorithm calculates the minimum completion time for each node and then assigns a task to a node with minimum completion time. It repeats the same procedure until no job remains in a waiting queue. This strategy increases the starvation of tasks which requires high completion time because as per the strategy defined every time a task with minimum completion time is selected for execution.

1.3 Max-Min Scheduling

In order to achieve maximum throughput and minimum response time this algorithm selects the task with maximum completion time and assigns it for execution. In contrast with the Min-Min algorithm it increases the waiting time of tasks which require a minimum completion time.

1.4 Opportunistic scheduling

This algorithm assigns the incoming task to a random node for execution. Task assignment in this algorithm is done without calculating the load on each node.

In [9] authors presented a global load balancing algorithm known as optimal load balancing. The basic objective of this work is to minimize the response time of task execution. It is bidding based algorithm in which each node in a distributed system is considered as an agent. Each node in a distributed system has a true value 't_i' which is equals to the inverse of computational capacity of that node. Central coordinator collects these true values from all nodes and bids on those values so that task allocation should be optimum. All nodes get the appropriate fraction of the incoming load based on the true values. The system returns the profit value to each node and validates weather that profit value is nonnegative. If the profit value is negative, then system recognizes that the true value given by the node was incorrect. This work increases the voluntary participation of nodes where each node is racing to get maximum profit value and gives maximum optimal solution to task execution. This system is centralized coordinated so it creates a bottleneck in load failure situations. This work assumes that total job arrival rate must be equal to aggregate job service rate but this assumption cannot fulfill the real time dynamic job processing requirements.

In [10] author recommends a static job scheduling policy which ensures to have very low overhead and is less complex as compared with dynamic load balancing strategies. Static load balancing typically contains two components: a workload allocation scheme and a job dispatching strategy. Workload allocation scheme calculates the relative fraction of total workload to be assigned to each node. Job dispatching policy assigns calculated fraction to each computational node. The jobs are dispatched using round robin mechanism. The attainment of load balancing is checked by using a mathematical analysis of system performance and optimized allocation of workloads. The performance metrics used in this experiment are: *Mean response time* which is the average completion time of all the jobs and *Mean response ratio* which is the ratio of the job's response time to its size. Job size is defined as the completion time of the job when it is executed on an idle machine with relative speed one. The one another attribute used for performance measurement is fairness. Fairness value is calculated by taking standard deviation of response ratio of all the jobs. This experimental work considers a load balancing as a non linear optimization problem.

It considers four job dispatching strategies:

1. Optimized Round Robin (ORR)
2. Weighted Round Robin (WRR)
3. Optimized Random (ORAN)
4. Weighted Random (WRAN).

Comparison of all these job dispatching strategies is provided at 70% overall system utilization. The research work concludes with the fact that ORR and ORAN achieves better fairness than WRR and WRAN. It shows factual analysis of response time which found more predictable under an optimized allocation strategy. It considers run queue length of CPU as only one parameter for scheduling, so this method is applicable for CPU intensive tasks only.

In [11] author has classified static load balancing into two categories namely Optimal and Sub-Optimal. In optimal static load balancing technique optimization criterion function is used to optimize throughput, resource utilization and response time. In Sub-Optimal technique more prominence is given to the scheduling policy. This method is carried out in homogeneous execution environment. This method does not provide load balancing in heterogeneous work environment so it limits its performance only for homogeneous environment.

In [12] author has proposed static load balancing based optimal load balancing strategy to minimize the mean job response time. Response time at each node is considered as an aggregation of queuing delay, processing time and communication delay. A mean job response time is defined as a function of load at a particular node. In this system job is either processed at the host node or it is transferred to another node. The decision of job transfer is not depending on the current system state so it is static in nature. The decision of job execution has taken by using two algorithms. First algorithm is known as parametric study algorithm which provides optimal solution as a function of communication time. This algorithm studies the effect of a speed of communication network on a job execution time. Second algorithm considers system parameters as attributes to provide optimal solution in job execution. The recommended research work considers the communication time effect on job execution including queuing time and job transfer time. The basic assumption of this work is the node characteristics and the communication delay at each node should be known prior to job execution.

Leland and Hendrickson in [13] present a comparative study of static load balancing algorithms. In this article authors have mapped scientific applications to the parallel computing infrastructure using static Chaco graph partitioning software package. Performance evaluation of the algorithms is performed on the basis of computational cost and quality of partition. The research work studied the effect of coordinate information and application run time information on the selection of partitioning algorithm. This work did the comparison of Kernighan-Lin (KL), Spectral and Multilevel partitioning algorithms. The effect of these attributes on static load balancing is studied with experimental evaluation. This method considers only bisectional graph partitioning approach. In this article author did not discussed about method of making more than two partitions.

The performance analysis of Round Robin, Randomized, Central Manager and Threshold based approaches of static load balancing is done by Rahmawan and Gondokaryono in [14]. The basic attributes considered for load analysis are CPU, Memory and Hard Disk (HDD).

According to Rahmawan and Gondokaryono load on CPU for execution of total program is calculated by using CPI i.e Clock per instruction and time required to complete one clock cycle by CPU and queuing time.

CPU load is calculated using Equation (2.1).

$Load_{cpu} = L_{qCPU} / L_{qMAX}$ Equation (2.1)

Where, L_{qCPU} =Total jobs waiting in a CPU queue,

L_{qMAX} = Maximum length of a CPU Queue.

$Load_{mem} = N_{mused} / N_m$ Equation (2.2)

Where, N_{mused}= Amount of memory used,

N_m =Total amount of memory.

$Load_{HDD}= L_{qHDD} / L_{qHDDMAX}$ Equation (2.3)

Where, L_{qHDD}= =Total jobs waiting in a HDD queue

$L_{qHDDMAX}$= Maximum length of a HDD Queue.

Load on memory and hard disk is calculated using Equation (2.2) and Equation (2.3) respectively.

Total time required for job execution is calculated using Equation (2.4),

$T_{job}=T_{CPU} +T_{mem}+T_{HDD}$ Equation (2.4)

Where, T_{job} =Total time required for a job to execute,

T_{CPU}= Total CPU time

T_{mem} = Total Memory time

T_{HDD}= Total Hard Disk time

Where, T_{CPU} is calculated using Equation (2.5) as,

$T_{CPU} =T_{Execution}+T_{qdelay}$ Equation (2.5)

And T_{mem} is calculated using Equation (2.6) as,

$T_{mem}= N_{baccess}$ X L_{tm} Equation (2.6)

Where, $N_{baccess}$ = Total byte of accessed from memory and

L_{tm} =Memory Latency.

Rahmawan and Gondokaryono concludes in [14] as follows,

Prediction of the performance of Round Robin and Randomized algorithm is complex as it does not use the system state for task execution. According to author randomized algorithm has better stability than Round Robin algorithm.

Memory is not appropriate index of load balancing in central

scheduling algorithm because it has less percentage utilization than CPU and Hard Disk. CPU and Hard Disk are appropriate load indices to be chosen for load balancing because it has 9% to 15% utilization for CPU and 8% to 23 % utilization for hard Disk.

In threshold based algorithms also CPU utilization proves its importance as a good index for load balancing. As this experimentation is carried out for static load balancing it is very difficult to achieve the run time utilization of CPU as well Memory and Hard disk. Performance of this type of experimentation can be enhanced by using real time dynamic state information of system parameters i.e. CPU, Memory and Hard Disk. To collect dynamic values of system parameters is not possible using static approach so these types of systems can give better results using dynamic state information. The system works as centralized distributed system so coordinator node becomes a bottleneck in case of failures. The system exhibits results using only homogeneous environments.

2. Dynamic Load Balancing

Definition 3. (Dynamic Load Balancing) *"Dynamic load balancing policy is defined as, a policy in which decision of mapping of a task to a set of networked resources is taken at run time, based on the dynamic, instantaneous state information of networked resources involved in the computation"* .

In dynamic load balancing policy [15-48] the efficiency of task execution depends on the run time state information of underlying resources.

According to the comparison of static and dynamic load balancing policies given by author [15],

1. Dynamic policies are believed to have better performance than static policies.
2. Dynamic load balancing systems are more adaptive in nature than static load balancing systems.
3. Response time achieved in dynamic load balancing is more than static load balancing.
4. Dynamic systems are more complex than static systems and it requires more communication overhead than static systems.

Static load balancing system is more stable than dynamic load balancing. Static load balancing is more predictable than dynamic load balancing. Static load balancing is preemptive in nature. Dynamic load balancing can be non preemptive. Static load balancing mechanism degrades the performance of the system where rate of incoming load changes rapidly with respect to time. For the systems which are having dynamic behavior, dynamic load balancing strategies are more suitable than static load balancing. Table 2. provides the comparison of static and dynamic load balancing approaches in more detail.

Table 2. Comparison of Static and Dynamic Load Balancing

	Static Load Balancing	**Dynamic Load Balancing**
Decision Making	Static i.e. at compile time	Dynamic i.e at run time
Overhead	Less	More
Resource Utilization	Less	More
Predictability	More	Less
Preemptiveness	Inherently preemptive	non preemptive
Load Adaptability	Non Adaptive	Adaptive
Reliability	Less	Relatively More
Response Time	Less	Relatively More
Stability	More	Less

In general, a dynamic load-balancing mechanism consists of three rules-namely, information rule, transfer rule, and location rule [16]. The information rule defines strategy of collecting and storing the decision making information. The transfer rule defines when to transfer and whether to transfer the job. The location rule provides location of the nodes to or from which jobs will be transferred.

These three rules are either applied at a central location or at local sites in a distributed manner. Load in dynamic load balancing systems fluctuates at high rate so it is the responsibility of decision maker to take the appropriate load balancing decision based on the dynamic state information. Decision maker acts as a matchmaker which matches the dynamic demands of the user with the available set of resources at any time instance. The node selected by the decision maker after executing dynamic load balancing algorithm fulfills the dynamic need of the user.

In [16] author has enumerated the desired policies of a good dynamic load-balancing system as:

A. Provide short average job response time

Response time is defined as the time in between job submission and job completion. A good dynamic load balancing strategy should result in a short average job response time.

B. Induce low overhead

The communicational overhead which includes overhead caused by job transfer and inter process communication should be low.

C. Adaptability to changing load

System should adapt the dynamically changing behaviour of job arrival rate.

D. Be reliable.

System should be reliable so that it can provide trustworthy services to the intended user.

The decision making in dynamic load balancing can be performed at local node or at remote node. Based on the location of decision maker Dynamic Load Balancing is classified in two major classes:

1. Centralized Dynamic Load Balancing
2. Distribute Dynamic Load Balancing.

2.1 Centralized Dynamic Load Balancing

Definition 4. (Centralized Dynamic Load Balancing) *"Dynamic load balancing approach in which among the set of all computing nodes if only one node collects the instantaneous state information from all other nodes in a network, and if it is responsible for taking the decision of task to resource mapping then that system is called as a Centralized Dynamic Load Balancing System".*

In centralized dynamic load balancing [17-33] technique, central coordinator does the scheduling based on the dynamic state information collected from all other nodes in underlying network. General form of centralized dynamic load balancing consists of Master-Slave architecture in which master node is responsible for collecting the state information from all other slave nodes. This approach is also called as global decision making approach. Based on the collected state information master takes the decision of dynamic load balancing which results in finding a slave node for execution of a desired load.

In [17] author proposed Multitier Task assignment based on Pre-emptive Migration (MTTPM). Multi-Cluster Task assignment algorithm works for managing the dynamic service time requirements. Each host in a multitier system is assigned a predetermined service time. It considers equal probability of task arrival and assigns arrived task to first dispatcher in a system. First

Dispatcher assigns that task to a first host in First Come First Serve (FCFS) manner. If the service time of that task is less than or equal to the predetermined service time of that particular host then task departs the system and result is sent back to the client. If the service time of task is greater than predetermined service time of first tier, then that task is sent to second dispatcher. The process continues till the end of task execution. The working of this algorithm depends on the service time of task and if the estimation of service time goes wrong the system may lead to wrong decision.

In [18] simulation based experiment recommends use of DAG (Directed Acyclic Graph) based List scheduling algorithm which results in shorter schedule length size. It works with the heterogeneous computing environment and executes iterative scheduling algorithm to minimize the schedule length. It performs well only when the assignment ratio of task to processor is high.

In simulation based research work [19], classification of task is done using the centralized dynamic load balancing algorithm for improving overall resource utilization. This algorithm works for improving data retrieval efficiency and experimental results are compared with the results of least weighted connection algorithm. This algorithm does the classification of tasks considering only CPU time and I/O time as parameters of task classification. It did not consider the memory intensive task classification.

In [20] impact of heterogeneity is studied using centralized dynamic load balancing based master slave architecture. The work presented by author on the basic assumption of one port model. At any instant of time master can communicate with only one slave. According to author their does not exists any optimal deterministic algorithm for dynamic load balancing with heterogeneous network. It limits the degree of task distribution and execution. As master can communicate with only one slave at time message broadcasting is not possible.

In [21] simulation based experimentation is presented to achieve improvement in both make span time and cost of computation. This is achieved using centralized dynamic load balancing based Apparent Tardiness Cost Setups-Minimum Completion Time (ATCS-MCT) scheduling heuristic. The make span is the maximum time difference between the start and finish of a sequence of tasks between involved computers which is calculated, after completion of the last task. This algorithm works to reduce both the make span time and cost of job execution. The basic factors considered by this algorithm are execution time, communication time, weight and the deadline of the task execution. It assumes that before any resource demand, resource administrator is aware of type, amount and all other resource related information. But in practical load balancing situations this assumption adds limitations.

In research approach [22] generalized model of centralized heterogeneous distributed system is presented for checking service reliability and availability in distributed systems. Instead of actual physical computational nodes this experiment assumes availability of virtual machines. The methods used for reliability testing in virtual environment and those used in actual environment practically differ. It cannot be always reliable for practical situations of load balancing where the physical set up is required.

In [23] randomized load balancing is achieved using a sliding window technique in centralized dynamic load balancing environment. This method is based on supermarket model in which

customers are serviced in FIFO (First in First Out order). It assumes the situation when all the servers are loaded; the customer selects the random server for execution. The optimality of this method is based on the random selection approach so cannot be an appropriate method of load balancing cannot provide optimal solution in all cases.

In [24] central server open queuing network strategy is used to carry out comparison of different distributed load balancing approaches. This work exhibits the possibility of conversion of all sender initiated distributed load balancing techniques into a central server open queuing network. The basic parameters used for comparison are average execution queue length and the probability of load migration. This research article presents a comparison of the load balancing approaches based on CPU queue length. This research work concludes that open central quieting network can be appropriately used only when the system is symmetric and homogeneous. Proposed work provides the average response time close to the response time achieved by simulation. This method does not consider the task characteristics as well as overall state information of all nodes while load balancing which creates practical limitations.

Based on the variation in computational environment used for experimentation Centralized Dynamic Load balancing systems can be classified in two prominent classes:

1. Centralized Homogeneous Dynamic Load Balancing
2. Centralized Heterogeneous Dynamic Load Balancing

2.1.1 Centralized Homogeneous Dynamic Load Balancing System

Definition 2.4 ***(Centralized Homogeneous Dynamic Load Balancing)*** *Consider a dynamic load balancing system with a set of N computational nodes as shown in Equation 2.7,*

$N = \{N_1, N_2, \ldots\ldots, N_n\}$ .....Equation (2.7)

Where, 'μ' is a set of respective service rates (computing capacity of CPU) of the computational nodes from N_1 to N_n as shown in Equation 2.8 ,

$\mu = \{\mu_1, \mu_2, \ldots\ldots \mu_n\}$ ...Equation (2.8)

For all nodes from N_1 to N_n, if $\mu_1 = \mu_2 = \ldots\ldots = \mu_n$, then the present dynamic load balancing system is called as a Homogeneous Dynamic Load Balancing System.

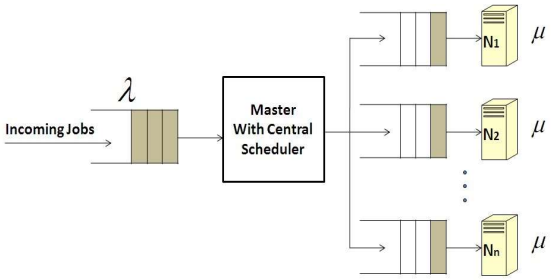

Figure 3. Centralized Homogeneous Dynamic Load balancing system

Figure 3 shows centralized homogeneous dynamic load balancing system [25-27]. Incoming jobs enter in a system through the queue associated with the master node.

Where 'λ' = Incoming rate of jobs in a queue.

Then all incoming jobs in a master queue are scheduled for execution on associated set of the slave nodes by the central scheduler at master node. As shown in figure 2.1 every node in a system is with same computational capacity i.e. μ.

$$\lambda = n\mu \quad \ldots\ldots \text{Equation (2.9)}$$

Equation 2.9 shows the stable state of the system where the rate if incoming jobs to a master node is exactly equals to the service rate of N slave nodes.

1. If $\lambda > n\mu$, this condition represents the situation when the incoming rate of jobs to a master is greater than the service rate of the system. In this case the queue length of a master should be large enough to store the waiting jobs, because the system cannot give justice to the incoming job as service rate of a system is less than incoming rate of jobs.
2. If $\lambda < n\mu$, this condition represents the situation when the nodes in a system remains idle as the service rate of the computing nodes is greater than the incoming job rates.

In order to achieve maximum throughput and make system steady an appropriate load balancing strategy is required. Dynamic load balancing decision in these systems does not depend on computational capability of the slave nodes, as all nodes are with the same computational capability. Decision of load balancing in these systems is based on a dynamic state of computational nodes. The master collects the resource utilization of each slave node at any instant of time and assigns the proportional load to a slave node for execution. Here central scheduler is responsible for resource allocation to every incoming job in a system.

Simulation based experimentation [25] considers a closed queuing network model with P homogeneous and independent processors. The basic objective of this experimentation is to study the performance issues associated with the scheduling mechanism in a parallel distributed computing system. This work is more focused on studying the number of tasks created per task to execute it in a parallel distributed system and also the task service demand variation with a time. This research work considers two step scheduling mechanism. In first step spatial scheduling is done for assigning the tasks to a processor queue. In second step temporal scheduling is carried out to define the sequence of execution for a task in a processor queue. It also addressed the issues created by an impact of overhead caused by the collection of global information in a system. This approach does not consider the memory as an attribute for load scheduling. As the network is homogeneous in nature, the computational and memory intensive tasks used to behave in a different manner which depends on the run time state of the system.

Research work in [26] recommends a homogeneous dynamic load balancing approach to carry out the trace driven simulation study using broadcasting. In this dynamic load balancing approach information of job arrival time, CPU and I/O demands are collected from production system and given input to a scheduling policy. Basic objective of this work is to study the effects of a system size (number of hosts in system), level of system load, values of attributes used for scheduling and immobile jobs (jobs that should be executed locally) on the performance of a load balancing mechanism. This work provides a solution for only CPU and I/O intensive tasks. It does not provide scheduling approach for a memory intensive task.

In [27] a homogeneous dynamic load balancing is applied to pipelined based applications using a concept of Dynamic Pipeline mapping (DPM).The key idea proposed in this work is, to use free computational resources by gathering fastest stages in a pipeline and boosts the performance of slowest pipeline stages by replication of free resources. This method does not provide a generic solution of load balancing because it is suitable for only those applications which are developed using framework/skeleton based tools only.

2.1.2 Centralized Heterogeneous Dynamic Load Balancing System

Definition 2.5 ***(Centralized Heterogeneous Dynamic Load Balancing)*** *Consider a dynamic load balancing system with a set of N computational nodes as shown in Equation 2.10,*

$N = \{N_1, N_2, ..., N_n\}$.....Equation (2.10)

Where, 'μ' is a set of respective service rates (computing capacity of CPU) of computational nodes from N_1 to N_n as shown in Equation 2.11,

$\mu = \{\mu_1, \mu_2, ... \mu_n\}$.....Equation (2.11)

For all nodes from N_1 to N_n, if $\mu_1 \neq \mu_2 \neq ..... \neq \mu_n$, then the present dynamic load balancing system is called as a Heterogeneous Dynamic Load Balancing System.

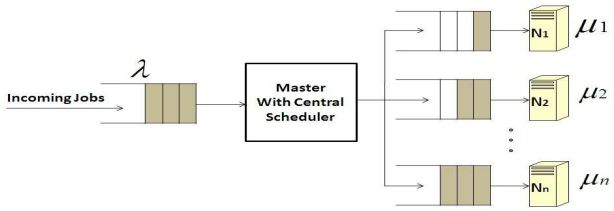

Figure 4. Centralized Heterogeneous Dynamic Load Balancing System

Figure 4. shows master-slave architecture based centralized heterogeneous dynamic load balancing system [28-33]. In this system incoming jobs enter in a system through the queue associated with the master node.

Where ' λ ' = Incoming rate of jobs in a queue.

Then all incoming jobs in a master queue are scheduled for execution on associated set of slave nodes by the central scheduler at master node. As shown in figure every node in a system is with different computational capacity i.e. from μ_1 to μ_n.

$\lambda = \mu_1 + \mu_2 + ... + \mu_n$ ....Equation (2.12)

The Equation (2.12) gives the condition when system is in stable state as the rate of incoming jobs is equals to the sum of service rate of all slave nodes.

If $\lambda > \sum_{i=1}^{N} \mu_i$, this condition represents the rate of incoming jobs is greater than the total service rate of system then the queue length of master node should be large enough to store the waiting jobs.

If $\lambda < \sum_{i=1}^{N} \mu_i$, this condition represents the situation when the rate of incoming jobs to master node is smaller than the total computing capacity of the system. In this case the slave nodes which have

respectively larger service rates will have more jobs in their queues and remaining nodes will remain in an idle state. This causes the starvation of slave nodes which have less service rates. To avoid this situation the load balancing is required in centralized heterogeneous dynamic load balancing system. Dynamic load balancing decision in these systems depends on variation in computational capability of computational nodes as all nodes are with different computational capability. Basic objective of load balancing system is to maintain the proportion of load and computational capability of node.

The basic difference between Figure 3 and Figure 4 is that in a homogeneous network, every node is with a same computational capacity μ but in heterogeneous network each node is having a different computational capacity i.e. $\mu_1, \mu_2, \ldots, \mu_n$. In homogeneous dynamic load balancing system the performance of a load balancing system depends on the algorithm which is chosen for the load balancing. In a heterogeneous dynamic load balancing mechanism, the load balancer should consider the variance in computational capacities of underlying network and should assign the load accordingly.

Research work in [28] recommends the dynamic task mapping techniques for matching and scheduling of independent tasks on a heterogeneous network. It presents two task mapping heuristics. First is online mode in which tasks are mapped on to a node in a heterogeneous network as soon as it arrives at mapper. In second heuristic batch of tasks is mapped to the nodes in a network. This work does not provide solution for load balancing of dependent tasks.

In [29] research work is carried out for heterogeneous multiple processor systems using queuing models to achieve dynamic load balancing. This research gives two approaches for dynamic load balancing which are, Deterministic and Nondeterministic. In a Nondeterministic approach state independent branching probabilities are used. Where as in a Deterministic approach criteria functions are used to enhance the performance of computing. This method does not consider the memory or I/O intensiveness property of the task while execution.

In [30] method of dynamic load balancing is recommended for both homogeneous as well as heterogeneous environment using Clustering Based Heterogeneous Earliest Finish Time (HEFT) with Duplication (CBHD) algorithm. CBHD is a combination of HEFT and Triplet Clustering algorithm.

HEFT algorithm is based on the DAG (Directed Acyclic graph) approach and it works in following steps:

1. The weight is assigned to each node of the graph to represent the average computation required by the task.
2. The weight on edges represents the average communication time required by the task.
3. Rank values for each node is calculated by traversing the graph in upward direction and summation of rank values of all possible immediate successor nodes.
4. Task assignment is done using the sorting of rank values in descending order of the rank values. Improvement in a HEFT algorithm is done by calculating the job processing limit value of each node.

Triplet algorithm works for the minimization of communication time and to minimize the number of clusters created while execution. The algorithm proposed by an author combines the features of both improved HEFT and Triplet algorithm to achieve the best possible task distribution methodology. The algorithm limits itself for only CPU intensive tasks and does not consider the memory parameter for decision making of task scheduling.

In [31] research work provides a dynamic load balancing based library to adapt parallel code on wide range of heterogeneous computational environment with a minimal overhead and a negligible programming cost. The method of dynamic load balancing proposed in this research article is implemented and tested only for a high performance computing using GPU networks so it is not applicable as a generic solution and for the low configuration networks.

In [32] dynamic load balancing technique is applied to a network of heterogeneous computing systems to achieve video encoding. The presented method of load balancing is application specific and cannot be used as a generic method of load balancing.

In [33] dynamic load balancing is applied to a network of heterogeneous computing nodes using the characteristics of previous tasks. It considers size of input data, operational intensity and hardware limitations. Analysis of results obtained is done by using Cellular Automata Finite Element (CAFE) Method. In this method of dynamic load balancing execution behavior of incoming task is depending on the historical analysis of properties of previous task. The accuracy of prediction becomes bottleneck for execution.

2.2.2 Distributed Dynamic Load Balancing

Definition 2.6 ***(Distributed Dynamic Load Balancing)*** *"If in Dynamic load balancing system decision of task to resource mapping is carried out at more than one nodes then that system is called as distributed dynamic load balancing system".*

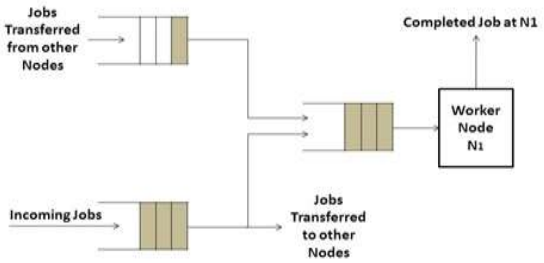

Figure 5. Distributed dynamic load balancing

Figure 5 shows decentralized dynamic load balancing system [34-37]. Unlike the centralized dynamic load balancing mechanism, this approach recommends the distributed role of coordinator. No any single node is responsible for the decision making. Incoming jobs submitted to computational node are executed either at that node or sent to the remote nodes for execution. The decision of job transfer is taken by the load balancer which is available at every node. The distributed role of a load balancer requires the information from neighboring nodes for making the decision of load balancing. Message passing mechanism is used to exchange the load information.

The communication cost required for this strategy limits the performance of distributed dynamic load balancing.

In [34] an application of game theoretic model is used in the distributed system using decentralized load balancing approach. In this approach the node plays game to balance the computational load in a network. This work is carried out using simulation based experimentation so the real time practical load balancing approach is missing in this work.

In research work [35] Game Theory based approach is used to solve the cooperative and no cooperative routing problems. This approach provides a solution to the distributed load balancing problem using team optimization approach. It uses approach of global team, cooperative team and non-cooperative team for solving the problem. The presented method limits by assumption that total arrival rate of customers to the system is always less than total service rate of the system. This assumption creates contradiction when it is used for real world problem solving purpose.

In [36] Predictive, Decentralized Load Balancing Approach based on CORBA (Common Object Request Broker Architecture) is recommended. Basic objective of this work is to minimize the problems in load monitoring and load prediction techniques. The basic limitation of this work is it does not consider the system parameters for load balancing. It works well only for CORBA compliant applications.

The author in [37] recommends a distributed load balancing policy for a multicomputer system. System consists of three algorithms to achieve load balancing:

1. Local load balancing algorithm which is used by each processor in a system for its own load balancing process.
2. Information Exchange algorithm which sets rules for inter information exchange process among the processors.
3. Process migration algorithm to maintain load balancing in a system by dynamic task migration from overloaded to under loaded processors.

This method of load balancing considers only CPU bound and I/O bound processes it does not considers the memory bound processes.

2.2.3 Adaptive Dynamic Load Balancing

Adaptive policy of dynamic load balancing [38-43] reacts to the change in the system state by collecting considerable system state information and finds the best possible choice of decision making. Since, load on the system is bound to change with time; an adaptive load balancing policy is the best to work with, as it addresses the problem of changing load [38].

Research work in [38] provides solution to the issues in parallel computing in the homogeneous as well as heterogeneous system. This approach gives a centralized adaptive threshold based dynamic load balancing strategy, for a parallel multiprocessor system. In this approach threshold values varies with the change in system load. Each incoming job is assigned to a scheduler. As the jobs are submitted continuously the load on the system varies with the time. The mean load of overall system is calculated and the values of lower and upper thresholds are depending on mean load are assigned. The difference in mean average load in the system readjusts the threshold values of lower and upper thresholds.

In [39] simulation of a Hypergraph-based dynamic load balancing mechanism is provided. The method of periodic repartitioning for the adaptive scientific computational data at dynamic time is proposed. The basic objective of this work is to reduce overall communication cost of application which results in effective rise in execution time.

In simulation based experiment [40] the author has proposed an adaptive dynamic load balancing strategy. The core part of this work is to distribute computational load among the neighboring processors. This simulation is carried out for the dynamic load balancing of particle simulation of three dimensional systems.

Research work in [41] recommends a neural network based server traffic prediction algorithm known as Radial Basis Function. This work recommends three layered neural network. It considers Input layer, Hidden layer and Output layer. In which weights on edges of neural network represents the actual resource utilization and updated according to the real resource usage. The weights assigned to edges of neural network are readjusted as the real work load differs. It considers load on resources like CPU, Memory and Hard disk. System always tries to maintain a normalized load using the resource weights.

Adaptive dynamic load balancing mechanism is presented in [42] which recommend a Distributed Approximate Optimized Scheduling Algorithm with Partial Information (DAOSAPI). This simulation based study gives an agent based distributed load balancing mechanism. The proposed DAOSAPI algorithm combines the features of distributed mode, approximate optimization and agent set scheduling approach. It gives short execution time by using DAOSAPI algorithm. It proves its advantage over the static agent scheduling mechanism.

Object-oriented, parallel finite element framework (OOFEM) with dynamic load balancing is presented in [43]. This research work gives a design and implementation strategy of parallel load balancing framework in object oriented finite element environment. Base of parallelization is made by domain decomposition and message passing mechanism. It uses OOFEM framework which is an open source finite element solver and written in C++ language. This experimental work results into a single framework, with the integration of individual components of the parallel, h-adaptive, dynamically load balanced analysis. Computational work assigned to a computational node is considered as a function of both number of processors available and the size of buffer available at that node. This work recommends two types of buffers i.e Static and Dynamic. Static buffer is having fixed size whereas the dynamic buffer is having dynamic size. The work load assignment to particular node checks the packet size and confirms the availability of space at buffer. If buffer is not available the dynamic buffer size is allocated to that node as per the requirement.

The common limitation observed in all adaptive load balancing algorithms is that, the efficiency of task execution in each algorithm depends on the mechanism of threshold adaptation. So threshold adaptation method becomes bottleneck for overall load balancing mechanism.

2.2.4 Resource Based Dynamic Load Balancing

Resource based dynamic load balancing [44-48] considers the dynamic state change of the resources like CPU, RAM, I/O. It becomes necessary to consider the computational cost of every job based on the dynamic utilization of the prime computational resources. The present thesis

contributes a Ranking Based Dynamic Load Balancing algorithm which considers CPU and RAM as the basic resources for dynamic load calculation. CPU queue length, I/O queue length are commonly used parameters for load estimation. The load in these cases is estimated as a function of time i.e. time required for completing the task. In centralized dynamic load balancing mechanism, the master node collects the values of resource utilization from all slave nodes and then takes a decision of load balancing. The master chooses a victim node which can give better efficiency and which has fewer loads for execution. Methodology of node selection depends on dynamic load balancing algorithm which is used for implementation.

Research work presented in [44], deals with the problem of dynamic load balancing by using weight based algorithm. In this approach dynamic load balancing is applied to a cluster of servers which provides platform for execution of microservices. This approach considers CPU, RAM, Hard Disk and number of server connections as basic inputs to a load balancing algorithm. Algorithm first calculates the change in resource attributes and calculates weight of each resource. Load calculation is done using the summation of dynamic loads on each resource. There are several advantages of this research work:

1. The load's weight value is not fixed it changes dynamically. This fact reflects in more realistic approach of load calculation.
2. It does not consider the fixed probability of load distribution to available servers. So when task comes to the system it is directly assigned to server with minimum load without checking the probability of load distribution.
3. Factors considered for server load calculation gives more realistic load calculations.

The distribution is limited only for micro service cluster. The load index values for resources are obtained from service provider but it is not realistic approach because the resource demands of different applications are not unique. So the resource index values should be application dependent.

In [45] LAN based dynamic load balancing is achieved using Platform for Mobile Agents Distribution and Execution (PMADE). This research work presents an idea of dynamic load balancing based on CPU, Memory and I/O. Present work elaborates the use of I/O attribute in a dynamic load calculation of data intensive applications. Each node's mobile agent calculates the load on that particular node using value function. Based on calculated load task execution is carried out dynamically. This solution requires the more communication time as the frequency of mobile agent creation and servicing is large.

In [46] dynamic load balancing is achieved for network based servers using the Open flow switches. Wildcards installed in each server controls the resource utilization. Floodlight is used as open flow controller and Mininet is used as a network emulator. It considers dynamic load balancing based on CPU usage and active connections to the server.

In [47] author recommends a grid based distributed dynamic load balancing system. It uses network aware scheduling mechanism. This work considers heterogeneity in terms of CPU speed, architecture and network speed. Recommended Dispersion algorithm is divided in two parts i.e. control and scheduling. Control part manages the mechanism of information collection.

Scheduling does the task allocation based on the dynamic information collected by the control part. Each node stores the information of neighboring node using message passing mechanism. Each job submits its CPU and I/O requirement to the algorithm. Based on this information it is determined that whether the job is CPU intensive or Memory intensive and according to that resource mapping is done by the scheduler. Because of the distributed nature of algorithm used, this approach takes maximum efforts for maintaining and collecting the information of dynamic state of the system than the scheduling.

Cost function F_c is calculated for each node by using Equation 2.13 in the system using the dynamic state information of that node.

$F_c = C_J * C_N + I_J + I_N + S_J / S_N$ .....Equation 2.13

Where, C_J =CPU time required by the job,

C_N =is the current CPU utilization of the candidate node,

I_J = is the I/O time required by the job,

I_N = is the I/O utilization of the candidate node,

S_J = is the size of the job in bytes,

S_N = is the application bandwidth in bytes per second between the node at which the job arrives, and the candidate node.

In current approach the job requirements i.e. CPU, RAM and I/O required to process the job are submitted by the user. Based on that it is decided that whether the job is CPU intensive or RAM intensive. If user submits the job requirements without any prior pre-processing, the result of the system will be redirected in a wrong direction. Communication cost required for five nodes, is calculated. Any node can receive 1.8 job packets per unit time and can send 4.8 packets per unit time.

Research recommendations in [48] provide a feedback control mechanism based dynamic load balancing. This work considers only I/O and Memory intensive work load distribution among the nodes in a cluster. Dynamic load balancing is achieved for networked resources by calculating the weights and I/O by adjusting buffer size.

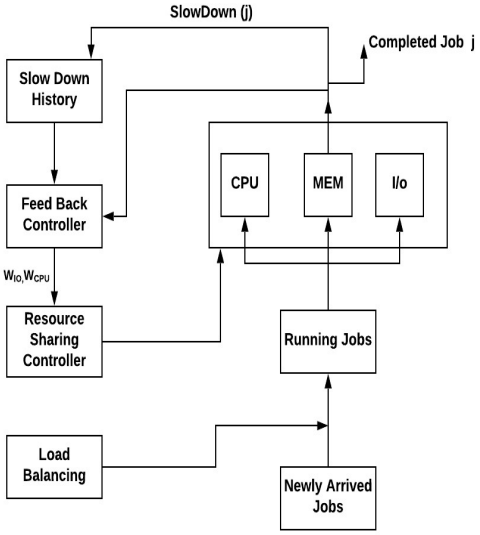

Figure 6. Architecture of Feed Back Control System

Figure 2.4 shows the basic control blocks involved in feedback control mechanism. It consists of Load Balancing mechanism, Resource Sharing Controller and slow down history manager. After completion of job execution, the slow down time of newly completed job and slow down history are given as input to feedback controller. Feedback controller then determines control action for the change in weight of CPU as well as IO. Feedback controller tries to attain the basic assumption that sum of W_{CPU} and W_{IO} should be equal to 1.

Feedback controller manage W_{CPU} and W_{IO} in three steps. In first step feedback controller calculates the slowdown time S_j of recently completed job j. In second step S_j makes its entry in the history table. Here S_j represents a pattern of recent slowdowns. S_{avg} represents the average of slowdowns in history table. In third step it is checked whether the $S_{avg} > S_j$. If so it reflects that the performance is increased therefore W_{IO} is increased otherwise W_{IO} is decreased. If $S_{avg} < S_j$ it reflects that performance is decreased which suggest to increase value of W_{IO} and vice versa. The disadvantage of this method is, it do not analyse the reason of slow down time, without analysing the reason it tries to adjust the resource weights this fact may leads the overall objective in wrong predictions.

REFERENCES

[1] Wearesocial Blog, https://wearesocial.com/us/blog/2018/01/global-digital-report 2018.

[2] IT @ Intel white paper: Data Center strategy Leading Intels Business Transformation, November 2017, https://www.intel.com/content/dam/www/public/us/en/documents/white papers/data-center-strategy-paper.pdf

[3] Jean s. Bozman, Katherine Broderick,white paper, server refresh: meeting the chang ing needs of enterprise it with hardware/software optimization, pages 1-13,July 2010.

[4] Tareq Ahram, Arman Sargolzaei, Saman Sargolzaei, Jeff Daniels, Ben Amaba, Blockchain technology innovations, IEEE Technology & Engineering Management Con ference, 2017.

[5] Najmul Hassan, Saira Gillani, Ejaz Ahmed, Ibrar Yaqoob, and Muhammad Imran, The Role of edge computing in internet of things, IEEE communications magazine, pages 1-6, 2018.
[6] Pradeep K. Sinha, Distributed operating systems concepts and design, the institute of electrical and electronics engineers, IEEE Communications Society, Computer Com munications and Networks, Second Edition, 1997 .
[7] Joshi Narayan Arvindkumar, Development of Algorithms for Optimized Process Migration for Load Balancing in Distributed Systems, 2015. 166
[8] T. Deepa, Dhanaraj Cheelu, A Comparative study of static and dynamic load balanc ing algorithms in cloud computing, International conference on energy, communication, data analytics and soft computing, 2017.
[9] Daniel Grosu and Anthony T. Chronopoulos, Algorithmic mechanism design for load balancing in distributed systems, IEEE Transactions On Systems, Man, and Cybernet icsPart B: Cybernetics, Vol. 34, No. 1, February 2004.
[10] Xueyan Tang and Samuel T. Chanson, Optimizing static job scheduling in a net work of heterogeneous computers, IEEE Proceedings 2000 International Conference on Parallel Processing ,pages 373-382,2000.
[11] Amit Chhabra, Gurvinder Singh, Qualitative parametric comparison of load balanc ing algorithms in distributed computing environment , IEEE International Conference on Advanced Computing and Communications, pages 58-61,2006.
[12] Asser N. Tantawi, Don Towsley, Optimal static load balancing in distributed com puter systems, Journal Of The Association for Computing Machinery, vol. 32, no. 2, pages 445-465, 1985.
[13] Robert Leland, Bruce Hendrickson, An empirical study of static load balancing algorithms, Proceedings of IEEE Scalable High Performance Computing Conference, pages 682-685 ,1994 .
[14] Hendra Rahmawan, Yudi Satria Gondokaryono, The Simulation of static load bal ancing algorithms, International conference on electrical engineering and informatics, pages 640-645, august 2009.
[15] P. Beaulah Soundarabai, Sandhya Rani A, Ritesh Kumar Sahai1, Thriveni J,K.R. Venugopal and L.M.Patnaik, Comparative study on load balancing techniques in dis tributed systems, International Journal of Information Technology and Knowledge Management, Volume 6, No. 1, pages 53-60, 2012. 167
[16] Hwa-Chun Lin and C. S. Raghavendra, A Dynamic load-balancing policy with a central job dispatcher (LBC), IEEE transactions on software engineering, vol. 18, NO. 2, 1992.
[17] Malith Jayasinghe, Panlop Zeephongsekul, Albert Y. Zomaya, Zahir Tari, Task as signment in multiple server farms using preemptive migration and flow control, Journal of Parallel Distributed Computing, pages1608-1621 Volume 71, 2011.
[18] G.Q. Liu, K.L. Poh, M. Xie, Iterative list scheduling for heterogeneous computing, Journal of Parallel and Distributed Computing, 65, pages 654-665, 2005.

[19] Zhuang Liu, Guang Dong, Hongwei Yang, Guannan Qu, Hongbin Wang and Lin Chen, Experiment research of dynamic load balancing algorithm based on task classi fication for data retrieval, Advanced materials research, Volume 684 , pages 559-562, 2013.
[20] Jean-Francois Pineau , Yves Robert , Frederic Vivien, The Impact of heterogeneity on master-slave scheduling, Parallel Computing Volume 34 , pages 158-176, 2008.
[21] Li-ya Tseng, Yeh-hao Chin , Shu-ching Wang, A Minimized makespan scheduler with multiple factors for grid computing systems, Expert systems with applications, Volume 36 , pages 11118-11130, 2009.
[22] Y.S. Dai, M. Xie, K.L. Poh, G.Q. Liu, A Study of service reliability and availability for distributed systems, Reliability engineering and system safety, Volume 79, pages 103-112, 2003.
[23] Yibing Wang, Robert Hyatt, An Improved Algorithm of Two Choices in Random ized Dynamic Load-Balancing, Proceedings of the Fifth International Conference on Algorithms and Architectures for Parallel Processing , 2002 .
[24] Ishfaq Ahmad, Arif Ghafoort, and Kishan Mehrotra, Performance prediction of dis tributed load balancing on multicomputer systems, Proceedings of the 1991 ACM/IEEE Conference on Supercomputing, pages 830-839,1991 168
[25] Helen D. Karatza Ralph C. Hilzer, Parallel job scheduling in homogeneous dis tributed systems, Simulation, Volume 79, Issue 56,pages 287-298, 2003.
[26] Songnian Zhou, A Trace-driven simulation study of dynamic load balancing, IEEE Transactions on Software Engineering, Volume 14, Issue 9, pages 1327-1341,1988 .
[27] A. Moreno , E. Cesar , A. Guevara , J. Sorribes , T. Margalef, Load balancing in homogeneous pipeline based applications, Parallel computing, Volume 38, pages 125- 139, 2012.
[28] Muthucumaru Maheswaran, Shoukat Ali, Howard Jay Siegel, Debra Hensgen and Richard F. Freund, Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems, Proceedings of the 8th heterogeneous comput ing workshop (hcw 99), pages 1-15,1999.
[29] Yuan-Chieh Chow and Walter H. Kohler, Models for dynamic load balancing in a heterogeneous multiple processor system, IEEE Transactions on computers, Volume c-28, Issue 5, pages 334-361, 1979.
[30] Doaa M. Abdelkader , Fatma Omara, Dynamic task scheduling algorithm with load balancing for heterogeneous computing system, Egyptian Informatics Journal Volume 13, pages 135-145,2012.
[31] Alejandro Acosta , Vicente Blanco, Francisco Almeida, Dynamic load balancing on heterogeneous multi-GPU systems, Computers and Electrical Engineering, Volume 39 , pages 2591-2602,2013.
[32] Svetislav Momcilovic, Aleksandar Ilic, Nuno Roma, and Leonel Sousa, Dynamic load balancing for real-time video encoding on heterogeneous CPU+GPU systems, IEEE transactions on multimedia, Volume 16, No. 1, pages 108-121, January 2014.

[33] Lukasz Rauch, Daniel Bachniak, Dynamic load balancing for cafe multiscale mod elling methods for heterogeneous hardware infrastructure, International conference on computational science, ICCS 2017, pages 1813-1822 , 2017. 169
[34] Sandip Chakraborty, Soumyadip Majumder, Diganta Goswami. Approximate con gestion games for load balancing in distributed environment, International Workshops on Distributed Systems, IIT Kanpur, India, 2010.
[35] Anastasios A. Economides and John A. Silvester, A game theory approach to co operative and non-cooperative routing problems, SBT/IEEE International Symposium on Telecommunications, pages 597-601, 2002.
[36] Dazhang Gu, Lin Yang, Lonnie R. Welch, A predictive decentralized load balancing approach, Proceedings of the 19th IEEE International parallel and distributed process ing symposium , 2005.
[37] Amnon Barak and Amnon Shiloh , A Distributed Load-balancing Policy for a Mul ticomputer, Software-Practice and Experience, Volume 15(9), pages 901-913 , 1985.
[38] Taj Alam, Zahid Raza, A Dynamic Load Balancing Strategy with Adaptive Thresh old Based Approach, 2nd IEEE International Conference on Parallel, Distributed and Grid Computing, pages 927-932, 2012.
[39] Umit V. Catalyurek, Erik G. Boman, Karen D. Devine, Doruk Bozdag, Robert Heaphy and Lee Ann Riesen, Hypergraph-based Dynamic Load Balancing for Adap tive Scientific Computations, IEEE International Parallel and Distributed Processing Symposium pages 1-11, 2007.
[40] Christoph Begau, Godehard Sutmanna, Adaptive dynamic load-balancing with ir regular domain decomposition for particle simulations, Computer Physics Communi cations, pages 51-61, Volume 190, 2015.
[41] Chen FU, Li-Jun ZHANG, Adaptive Load Balancing Strategy Based on LVS, ITM Web of Conferences, 2017.
[42] Qingqi Long , Jie Lin, Zhixun Sun, Agent scheduling model for adaptive dynamic load balancing in agent-based distributed simulations, Simulation Modelling Practice and Theory, Volume 19, pages 1021-1034, 2011. 170
[43] B. Patzak , D. Rypl, Object-oriented, parallel finite element framework with dynamic load balancing, Advances in Engineering Software , Volume 47 pages 35-50, 2012.
[44] Chang Yi, Xiuguo Zhang, Wei Cao, Dynamic Weight Based Load Balancing for Microservice Cluster, Proceedings of the 2nd International Conference on Computer Science and Application Engineering, 2018.
[45] Neeraj Nehra, R.B.Patel, Towards Dynamic Load Balancing in Heterogeneous Clus ter using Mobile Agent, International Conference on Computational Intelligence and Multimedia Applications, pages 15-21, 2007.
[46] S.Wilson Prakash P. Deepalakshmi, Server-based Dynamic Load Balancing, International Conference on Networks & Advances in Computational Technologies, pages 25-28, 2017.

[47] David Solomon Acker, Sarvesh Kulkarni, A Dynamic Load Dispersion Algorithm for Load-Balancing in a Heterogeneous Grid System, IEEE Sarnoff Symposium, 2007.
[48] Xiao Qin, Hong Jiang, Yifeng Zhu, and David R. Swanson Dynamic Load Balancing for I/O and Memory-Intensive Workload in Clusters Using a Feedback Control Mechanism, Euro-Par, LNCS 2790, pages. 224-229, 2003.