CHAPTER 2

LITERATURE SURVEY

2.1 OVERVIEW

In this chapter, we provide the literature of related works in the areas pertaining to Distributed Systems. The survey is majorly based on the performance improvement in various levels of distributed systems. We have reviewed research works done on election algorithm which focuses on the synchronization of heterogeneous nodes. We also have surveyed the load balancing aspects of various servers of the system. Towards the end we have referred the literature on the distributed file framework with Hadoop open source.

2.2 ELECTION ALGORITHMS

Sung Hoon Park [1] has proposed a concept called Failure Detector which works as an independent module with a function that detects crash and recovery of a node in a system. This report can be given to any process at request. The author modified the bully algorithm using the failure detector. The performance of the system goes down because of the overhead of Failure Detector. And as the Failure Detector is the centralized component, it has the problem of single source failure and also creates bottleneck situation to access the module.
Sandipan Basu [2] has addressed the issue of bully algorithm and proposed his modified algorithm. In the original bully algorithm, when the leader process is crashed, immediately the new leader is elected. But, if the old leader process comes back, it once again initiates the election. The author suggests that there need not be another election; instead, the old leader process can accept the new leader process by sending the new request of who the leader will be, to its neighbour. In the next round of election, it can try becoming the leader. Muhammad Mahbubur Rahman et al., [3] have also proposed a modified bully election algorithm. In their paper, they say that the bully algorithm has $O(n^2)$ messages which increases the network traffic. In the worst case, there will be $n$ number of elections occur in the system which again in turn yield a heavy network traffic. They have proposed the same algorithm but with Failure Detector, Helper processes to have unique election with the Election Commission.

Chang Young Kim et al., [4] have proposed the election protocol for reconfigurable distributed systems which was based on Garcia Molina’s bully election algorithm. As the election algorithm is majorly used in many real time applications such as group communication, atomic commit and data management as many distributed protocols depend on election protocol. The authors have suggested a model for the mobile computing environment as this area is more vulnerable to failures due to its dynamism in nature. The actual election is run by the base stations making the protocol, energy efficient.

M S Kordafshari et al., [5] discussed the drawback of synchronous bully algorithm and modified it with an optimal message algorithm. The proposed algorithm has only one election at any point of time, which brings down the number of messages being exchanged drastically. But this modified version of bully algorithm is also time bounded. It has $O(n^2)$ complexity during the
worst case and all the processes should know the priority number of other existing processes.

Sepehri M et al., [6] have dealt with the distributed leader election algorithm for a set of processes which are connected by a tree topology. They have used a heap structure to elect the coordinator processes. The process which is stored at the root level will initiate the coordinator protocol and broadcasts the others that from now on it is going to be the new coordinator. The authors have tested the algorithm and achieved a time complexity of $O(p) + \Theta(\log p)$ where $p$ is the number of processes. The limitation of heap structure is always present with their algorithm.

Cuibo Yu et al., [7] have proposed a different idea on SN(Super Node) election algorithm based upon district partition which divided the whole overlay into $k$ small districts and the algorithm uses distributed and parallel computing in these small units. Their model decreases the message complexity to $O(n^2/k)$ as well as increases the speed of the election process. The elected SN is also distributed evenly in the whole overlay.

Ben Ari [8] has proposed a new approach with fault tolerant mechanisms based on heap for coordinator [9] finding in wireless environment. They have compared the algorithms running time and message complexity with the existing algorithms. The algorithms works better as it uses maxheap method to find the leader process. Basim Alhadidi [10] et al., have analyzed the existing Bully algorithms and few of the modified Bully algorithms. Their main intention to this research work is to reduce the time to decide the next eligible leader when the original leader fails.

Zargarnataj [11] has developed an algorithm which is based on Bully election algorithm with an additional feature of an assistant to the new leader. If the present leader node crashes, the assistant leader would become the new leader.
without any overhead of election. Whenever any process realises the absence of the leader, it immediately sends a message to the assistant leader to alert it. When the assistant leader receives any such message, it confirms the unavailability of the leader by timeout message and if it is true, it broadcasts the leader message to all the processes. But the limitation of this algorithm is that if the assistant leader is also not available then there is a lot of messages sent and time delays to invoke the election algorithm once again.

Svante Janson [12] has developed a leader election algorithm in which a set of distributed objects such as people, computers, processes etc., try to elect one object as their leader. The election process is randomized, that is, at every stage of the algorithm. The objects that survived so far flip a biased coin, and those who received, say a tail will survive for the next round. This election process continues until only one of objects remains. They have evaluated the limiting distribution and the first two moments of the number of rounds needed to select a leader process using the algorithms such as: analytical poissonization and depoissonization, Mellin transform and complex analysis.

Garcia-Molina [13] has proposed the Bully Algorithm for a leader election which in the worst case requires $O(n^2)$ messages. However, in the best case, the Bully Algorithm requires only $(n-2)$ messages. Tanenbaum [14] and Silberschatz et al.[15] have described A Ring Algorithm for doing the election in the distributed systems. In the Ring Algorithm, irrespective of which process detects the failure of the coordinator and initiates an election, an election always requires $2(n-1)$ messages. But if the election is started by almost all the processes simultaneously then we will have about $n$ numbers of $2(n - 1)$ messages which is of order $n^2$. Gallager et al., [16] have developed a leader election algorithm for arbitrary networks whose message complexity is $O(N \log (N+E))$ where $N$ is the number of nodes and $E$ is the number of edges in the
network. Santoro in [17] has studied how the knowledge of topology of the network and the orientation of the links in the network affect the message complexity of leader election algorithms in networks.

Seok Hyoung Lee et al., [18] have studied the bully algorithm and they have proposed the fast bully algorithm by identifying the limitations of the existing bully algorithm. The bully algorithm is having a higher message complexity and time complexity. When the coordinator process fails, and a new process of higher order joins the system during the failure of the leader process, according to the algorithm, the new job with the higher priority number becomes the leader. But without knowing all these activities, when the old leader process joins back the system after recovery, it bullies the other processes saying it is going to be the leader process once again. This situation leads to the presence of two leaders at the same time. The authors have suggested a new algorithm named “Fast Bully Algorithm” and eliminated the problem of the presence of two leaders in a group. Their algorithm proves better performance with respect to time complexity and message complexity which of $O(n)$.

Amir Azim Sharifloo et al., [19] have discussed the concept of reliable multicasting to hold election as it is very important for large groups which would have lots of reliability problems. The researchers have used hierarchical structures to solve these problems. A problem in hierarchical structure is holding election when coordinator or a local coordinator crashes. The existing election algorithms such as bully and ring do not support hierarchical structure and so they have introduced a new election algorithm for the cluster groups in the hierarchical structure with the help of global election and big election messages.

As the coordinator election algorithms are inevitably used in all the distributed systems, there is a broad research literature available in various network topologies. But there are very limited research for this major issue in mobile ad
hoc networks. The algorithms discussed in [21] by K. P. Hatzis et al., are taxonomied into noncompulsary and compulsory protocols. Their protocols require each node to meet the other nodes and exchange their information to choose the new leader. The algorithms presented in [22], [23] and [24] are based on a routing algorithm called TORA [25] wherein nodes keep updating the variable height which is maintained locally to decide the next leader by pointing at the nodes in decreasing order of their identification numbers.

2.3 LOAD BALANCING IN DISTRIBUTED SYSTEMS

Considerable amount of research is done on load balancing with various factors of the system. The authors have focused their load balancing on P2P systems with high concentration on middleware-level based load balancing strategies over a large dynamic peer-peer network. The studies have been done on both structured and unstructured P2P systems [29][30][31]. The load balancing algorithm based on the network utilization has been developed by Saito et al., [29] which controls the traffic flow in the network and hence minimizes the over utilized and underutilized network links.

The various system level resources like the memory, processor power and the system files have been considered for the operating system level load balancing by R Krahl et al., [32] for P2P network. The load balancing taxonomy focuses on various factors such as centralized and decentralized, client oriented and server oriented, heuristic, deterministic and non-deterministic algorithms which have been explained in detail by A R Karthik [33]. Triantafillou et al. [34] have developed load balancing algorithms for P2P systems which distributes global meta-data contents over a two-level hierarchy unstructured system.

Barazamdej et al., [35] developed two methods for the load balancing in the distributed systems which were based upon hierarchical structure. They have developed the algorithms dynamically to which work on the current load
weight. They also have improved the round robin algorithm by having the high priority for the servers which has lesser load state. Since the centralised approach is used, the algorithms suffer the traditional problems of single source of failure and the bottle neck delays. The algorithms that are based on hash tables with the homogenous objects, virtual servers with heterogeneous objects are also discussed and a comparison has been carried out and Adaptive CORBA load balancing model is presented by Othman and his team [36]. Yang Jiao et al., [37] have presented the problems pertaining to the load balancing algorithms. They have mainly focused on unscientific and inaccurate algorithm, imperfect and impractical load-balancing system design. They also have proposed and implemented web server load balancing algorithm. Grosuand D et al., [38] have designed a load balancing algorithm which is very much dynamic wherein the users and the systems are eligible to manipulate it according to their interest. Their study was based on the techniques from mechanism design theory. They have also proposed a fair load balancing protocol.

Lei Shi et al., [39] discussed the limitation of flow based hashing algorithms and proposed flow slice algorithm that cuts off every flow into slices through which balancing the load with fine granularity.

In today’s redundant high-availability computer systems it is common that the incoming network traffic is distributed on network level by using any one of the commonly used network load balancing algorithms (like: random allocation, round-robin allocation, weighted round-robin allocation, etc.) [43]. These algorithms predominantly use network parameters of incoming traffic to make decisions where to forward traffic, without considering any other information like current load of application or database servers.

Load balancing is very essential in distributed computing systems to improve the quality of service by managing customer loads that are changing over time.
Branko Radojevic et al., [44] have presented a load balancing model in which the request demands of incoming requests are optimally distributed among available system resources to avoid resource bottlenecks as well as to fully utilize available resources.

The main purpose of load balancing techniques, either static or dynamic, is to improve performance by redistributing the workload among available server nodes. Hisao Kameda et al., [45] have explored that the dynamic load balancing techniques react to the current system state, whereas static load balancing techniques depends only on the average behaviour of the system in order to balance the workload of the system. This makes dynamic techniques necessarily more complex than static one. But, dynamic load balancing policies have been believed to have better performance than static ones through the study of J. Li et al., [46]

Zeng Zeng et al., [47] have proposed Rate-Based and Queue-Based Dynamic Load Balancing Algorithms. Their study suggests that if the load state is not in the overloaded level then, the process is assigned locally. If it is overloaded then, a remote under loaded server node is selected and assigned the process. These algorithms have less inter process communication but more number of local process allocations. The later decreases the overhead of remote process allocations and the overhead of remote memory accesses, which leads to improved performance [48][49].

Recently, analytical approaches of the Dynamic Load Balancing Techniques used in parallel computing systems and Load Balancing across Near-Homogeneous Multi-Resource Servers are presented by researchers [50]. An Empirical Study and Analysis of the Dynamic Load Balancing Techniques used in Parallel Computing Systems is presented by Parveen Jain et al., [51]. None of
these approaches, however, addresses load balancing or optimization of computing system resources.

In the central manager allocation algorithm [52] by P.L. McEntire et al., a central manager node selects the host for new process. A server node which is least loaded when compare to all other nodes in the group of servers is selected when process is created. In order to select the least loaded server, the load manager uses the server’s current system load information which will be locally stored on the manager node. This information is updated by all the remote server nodes which are in the group by sending a message every time when the load on them changes.

2.4 LOAD BALANCING WITH AVAILABILITY CHECKER AND LOAD REPORTERS

Elena Renda et al., [53] have addressed load balancing to reduce the network traffic in sensor network. This research was exclusive for the sensor network, where the sensed data is stored in the sensor node itself. This data does not travel to the base station and through queries to the geographic hash table, the data would be accessed. Complex geometric transformations were used along with complicated routing algorithms for accessing the data. The authors have designed “reverse engineer hash function” to store the data in the node, and also proposed load-aware key ranges for the generated data. They have come out with analytical and heuristic approaches were used to query the data.

Gaochao Xu et al., [54] have proposed game theory to the existing load balancing algorithms in the public cloud. The authors have focused on the cloud partitioning types and based on the various types of the partitioning, the different strategies have been applied to balance the load which accesses the servers in the cloud. They have also discussed the rules to divide the cloud as it requires a dedicated methodology to make the divisions in cloud. They have
suggested that the division rule should be based purely on the geographic location irrespective of whether the nodes in the cluster are nearer or farther. The cloud partition balancers should get refreshed often. The refreshing duration should neither be very small which would force a heavy load on the system affecting its performance nor be too long, which looses the important in-between information about the cloud’s partition.

A comparison of the existing load balancing techniques in distributed computing systems is presented [55] with both static and dynamic load balancing algorithms and done a comparison of various static algorithms alone and the dynamic algorithms alone. Finally static and dynamic algorithms were compared with various parameters such as reliability, complexity, stability, response time etc. The merits and demerits of these algorithms were also presented. It is very much essential to balance the load among the servers to avoid bottleneck situation and to have maximum utilization of server resources [56] [57].

“Load balancing” describes the even distribution of the jobs to various available servers. In distributed computing systems, the customer requests have to be given high preference and all of them have to be serviced without much delay. To provide the better service, the server processes are replicated into many copies and all of them together serve the customer requests. It is very much essential to balance the load among the servers to avoid bottleneck situation and to have maximum utilization of server resources [58] [59]. Load balancing plays a vital role in scaling of system which is very common in distributed systems.

The issues pertaining to the static and dynamic load balancing algorithms have been analysed [60] with respect to cloud environment. The load balancers rely either on session-switching at the application layer, packet-switching mode at the network layer or processor load balancing mode, takes the higher priority. The authors suggested a new integrates information from virtualized computer
environments and end user experience in order to be able to proactively manipulate load balancing decisions or reactively change decision in handling critical situations.

Kenthapadi and G S Manku [61] have discussed the “Availability Checker” that works as a client stub for Load Balancer. Load Reporters (LRs) updates the AC with all the load details from the various available servers which get stored in a hash table or database.

A parallel computer system [62] is a collection of processing nodes that communicate and cooperate with each other with tightly coupled software, to solve large computational problems efficiently. To achieve this, the large computational problem needs to be partitioned into several tasks with different workloads and then are allocated to the different processing elements for processing. They have discussed various dynamic load balancing techniques in brief which were exclusively used in parallel systems and concluded with the comparative performance analysis result of these algorithms.

Parveen Jain and Daya Gupta [64] have discussed that the dynamic load balancing have the potential of performing better than static strategies but they are certainly more complex than the static load balancers due to various overheads involved during the dynamic execution. They examined the Load balancing model and proved that the dynamic ones reduce drastically response times. But as the centralised systems suffer the problem of bottleneck and single point failure, they have suggested that the centralized node could be replaced with replication of systems. Their scheme reduces the average waiting time spent drastically.

Satish Penmatsa and Anthony T [65] Chronopoulos have reviewed two existing static load balancing schemes based on M/M/1 queues and they have proposed two dynamic load balancing schemes Dynamic Global Optimal Scheme
(DGOS) and Non Cooperative Scheme with Communication (DNCOOPC) for multi-user (multi-class) jobs in heterogeneous distributed systems based on the existing models Global Optimal Scheme (GOS) and DNCOOPC. The objective of one scheme is to minimize the likely expected response time of the overall system whereas the other tries to minimize the expected response time of the individual nodes of the system. Their simulation and the performance of the dynamic schemes with various loads and performance evaluation parameters such as system utilization, overhead for job transfer, bias and the exchange period of state information. Their results prove that, at low communication overheads, the dynamic schemes show better performance over the static schemes. But as the overheads increase, the dynamic schemes yield similar or even lower performance to that of the static schemes. They have also discussed the further research idea of dynamic load balancing model which not only balances the load from various machines which differ in their arrival time but also should differ in their execution time as well.

Ardhendu Mandal and Subhas Chandra Pal [66] have done the empirical study on the various dynamic load balancing techniques used in parallel computing systems namely the centralized model and optimal, heuristic and approximation models of distributed approach. Some of the main goals of a load balancing algorithm, as pointed out by the authors are Performance Improvement, Job Equality which is to treat all jobs in the system equally regardless of their origin, Fault Tolerance, Modifiability are the important factor discussed by them is that if the system requires good scalability, then the non-cooperative distributed dynamic load balancing techniques provide better solution as the overhead is limited when compared to the other techniques.

Parveen Jain and Daya Gupta [67] have proposed a model for dynamics of a distributed computing system load balancing. Firstly they have implemented the centralized model for balancing the load of the system. This Model considers
two approaches. The centralized node of the load balancer model is partitioned into number of small size nodes. The partitioned node need not be homogenous here. These smaller nodes are given extra load from primary nodes. In the second model the algorithm checks if all the smaller nodes have jobs to execute and identify the idle ones. Idle time of such nodes is utilized to process some extra load. The algorithm uses ISR (Interrupt Service Routine) for handling the interrupt. In ISR, Each supporting node maintains a priority queue, which has the priority number of each process. They have considered switching between the processes when there is an entry of higher priority process. As main aim in distributed system is to execute the process at minimum cost i.e. time is most important factor that can be considered in cost calculation. They have used the ant colony optimization technique to minimize the complexity. As time is the more highly valued attribute with respect to load balancing, here the authors have shown that their models work better as their algorithm tries to identify and eliminate the idle time of the secondary nodes.

Zhiling Lan [68] et al., have proposed a dynamic load balancing scheme for Structured Adaptive Mesh Refinement (SAMR) application on distributed systems. Their proposed scheme can deal with heterogeneity of processors, heterogeneity and dynamic load of the networks. They have divided the load-balancing processes in two phases: global load balancing and local load balancing. Heuristic method is followed to evaluate the computational gain and redistribution cost for global redistribution. Their Experiments show that by using the distributed Dynamic Load Balancer scheme, the execution time can be reduced by 9 to 46% as compared to using parallel Dynamic Load Balancer scheme which has the limitation of not supporting the heterogeneous and dynamic features of distributed systems.

Shailesh Saxena et al., [69] have presented the analytical comparison of dynamic load balancing algorithm with fuzzy logic. The fuzzy logic is based on
an intelligent fuzzy grouping approach. Their algorithm redirects the load effectively in less time from one overloaded node to less load node. It also proves to have short response times, high throughputs, and short turn-around times which yield to a better performance ratio.

Mayuri A. Mehta et al., [70] have suggested that selecting and configuring the appropriate. Information, transfer, selection and location polices were identified as the essential components. The authors have proposed a Modified Demand Driven Information Policy (MDDIP) and Limited Broadcast Location Policy (LBLP) which reduces the communication overhead. The overhead is reduced by limiting the number of nodes consulted to check their load status.

Sagar Dhakal et al., [71] have created a continuous-time stochastic model for the load balancing of distributed system. It considers the arrival of the random loads and the random delays. They have formulated an optimal solution for minimizing the overall completion time. This is reduced by one-shot load balancing model over the balancing gain and the balancing instant. They have proved that when the average transfer delay per task is larger compared to the average processing time per each single job they have shown that it would minimize the average completion time. When the average transfer delay per task is large compared to the average processing time per task, reduced load-balancing strength minimizes the average overall completion time. They have executed optimal one-shot load-balancing model that does the distributed dynamic load balancing in which, at every external load arrival, the receiver node automatically performs the load balancing.

The dynamic model is compatible with the changing traffic in the channel and the task processing rate from various application programs. Their model minimizes the average completion time per job and thus improves the system throughput. They have also improved the queuing delays and the transfer
delays, and have reduced the likelihood of nodes being idle as there are tasks in the system, comprising tasks in the queues as well as those in passage. This dynamic model may be adapted in wireless sensor network where the nodes are confined in computing power and power consumption at the same time.

Azzedine Boukerche et al., [72] have proposed a hierarchical dynamic load balancing scheme to redistribute the load for High Level Architecture (HLA) based simulations with a large scale distributed computing systems. This scheme uses grid services in the architecture of the system for scalability and to overcome are introduced in the scheme’s architecture to support the balancing system to scale accordingly, to prevail over the issues related to heterogeneity, and to take care of unreliable resources through data transfer for immigration. The balancing model with a tree structure, divides the balancing algorithm into three sequential phases that detect, redistribute, and to migrate the load, and also identifies the external background loads and load differences in heterogeneous resources. The system uses Grid services to recover monitoring information from shared resources and to support associate migration. The architecture of the system gives a solution that is scalable for the management of resources and load of large scale distributed systems. But the authors have failed to eliminate the communication overhead which is a hindrance for the system’s performance.

2.5 DISTRIBUTED BIG DATA ANALYTICS WITH HADOOP FRAMEWORK

Almost every domain is reliant on information that can be harnessed from Big Data to help move towards progress [73][74][75][76][77]. The researchers have claimed that Big Data can be typically characterized by the three V’s – Volume – Exponentially growing data, Variety – Data comes in all shapes, sizes and forms and Velocity – Rate of data creation and the rate at which data is
analyzed and harnessed for useful information. These papers propose an implementation of Big Data analytics using Hadoop and Hbase.

S Sathya [78] et al., have discussed the problems related to the heterogeneous database integration by providing the implementation of Virtual Database Technology (VDB) through Hadoop’s MapReduce. They have built a virtual database engine to get the high performance data integration for large volume of heterogeneous data. Zheyuan Liu [79] have presented a detailed study of the resource consumption profiles for MapReduce of Hadoop with respect to CPU and memory by altering the configuration parameters related to input output buffer. Yanfeng Zhang [80] has focused their research on computations which involve lots of iterations on massive data sets. They have evaluated the prioritization to converge the iterative results and through which they have achieved better speed over the Hadoop iterative algorithms.

Yongwei Wu [81] et al., have proposed a systematic and practical performance analysis framework, with respect to architecture and design models for defining the structure and behaviour of typical master/slave DFSs. They have evaluated the performance of DFA both qualitatively and quantitatively and the Hadoop Distributed File System (HDFS). The performance of MapReduce is checked in three different virtual machines and different number of virtual machines is done by Yang Yang [82] et al., and they have discussed that this performance test in turn would help the researchers to design adaptive scheduling algorithms through which better performance could be achieved with more virtual machines.

Ronald C Taylor [83] has presented an overview of Hadoop framework along with HDFS, HBase and MapReduce concepts and the applications which are being benefited in the field of bioinformatics. Skewed workload problem in MapReduce is addressed by Venkata Swamy Martha [84] et al., by hierarchical
MapReduce which splits the heavy tasks into children tasks and they are again given to MapReduce as a new job recursively till the job becomes fit to get executed. The detailed study on the merits and demerits of in-storage processing on current Solid-State Disk (SSD) architectures is presented by Yangwook Kang [85] et al., and the workflow manager developed with the name Nova and deployed at Yahoo and gets executed in Hadoop clustered framework. Two models named flat platform performance model and workflow performance model have been proposed and tested by Zhuoyao Zhang et al., which ensures the consistency, flexibility, efficacy and performance of MapReduce framework of Hadoop [86][87]. There are other performance enhancement models with workload sharing, enhanced configuration and remodelling has been proposed by researchers in various circumstances. [88][89][90][91][92].

Reviewing the above research works in distributed system, By finding the limitations of their work and by reading the future work of those researchers, we could channelize our the research problems clearly identified and we have attempted to improve the performance of distributed systems by modifying few algorithms and proposing new models in the load balancing models.