CHAPTER 8

SUMMARY AND CONCLUSIONS

8.1 SUMMARY

For last two decades the development in the Computing engineering is towards Distributed, cost effective and high capacity products. Furthermore software development activity has become more of decentralized in this manner increasing more development opportunities for the manufacturers. The performance of computing system has improved significantly with the help of multiprocessors and multi-computers. Distributed systems have many interesting research issues such as scalability, security, fault tolerance, manageability, and performance. The Performance analysis of distributed system is a great issue as it is expected that the distributed system should be of high speed, fault tolerant and cost efficient. Additionally the capability to estimate the potential performance of a large and complex distributed software system at design time could reduce the software cost and also the risk to a greater extent.

An improved election algorithm for the ring topology is presented in chapter three for the distributed system’s synchronization. This is based on the existing ring election algorithm. Election algorithms are basically for choosing the new coordinator process for a group of processes during the failure of the old coordinator process. As all distributed process need high degree of interaction between them, there should be a leader process to synchronize all of them and the processes should not be held waiting due to the failure of such a leader process. Existing ring election algorithm elects the leader process with O(n²) message complexity during
the worst case, wherein our proposed ring election algorithm is efficient with $O(n)$ messages.

In our algorithm, the duplicate election messages are destroyed and so the respective coordinator messages are also avoided. There is only one election message which goes for a complete circulation and only one round of coordinator message. This scenario is applicable for all the best, average and worst cases. N number of coordinator messages is circulated in worst case of existing Ring Algorithm and N-1 rounds of duplicate election messages were also present in the existing algorithm, but our algorithm destroys all of them as early as possible by the help of the clock time. These two factors give better performance in terms of time and messages.

Various election algorithms are practiced for different network topology. Bully election algorithm is very famous designed by Garcia-Molina for the topology where broadcasting is feasible. In chapter four, we have discussed the drawbacks of Bully algorithm and then we presented an optimized model of Bully algorithm called modified bully algorithm. Modified Bully algorithm is having all advantages of Bully algorithm. The additional advantages of modified Bully algorithm are that this algorithm is very simple; having fail-safe mechanism, no parallel election, and needs reduced number of messages compared to the original model. Our analytical simulation shows that our algorithm is more efficient rather than the Bully algorithm, in both number of message passing and the number of stages, and when only one process run the algorithm message passing complexity decreased from $O(n^2)$ to $O(n)$. In this analysis we consider the worst case in modified algorithm. Implementation result of this analysis clearly shows that modified algorithm is better than bully algorithm with fewer message passing and the fewer stages.
When the requirement for the resource utilization and the computing power increases, the load balancing issue becomes very important which is inevitable in distributed systems as the goal of distributed system itself is sharing the workload among the available servers effectively. The issue of load balancing in distributed system is an most significant and challenging area of research in computer engineering because the load balancing in distributed system has a key role in the overall system’s performance. Load balancing is the method of developing the performance of system through a redistribution of system load among less-weighted processors. Chapter five analyzed the various existing load balancing models namely the static and dynamic load balancing models. We have discussed the inherent characteristics, operational process, advantages and disadvantages of various static load-balancing techniques and dynamic load-balancing techniques. Also we made a comparative study on various parameters of the load-balancing techniques. Load balancing of servers is very crucial these days as the number of requests to be services at every moment is very huge. To increase the performance of the system, we may deploy a whole lot of servers but managing all these servers dynamically with balanced load is also essential as the performance is directly related to the load balancing method and algorithms. We have brought out the limitations of the existing simple LB model and proposed an alternative LB-ACLRs model. When simulated, our algorithm shows better result in comparison with the simple LB with respect to memory and CPU utilization.

In chapter six, we have presented our load balancing thinker model which is a collaboration of all existing load balancing algorithms. Each of these individual algorithms work well only during certain situations and so our thinker decides which algorithm to choose based in a particular situation and invokes it to get the best efficiency. And it switches through the
algorithms at every time interval if the situation changes. Our model also facilitates the inclusion of the newer load balancing algorithms which would be developed by the future researchers. We have used time-outs for the thinker for our implementation. But it is better if the thinker gets invoked whenever the situation changes instead of the fixed clock intervals. The centralized approach of thinker model might face the problems of bottleneck situation and the single-source of failure.

Chapter seven discusses the application structure for big data analytics that was executed with Hadoop distributed framework. Hadoop distributed framework gives flexibility to the programmer to choose the feature that he is comfortable to code with and it is also flexible and if the distributed system concepts go with hadoop then it would be a perfect blend that shoots up the system’s performance drastically. Since writing a lot many lines of code in Java would be too difficult to work on the relational database model and so we have chosen the Hive and the Pig models to work on the datasets and Pig has the advantage of creating MapReduce code for us. We have executed business logic on a 2GB dataset and could achieve a better performance when we have executed it in Hadoop and on a centralized system.

8.2 CONTRIBUTION OF THIS WORK

• A message efficient algorithm for the leader election algorithm is proposed for the ring topology in distributed systems which exterminates all the duplicate election messages and makes sure that only the original election message gets passed and thus drastically reduces the number of messages required to find the leader process.

• An improved version of existing Bully Election Algorithm for Distributed Systems for the mesh topology is proposed for a better
synchronization which is simple, fault tolerant and faster than the existing bully algorithm. This also reduces the number of simultaneous election messages. These two proposed election algorithms work better during the worst case which is very common in a dynamic distributed system and yield better performance during fault.

- **Situation Based Load Balancing Algorithm for Distributed Computing Systems** is proposed to enhance the load balancer which implements the best load balancing algorithm during a particular situation. The algorithm works with FCFS, Round Robin and optimized weight algorithms and among the three, one algorithm would be selected by the thinker to decide which is best for the time being based on the number of servers alive. This algorithm is flexible enough to give room for the future load balancing algorithms.

- **Another load balancer which works with Availability Checker and Load Reporters (LB-ACLRs)** is proposed for improving the performance of distributed systems which splits the monolithic load balancer into modules like the simple load balancer, availability checker and load reporters. Our load balancer works better as its responsibilities are reduced and it completely focuses on the balancing of the load alone wherein other modules which are spread across the system take care of the overheads.

- **An Approach for big data analytics using Hadoop Distributed File System** has been proposed to show that the open source is very much valuable to analyse huge volume of data in minimal time duration. The approach also suggests that the better performance can be felt if we apply the transformation and projection logic on the original data which filters only the needed attributes instead of analysing the entire big data.
8.3 FUTURE WORK

The concepts of election algorithms can be implemented in the distributed sensor networks where the energy saving is of greater importance. The groups of sensors need to elect their leader to transmit their sensed data to the base station with reduced energy wastage. The ring election algorithm may be implemented with the physical clock synchronization model so that real-time critical applications like aircraft control system, tsunami warning system etc., as the processes of these systems cannot waste much time to decide the next leader during the failure of leader process.

Situation based load balancer has been implemented with the three algorithms. Recent literature review of load balancing models should be studied and may be added to the load balancer to render the best performance with the best of load balancing algorithms.

The load balancer framework with the availability checker and the load reporter model has the centralized load balancer which might suffer the problem of bottle neck situation and single source of failure. Further it can be improved if we distribute the load balancer among the nodes. But synchronization should be addressed perfectly to make the system coherent.