5. Prediction based Dynamic Load Balancing Algorithm (PDLB)

In the chapter, we have proposed a prediction based DLBA (PDLB). The algorithm has two phases, first is prediction phase and second is load balancing phase. In the first phase algorithm try to predict cluster load status, and in the second phase, it tries to balance the load among the clusters using the predicted cluster load status data. The chapter includes pseudocode and detail discussion of the proposed algorithm. The proposed algorithm is simulated in the ALEA simulator using the real-time dataset, and cluster configuration explained in the previous chapter. The simulation result of proposed DLBA is compared with the priority scheduling algorithm at the end of the chapter.

5.1 Prediction based Dynamic Load Balancing Algorithm (PDLB)

The proposed PDLB takes load information of cluster for the defined period as an input. It maintains the load status counter for each cluster per day on an hourly basis. It learns from the observations and predicts the future load status of the cluster in a DS. The proposed DLBA selects the best suitable cluster for the execution of incoming jobs which balances load in the proposed architecture of DS.

The algorithm first allocates incoming jobs to the cluster having underloaded status for a particular duration in the past. It changes the future status of underloaded as well as overloaded clusters as medium loaded clusters. The algorithm enables the system to take actions by prediction and avoid the problem of unbalance load among the clusters of distributed systems.

5.1.1 Cluster Load Status Prediction Algorithm

Algorithm [1] show the steps of the cluster load status prediction algorithm. The algorithm takes CPU load information of the clusters as an input and gives the predicted status of the clusters as an output. The observation cycle is defined in step 3. We take 89 days as an observation cycle as the selected dataset has data of 89 days. Step
4, is executed for all the numbers of days in observation cycle and step 5 is executed for 24 hours of each day.

The steps 6, 7, and 8 calculates system threshold (ST) value and cluster threshold (CT) value respectively by using equations 3.1 and 3.2 presented in section 3.4.1. The steps 9-14 find the load status of each cluster and updates the size of the underloaded and overloaded counter for a particular hour. The rule has been defined for checking the status of the cluster on an hourly basis. The cluster considered as an overloaded for a particular hour if the CT value of cluster is high compared to ST value and vice-versa. Step 15-20 sets the status of the cluster for a particular day by below-defined rules. At last the load status of clusters is updated for the next execution cycle in step 21.

**Algorithm 1** Cluster Load Status Prediction

1. **Input:** load information of clusters
2. **Output:** predicted status of clusters
3. **define observation cycle**
4. **for each day in observation cycle**
   5. **for each hour in day**
      6. calculate system threshold value $ST$
      7. **for** $j = 1$ to $m$
         8. calculate cluster threshold value $CT_j$
         9. **if** $CT_j > ST$ **then**
            10. $Status_j = Overloaded$
            11. $Overloaded_j++$
         **else**
            12. $Status_j = Underloaded$
            13. $Underloaded_j++$
      14. **if** $Overloaded_j > t_{upper}$ **then**
         15. $DayStatus_j = overloaded$
      **else if** $Underloaded_j > t_{upper}$ **then**
         16. $DayStatus_j = underloaded$
      **else**
         17. $DayStatus_j = mediumloaded$
      18. **update load status of cluster for next execution cycle**

The rule is defined by considering the threshold algorithm explained in section 2.5.4. Here, we have used the predefined double threshold values $t_{upper}$ and $t_{under}$ as a threshold policy. The selected threshold values are $t_{under} = 8$ hours and $t_{upper} = 16$ hours in a day. The selected value of $t_{under}$ is the one-third part of the total
hours in a day, and the $t_{\text{upper}}$ is the two-third part of the same. The one-third part and the two-third part of the total hours in a day is selected as threshold values because we have decided to categorize the cluster status in three different states like underloaded, mediumloaded, and overloaded.

The following rules are defined to set the day wise status of the cluster based on the selected threshold policy.

**Overloaded** if cluster remains overloaded for more than $t_{\text{upper}}$ value in a day

**Underloaded** if cluster remains underloaded for more than $t_{\text{upper}}$ value in a day

**Mediumloaded** if cluster remains overloaded for more than $t_{\text{under}}$ value and underloaded for less than $t_{\text{upper}}$ value in a day

![Cluster Load Status on Hourly Basis](image)

Figure 5.1: Cluster Load Status on Hourly Basis

The figure 5.1 show the screenshot of a data file created after the execution of proposed cluster load status prediction algorithm. The first column represents the hours, and second to ninth columns represents the load status of first to eighth clusters for that particular hour. The "0" value in the second to ninth columns indicates
the particular cluster is underloaded and "1" value indicates the particular cluster is overloaded in that hour. Furthermore, the second last column represents the total numbers of underloaded clusters, and the last column represents the total numbers of overloaded clusters in the particular hour. The data files have total 2136 rows representing the load status of all eight clusters for each hour in 89 days of execution cycle. The data file is used to predicts the day-wise cluster load status.

5.1.2 Dynamic Load Balancing Algorithm

**Algorithm 2** Dynamic Load Balancing

1: *Input:* predicted load status of clusters, and incoming jobs
2: *Output:* schedule incoming jobs to best suitable cluster for an execution
3: define execution cycle
4: **for** each day in execution cycle **do**
5: read predicted load status value of clusters
6: prepare UL, ML, and OL list
7: **for** all incoming job in a day **do**
8: if sizeofUL > 0 then
9: randomly select one cluster from UL list
10: submit job to the selected cluster
11: **else if** sizeofML > 0 then
12: randomly select one cluster from ML list
13: submit job to the selected cluster
14: **else**
15: randomly select one cluster from OL list
16: submit job to the selected cluster
17: update load status of cluster

The pseudo code of the proposed DLBA is shown in the algorithm 2. The algorithm takes the output of the proposed cluster load status prediction algorithm discussed previously as an input and gives the best suitable cluster for executing the incoming job as an output. The execution cycle is defined in step 3, which is the same as the observation cycle. It is required because the performance of this algorithm depends on the output of the cluster load status prediction algorithm. Step 4, is executed for all the numbers of days in the execution cycle. In step 5, an algorithm read the predicted load status for all clusters, and it prepares the lists of underloaded, mediumloaded, and overloaded clusters in step 6. Step 7 one by one selects each incoming jobs for the execution.
In step 8, the algorithm checks for the underloaded clusters and if it is there then it submits the incoming job to anyone randomly selected clusters from the list in steps 9-10. The steps 11-13 executes only if not a single cluster is available in UL list. Step 11 checks for the medium loaded clusters and it submits the incoming job to anyone randomly selected cluster from the list in steps 12-13. At last, the algorithm submits an incoming job to one of the overloaded clusters as it is an extreme situation where all the clusters in the system are overloaded in steps 14-16. The algorithm updates the cluster load status in last step 16 which is useful for the next execution cycle.

5.2 Result Analysis of Priority Scheduling and PDLB Algorithms

The performance of the proposed PDLB is evaluated using the simulation setup described in chapter 4 and compared against the priority scheduling (PS) algorithm. We have select PS algorithm for the comparison because the selected dataset having prioritized jobs. The dataset has three different queues for categorized incoming jobs as per the priority which is mentioned in section 4.4.1. We have considered the percentage of cluster usage, average cluster utilization, and imbalance load metric as evaluation parameters.

5.2.1 Result Analysis regarding Cluster Utilization

Table 5.1 show the utilization of each cluster and average cluster utilization. The utilization of the first cluster is 70.67% which is highest while utilization of the fifth cluster is only 8.58% which is lowest in priority scheduling. The much difference between highest and lowest indicates an imbalance of load among clusters. Even the utilization of second to sixth clusters is very much closed to the lowest side in the existing approach. It shows that balancing is required among the clusters. The problem of imbalance load among clusters resolved up to some extent in the prediction approach.

The results show that the ACU of available resources is increased by 6% in PDLB algorithm compared to the PS algorithm. The utilization of individual cluster is im-
proved in six clusters namely gaia-[61-62], gaia-[63-72], gaia-73, gaia-74, gaia-[75-79], and gaia-[120-151] out of eight clusters in the proposed approach. The PDLB increases the utilization of second to sixth total five clusters by 10% which leads the ACU of the system by 6%. In the existing approach, the utilization of the first cluster which has the highest amount of processing power is 70.67%, it indicates most of the incoming jobs are executed on the first cluster. In PDLB algorithm the utilization of the first cluster reduces by 6% means utilization of other clusters increased in the system. The range of cluster utilization is from 8.58% to 70.67% in PS which reduces to 19.05% to 63.86% in PDLB, which shows that the level of load balancing increases in PDLB algorithm compared to PS algorithm.

### 5.2.2 Result Analysis regarding Day wise Cluster Utilization

Figures 5.2 and 5.3 show the cluster usage on a day-to-day basis of PS and PDLB algorithms respectively in graph format. The X-axis represents numbers of days, and the Y-axis represents clusters in descending order of CPUs. The rows separated by thin white lines in the figures represent the load status of the individual cluster. There are total eight rows display in the figures because the selected data-set has eight clusters. The green, yellow, and red colors representing under, medium, and overutilization of clusters respectively. In figures, rows are colored for eighty ninety days because the selected data-set log is having data of eighty ninety days from 22nd May 2014 to 19th August 2014 [46].
Figures 5.2 and 5.3 show total eight rows which represent individual clusters as the selected dataset having eight clusters.

Figure 5.2: Cluster Utilization per day in PS Algorithm

Figure 5.3: Cluster Utilization per day in PDLB Algorithm

Figure 5.2 shows that the first cluster remains busy for most of the days. The utilization of gaia-[1-60], gaia-[80-119], and gaia-[120-151] clusters is high compared to other clusters. Figure 5.3 shows that PDLB can predict the loaded situation of gaia-[1-60], gaia-[80-119], and gaia-[120-151] clusters and it schedule incoming jobs to other clusters which increases the overall utilization of available resources. Figure 5.3 shows that for the first 35 days there is an improvement regarding load balancing in the prediction approach, but after that, the PDLB performs same like PS algorithm.
5. Prediction based Dynamic Load Balancing Algorithm (PDLB)

It is a limitation of PDLB algorithm that it fails to predict the cluster load status after 35 days and schedule incoming jobs as per the priority.

5.2.3 Result Analysis regarding Load Imbalance Metric

Figure 5.4: Load Imbalance Metric of PS and PDLB Algorithms

Figure 5.4 shows the load imbalance metric of PS and PDLB algorithms in graph format. The X-axis of the graph represents the load imbalance metric, and the Y-axis represents the clusters. The eight columns in the graph represent the eight clusters of the Gaia cluster configuration. The points plotted over the X-axis indicates the overutilization of the respective cluster. For example, in the first column, both the points representing PS and PDLB algorithms plotted above X-axis indicates the first cluster over-utilized in both the algorithms. The points plotted below the X-axis indicates underutilization of the respective cluster. From the figure 5.4 it is apparent that GAIA-[61-62] to GAIA-[75-79] clusters are under-utilized in both the algorithms.

The ideal situation or the expected output in this graph is when all the points are plotted on the X-axis as explained in section 3.4.3. Hence, the PDLB performs better compared to PS as the line connecting all the plotted points for PDLB is closer to X-axis compared to the line connecting all the plotted points for PS algorithm.
5. Prediction based Dynamic Load Balancing Algorithm (PDLB)

It indicates the level of imbalance load of all the eight clusters is reduced in PDLB algorithm. However, the points are not plotted on X-axis even in PDLB which indicates there the scope of improvement regarding imbalance load.

5.3 Conclusion

The objective is accomplished as the prediction based DLBA is proposed for the DS. The algorithm predicts the future load status of the cluster by past observation. It dynamically changes the predicted status after every execution cycle. The predicted information plays a vital role in the selection of the best destination cluster for incoming jobs. The traditional priority-based SA and PDLB algorithm are simulated in ALEA simulator using a real-time dataset and cluster configuration. The cluster utilization, load imbalance metric, and average cluster utilization are calculated as an evaluation parameter.

The simulation results show that the proposed PDLB algorithm improves the ACU and decreases the LIM compared to PS algorithm. The PDLB algorithm performs well in the initial phase of schedule but after that, it performs the same like PS algorithm, so there is a scope for improvement in the PDLB algorithm. The performance of PDLB is not stable as it depends open the predicted values. Further, it is not considering the current load status of the cluster. As a future work, the proposed algorithm can be modified which schedule incoming jobs by looking predicted status as well as current status of the clusters.