6. Pluggable to Scheduler Dynamic Load Balancing Algorithm (P2S_DLBA)

The imbalance load is one of the significant issues of the DS which leads to the worst utilization of available resources [49]. In the previous chapter we have tried to resolve the problem of imbalance load by proposing prediction based DLBA (PDLB), the algorithm reduced the level of imbalance load but not up to the mark which indicates prediction approach is not the optimal solution of the problem of imbalance load in a distributed computing environment. Hence it is the prime requirement to design a DLBA which resolve said problem up to a satisfactory level.

Ordinarily, DLBAs start balancing activities after the system becomes imbalanced. The DLBAs balancing load among clusters by redistributing or transferring load from overloaded cluster to underloaded cluster by doing process migration [18][50]. It is more appropriate to design DLBA which is pre-active and start the balancing activity before the system becomes imbalanced. We have proposed pluggable to scheduler DLBA (P2S_DLBA) which schedule incoming jobs by considering the current load status of the clusters.

The designed DLBA can be plug to any scheduling algorithms which start the balancing act in parallel with scheduling activity. It helps scheduling algorithm by suggesting cluster with the highest number of free resources at the time of scheduling. Therefore the chances of the system going into imbalance state are reduces, and the system remains in a balanced state. The best part of the P2S_DLBA algorithm is it balances the load without migrating a single process from one cluster to another cluster. Hence, the algorithm doing load balancing without increasing extra system overhead due to the process migration.

In this chapter, the first section includes the pseudo code of P2S_DLBA algorithm and the detail explanation about the working of it. The second section presents comparative result analysis of P2S_DLBA with PDLB and Priority Scheduling algorithms concerning three different evaluation parameters. The conclusion is stated at the end of the chapter on the basis of comparative result analysis.
6.1 Pluggable to Scheduler Dynamic Load Balancing Algorithm (P2S_DLB)

The algorithm takes the current status of the clusters resources and incoming jobs as inputs and schedules the incoming jobs to the best suitable cluster for execution as an output. The prime focus of the proposed algorithm is to target the clusters having a higher number of free resources for incoming jobs execution. Hence, the first condition of the best suitable cluster here is the cluster with the highest number of free resources at the time of scheduling. Moreover, the second condition is the number of running resources out of free resources should be more than the required resources.

It is multi-level DLBA where the multi-level load balancing threshold policy decides the levels of load balancing. In the first level, it tries to find a highly under-loaded cluster, in the second level it tries to focus on under-loaded clusters. Furthermore, algorithm targets medium-loaded and over-loaded clusters in third and fourth levels respectively. At last, the algorithm targets highly over-loaded clusters as it is an extreme situation where all the clusters are over-loaded. The pseudo code of proposed pluggable to scheduler DLBA is shown in Algorithm 3.

The proposed algorithm has a total of four stages, in the first stage, it tries to achieve more than 50% utilization of available resources in every cluster. Furthermore, the algorithm tries to achieve 66%, 75%, and 80% utilization of every cluster in second, third, and fourth stages respectively. The steps 4-9 written in the algorithm is for the first stage. Step 5 find out cluster having more than 50% free resources and the step 6 check the running resources into 50% free resources is higher than the requirement. Steps 5-6 are checking the two required conditions of the best suitable cluster. Step 7 schedule an incoming job to find out under-loaded cluster for execution. The status of the selected cluster updates in step 8. Once the selected job is scheduled, the algorithm immediately comes out of the stage and start from the beginning by selecting the next job.

The group of steps 9-15, 16-22, 23-29 written in the algorithm are for second, third, and fourth stages correspondingly. The working of an algorithm remains the same in every stage as steps are similar, but they are checking different threshold values of free resources in the clusters. At last, when the two required conditions are
not fulfilled in all previous four stages, steps 30-33 schedules job to the cluster by only checking the second required condition. It is like default case in switch statement of C programming language. Here, in this algorithm control go to steps 30-33 only when every cluster utilization reached to more than 80%.

Algorithm 3 Pluggable to Scheduler Dynamic Load Balancing Algorithm (P2S_DLB)

1:  \textit{Input:} incoming job and current resource status of the cluster
2:  \textit{Output:} schedule job to cluster for execution
3:  \textbf{for} \(i = 1\) to \(n\) \textbf{do}
4:  \hspace{1em} \textbf{for} \(j = 1\) to \(m\) \textbf{do}
5:  \hspace{2em} \textbf{if} \(\text{Num Free PE}_j > \text{Num PE}_j / 2\) then
6:  \hspace{3em} \textbf{if} \(\text{Num Running PE}_j \geq \text{Num Required PE}_i\) then
7:  \hspace{4em} Schedule \(i^{th}\) job to \(j^{th}\) cluster for execution
8:  \hspace{4em} update load status of \(C_j\)
9:  \hspace{4em} exit from the loop and select next job
10: \hspace{1em} \textbf{for} \(j = 1\) to \(m\) \textbf{do}
11:  \hspace{2em} \textbf{if} \(\text{Num Free PE}_j > \text{Num PE}_j / 3\) then
12:  \hspace{3em} \textbf{if} \(\text{Num Running PE}_j \geq \text{Num Required PE}_i\) then
13:  \hspace{4em} Schedule \(i^{th}\) job to \(j^{th}\) cluster for execution
14:  \hspace{4em} update load status of \(C_j\)
15:  \hspace{4em} exit from the loop and select next job
16: \hspace{1em} \textbf{for} \(j = 1\) to \(m\) \textbf{do}
17:  \hspace{2em} \textbf{if} \(\text{Num Free PE}_j > \text{Num PE}_j / 4\) then
18:  \hspace{3em} \textbf{if} \(\text{Num Running PE}_j \geq \text{Num Required PE}_i\) then
19:  \hspace{4em} Schedule \(i^{th}\) job to \(j^{th}\) cluster for execution
20:  \hspace{4em} update load status of \(C_j\)
21:  \hspace{4em} exit from the loop and select next job
22: \hspace{1em} \textbf{for} \(j = 1\) to \(m\) \textbf{do}
23:  \hspace{2em} \textbf{if} \(\text{Num Free PE}_j > \text{Num PE}_j / 5\) then
24:  \hspace{3em} \textbf{if} \(\text{Num Running PE}_j \geq \text{Num Required PE}_i\) then
25:  \hspace{4em} Schedule \(i^{th}\) job to \(j^{th}\) cluster for execution
26:  \hspace{4em} update load status of \(C_j\)
27:  \hspace{4em} exit from the loop and select next job
28: \hspace{1em} \textbf{for} \(j = 1\) to \(m\) \textbf{do}
29:  \hspace{2em} \textbf{if} \(\text{Num Running PE}_j \geq \text{Num Required PE}_i\) then
30:  \hspace{3em} Schedule \(i^{th}\) job to \(j^{th}\) cluster for execution
31:  \hspace{3em} update load status of \(C_j\)

In P2S_DLB the incoming jobs are scheduled in such a way that indirectly it balances the load among clusters. The algorithm achieves better resource utilization and increases the level of load balancing step by step. The vital point to note here is the algorithm goes to the next stage only when resource utilization of every cluster is more than the set threshold value. It means after first stage not a single cluster is
available in the system with highly under-loaded status.

The P2S_DLBB is dynamic algorithm as it takes a run-time decision as per the set threshold value. The algorithm is flexible regarding the threshold value; the user can change the threshold value as per the incoming job requirements. Even the new stage can be added to reduces the gap between load balancing levels. In this section the working of the P2S_DLBB is described, now next is the result of the algorithm. The next section presents the simulation results and the comparative result analysis of the proposed algorithm with the existing algorithms.

6.2 Result Analysis of PS, PDLB, and P2S_DLBB Algorithms

The performance of the P2S_DLBB algorithm is evaluated and compared to the Priority Scheduling and our previously proposed PDLB algorithms. The detail explanation of PDLB is given in chapter 5. We have simulated the algorithms on the real-time data-set ("University of Luxemburg Gaia Cluster Log") in the ALEA simulator. The detail description about selected data-set and the ALEA simulation tool is stated in chapter 4. The cluster utilization, average cluster utilization, and the load imbalance metric are considered as performance evaluation parameters. The equations and impact of parameters are discussed in chapter 3. The objective of our research work is to improve the cluster utilization and to reduce the load imbalance metric.

6.2.1 Result Analysis regarding Cluster Utilization

Table 6.1 shows the cluster utilization and ACU of clusters for all three algorithms. The cluster utilization is improved for all eight clusters in the result of P2S_DLBB. The improvement in every cluster’s performance is due to the threshold policies defined in the algorithm which targets the under-utilized clusters. The plus point to be noted here is the even improvement in the every cluster utilization.

Furthermore, the utilization of the first cluster which has the highest amount of processing units is reduced to 51.13% in P2S_DLBB from 63.86% in PDLB and 70.67% in PS algorithm. It indicates that P2S_DLBB overcome the problem of giving higher preference to the cluster having higher numbers of resources in PS and PDLB
algorithms. The utilization of other clusters is increased in P2S_DLB as the utilization of the first cluster is reduced. Therefore, the average utilization is increased by 15% compared to PS and 9% compared to PDLB in the P2S_DLB algorithm.

Table 6.1: Cluster Utilization of PS, PDLB, and P2S_DLB Algorithms

<table>
<thead>
<tr>
<th>Cluster Id</th>
<th>Cluster Name</th>
<th>PS</th>
<th>PDLB</th>
<th>P2S_DLB</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>gaia-[1-60]</td>
<td>70.67%</td>
<td>63.86%</td>
<td>51.13%</td>
</tr>
<tr>
<td>2</td>
<td>gaia-[61-62]</td>
<td>10.01%</td>
<td>19.71%</td>
<td>38.04%</td>
</tr>
<tr>
<td>3</td>
<td>gaia-[63-72]</td>
<td>13.91%</td>
<td>24.63%</td>
<td>39.38%</td>
</tr>
<tr>
<td>4</td>
<td>gaia-73</td>
<td>16.28%</td>
<td>27.38%</td>
<td>37.54%</td>
</tr>
<tr>
<td>5</td>
<td>gaia-74</td>
<td>8.58%</td>
<td>19.05%</td>
<td>38.54%</td>
</tr>
<tr>
<td>6</td>
<td>gaia-[75-79]</td>
<td>10.05%</td>
<td>21.42%</td>
<td>35.37%</td>
</tr>
<tr>
<td>7</td>
<td>gaia-[80-119]</td>
<td>42.68%</td>
<td>41.30%</td>
<td>45.02%</td>
</tr>
<tr>
<td>8</td>
<td>gaia-[120-151]</td>
<td>33.19%</td>
<td>35.76%</td>
<td>40.25%</td>
</tr>
</tbody>
</table>

| Average Cluster Utilization             | 25.67% | 31.64% | 40.66% |

The utilization of the first cluster is 70.67% in PS and 63.86% in PDLB, on the other hand, the fifth cluster utilizes only 8.58% in PS and 19.05% in PDLB which show imbalanced load and uneven resource utilization. Also, the utilization of second to sixth clusters is very less in the existing approach (PS). It shows that balancing is required among the clusters. The problem is resolved in prediction approach as the PDLB improves utilization of second to sixth clusters by approximately 10% which increases the average utilization by 6% in PDLB. The P2S_DLB algorithm further increases the utilization of second and fifth clusters by approximately 15%, third and sixth clusters by approximately 14%, and fourth cluster by 10% which improves the average utilization by 9% in P2S_DLB compared to PDLB.

The less utilization of second and fifth clusters in PS, which has least processors show that PS under-utilized the clusters having less processing units. The PDLB improved the utilization of second and fifth clusters but compared to the other clusters second cluster remain at second last positions and fifth cluster at last position regarding cluster utilization. The P2S_DLB overcomes this problem by escalating the utilization of the second cluster to 38.04% and fifth cluster to 38.54% which are close to average utilization and higher than the utilization of other clusters. It shows that the performance of P2S_DLB is independent of the numbers of resources available to the cluster.
6. Pluggable to Scheduler Dynamic Load Balancing Algorithm (P2S_DLB)

Therefore, we have concluded that the P2S_DLB is a best-suited algorithm for heterogeneous clusters. It was one of our objectives stated in chapter 1, to proposed the DLBA which can work in a heterogeneous environment and objective is contended successfully. The starting and ending range of cluster utilization values are 8.58% and 70.67% in PS, 19.05% and 63.86% in PDLB, which is reduced to 35.37% and 51.13% in P2S_DLB. The optimized range of cluster utilization indicates that the level of load balancing is improved a lot in P2S_DLB.

6.2.2 Result Analysis regarding Day wise Cluster Utilization

In previous subsection result analysis regarding cluster utilization is presented, in which the statistics of overall utilization of cluster and the average utilization in percentage are explained. Here, in this subsection, each day wise cluster utilization is presented in graphical format. Figures 6.1, 6.2, and 6.3 show the individual cluster usage on a day-to-day basis in PS, PDLB, and P2S_DLB algorithms respectively. In figures, the X-axis represents days and Y-axis represents the 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. In figures, rows are colored for eighty ninety days because the selected data-set log is having data of eighty ninety days from 22\textsuperscript{nd} May 2014 to 19\textsuperscript{th} August 2014 as mentioned in section 4.4.1.

The green, yellow, and red colors represent under, medium, and over-utilization of clusters. As indicated in the right-hand side of the figures, the green, yellow, and red colors represent 0%, 50%, and 100% cluster utilization, while the mixture of green and yellow colors represents 25% cluster utilization. Likewise, the mixture of yellow and red colors represents 75% cluster utilization. All the rows are colored yellow for most of the days is the expected results with respects to the day-wise cluster utilization parameter. The expected result is concerning to the selected data-set (“University of Luxemburg Gaia Cluster Log”), it may change for other data-set because it depends on the total number of incoming jobs and the total numbers of the computing resources available in the system.

Figure 6.1 shows the performance of priority scheduling algorithm, in which the
first row is colored red for most of the days, which indicates that the first cluster (gaia-[1-60]) remains busy for almost all days. Likewise, the second row is colored red for half of the days which indicates second cluster (gaia-[80-119]) remains busy for half of the days. While the remaining rows are colored green for most of the days; it means leftover clusters (gaia-[120-151], gaia-73, gaia-[63-72], gaia-[75-79], gaia-74, and gaia-[61-62]) remains almost free for most of the days. The reason behind the over-utilization of the first cluster is a number of resources, gaia-[1-60] have total 720 CPUs as explicitly mentioned in the figure, which is highest in all clusters. Because of the large amount of CPUs, gaia-[1-60] is executed most of the jobs, same way gaia-[80-119] having second highest 480 CPUs over-utilized for half of the day when all the CPUs of the first cluster was busy.

The first two clusters have a total 1200 CPUs which is more than the total CPUs with other clusters which are 804 CPUs. It is the reason behind the under-utilization of the third to eight clusters because they hardly get a chance to execute the job as most of the job executed by first or second clusters. It shows the PS algorithm is not balancing the load among clusters; it is but oblivious in case of PS algorithm because it is a scheduling algorithm. Hence, it is simply scheduled the incoming job to the first cluster without considering the load status of the cluster; it schedules a job to second and other consecutive clusters only if the previous cluster does not have free resources.

Figure 6.2 shows the performance of PDLB algorithm. In the figure, all eight rows
colored with red, yellow, and mixture of red and yellow colors for the period of twenty
days, from the fifth day to the twenty-fifth day. It indicates PDLB is able to predict
the over-loaded situation of first and second clusters and scheduled incoming jobs
to under-loaded clusters which increased the overall resource utilization. The PDLB
predicts the status of clusters by past simulation results. We have simulated the PS
algorithm first, and then we have simulated PDLB algorithm on the same data-set.
The PDLB algorithm considers the cluster’s day-wise load status of PS simulation. It
is clear from the figures 6.1 and 6.2 that the PDLB algorithm balances the load among
the cluster. However, the PDLB is balanced the load better compare to PS for first
thirty-five days only, after that it performers same as the PS for remaining fifty-five
days, which indicates the scope of improvement in the PDLB.

The point to be noted here is the amount of incoming jobs, which is highest for
the first thirty-five days. PDLB is able to predict the load status of clusters in the
first thirty-five days, but in remaining days it is not able to predict the load status of
the clusters. Even in the performance of PDLB with other data-set ("MetaCentrum
Cluster Log") present in chapter 9, the algorithm is not able to predict the load status
for entire 200 days of simulation. The algorithm is not able to predict the load status
for 40 days. We have concluded from the multiple results that PDLB is giving better
result compare to PS algorithm, but it is not giving a consistent result.

Figure 6.3 shows the performance of the P2S_DLB algorithm. In the figure,
Figure 6.3: Cluster Utilization per day in P2S_DLB Algorithm

all eight rows are colored yellow for many days which indicates all the eight clusters are utilized, and clusters remain medium-loaded for most of the days. Moreover, the number of days when rows are colored green is less which shows the clusters remains under-utilized for fewer numbers of days. The clusters remain under-utilizes or unutilized only when there are less or no incoming jobs, for example, the first four days all the row are colored green in all three figures because there are no incoming jobs. Same way, in ten days from fifty-five to sixty-five clusters are under-utilized because the incoming jobs are less.

The decisive point to note from the result of P2S_DLB is the utilization of third to eight clusters. The third to eight clusters are colored green for most of the days in PS and PDLB, but in the case of P2S_DLB, they are colored yellow for most of the days. It indicates the utilization of third to eight clusters are increased. The most important point to consider from the figure is very fewer numbers of days in which rows are colored red, it means clusters hardly go into an overloaded condition. The figure shows that the proposed P2S_DLB algorithm performed best regarding load balancing and optimal resource utilization by equality divides the load among clusters.

### 6.2.3 Result Analysis regarding Load Imbalance Metric

In the previous two subsections result analysis with respects to cluster utilization is presented. This subsection represents a result analysis regarding load imbalance metric.
As explained in chapter 3, load imbalance metrics characterize how unevenly work is distributed among clusters. Figure 6.4 shows the percent of load imbalance metric between PS, PDLB, and P2S_DLB algorithms in graph format. The X-axis of the graph represents the percent of load imbalance metric, and the Y-axis represents clusters. The total of eight columns in the graph represent the total eight clusters in selected Gaia cluster configuration. The Gaia cluster configuration is explained in section 4.5.1.

It presents the difference between the average utilization and cluster utilization. The points plotted over the X-axis indicates the respective clusters are utilized more compared to average cluster utilization. For example, in the first column, all three points representing PS, PDLB, and P2S_DLB algorithms are plotted above the X-axis, which indicates the first cluster is utilized more compared to average utilization in all three algorithms. Same way the points plotted below the X-axis indicates the respective clusters utilized less than the average cluster utilization. From the figure, it is clear that 2nd to 6th clusters are utilized less than the average cluster utilization in all three algorithms.

![Image: Load Imbalance Metric of PS, PDLB, and P2S_DLB Algorithms]

The idea situation or the expected output in this graph is when all points are plotted on the X-axis as explained in section 3.4.3. Hence, the PDLB performs better
compare to PS as the blue line connecting all the plotted points of PDLB is nearer to the X-axis compare to the red line connecting all the plotted points of PS algorithm. It shows the load imbalance level is reduced in PDLB compare to PS but, it is not up to the mark as all the plotted points are not on the X-axis. The problem is resolved in P2S_DLB as the green line connecting all the plotted points of P2S_DLB is close to X-axis, which indicates the load imbalance level reduced in P2S_DLB compare to PS and PDLB algorithms.

The seven plotted points (2\textsuperscript{nd} to 8\textsuperscript{th}) out of eight points for the P2S_DLB algorithm is almost on X-axis, which specifies the level of load imbalance is very less in seven clusters. In, the remaining plotted point (1\textsuperscript{st}) the position of the point representing P2S_DLB is improved a lot compare to PDLB and PS algorithms. Here, the objective is to reduce load imbalance level as much as possible. The minimum load imbalance level is achieved in the result of P2S_DLB, hence the objective is fulfilled in the proposed algorithm.

### 6.3 Conclusion

An idea of balancing the load among resources at the time of scheduling is suggested in the chapter. The DLBA named as P2S_DLB is proposed which can plug to any scheduling algorithm in the DS. The algorithm finds clusters with higher numbers of free resources and schedules incoming jobs to that clusters. It steps by step increases the average cluster utilization in the system. The P2S_DLB algorithm is plugged into traditional priority-based scheduling algorithm. The Priority scheduling and P2S_DLB algorithms are simulated on the real-time dataset and real-time cluster configuration using ALEA simulator.

The results show that P2S_DLB gives best performance compare to PS and PDLB algorithms. The proposed algorithm increases the average cluster utilization and decreases the load imbalanced metric of the clusters. It is our objective to design DLBA which can work with scheduling algorithm is accomplished in this chapter. Furthermore, the additional objective is to make designed DLBA generalized so that it can work with any scheduling algorithms.

Therefore, as a future scope, the designed algorithm can be plugged into other
well-known scheduling algorithms for checking stability. The work has been carried out and presented in the next chapter. Moreover, the combination of PDLB and P2S_DLB algorithms can be proposed which may further increase average cluster utilization and decreases load imbalanced metric in the system. The algorithm is proposed and the comparative result analysis is done, which is presented in chapter 8.