AN IMPROVED LEAST-LAXITY-FIRST SCHEDULING ALGORITHM FOR REAL-TIME TASKS

H.S Behera, Satyajit Khuntia & Soumyashree Nayak,

Department of Computer Science & Engineering
Veer Surendra Sai University of Technology (VSSUT), Burla, Odisha, India

email : hsbehera_india@yahoo.com, satyajitkhuntia@yahoo.com, nayaksoumyashree786@gmail.com

Abstract:
Scheduling algorithms play an important role in design of real time systems. Least Laxity First (LLF) is a well-known and extensively applied dynamic Scheduling algorithm which has been proved to be optimal on uniprocessor systems. The Least-Laxity-First (LLF) Scheduling algorithm assigns priority based on the slack time of a process. Its most common use is in embedded systems, especially those with multiple processors. The algorithm is impractical to implement because laxity tie results in the frequent context switches among the tasks. The least switch and laxity first scheduling algorithm is proposed in the paper, which improves the least laxity first algorithm for periodic task by searching out an appropriate common divisor along with Modified Least Laxity First (MLLF). In this paper, we present an Improved Least-Laxity-First Algorithm (ILLF) which eliminates the disadvantages of MLLF Scheduling algorithm with intelligence time slice and prove its optimally

Keywords: Processes; laxity value; scheduling algorithm; real-time system; greatest common divisor

1. Introduction

LLF (Least Laxity First) is one of the classic dynamic priority scheduling algorithms in which the scheduling priority of a task is determined based on its precise slack time. Operating System overhead reduces the performance of a computer system especially for real-time system where task must complete prior to a given deadline. Among the Real-Time Scheduling algorithms, the algorithm based on the priority of tasks is very common which is a kind of dynamic-priority scheduling algorithm and has widespread application in the above real-time systems.

Real-time system is required to complete its work and deliver its services on the basis of time. The results of real time systems are judged based on the time at which the results are produced in addition to the logical result of computations.

With the LLF scheduling algorithm, however, if two or more tasks have same laxities, laxity-tie occurs. Once laxity-tie occurs, context switches takes place every scheduling point until the tie breaks. The laxity-tie in the LLF scheduling algorithm results in the poor system performance due to the frequent context switches.

In this paper, we propose an Improved Least Laxity First Scheduling Algorithm with intelligence time slice to solve the frequent context switches problem of the LLF Scheduling algorithm. This is an optimal approach to solve the no of context switching. This Improved-Least-Laxity-First Scheduling algorithm , however is better than that of the LLF Scheduling algorithm as well as MLLF(Modified-Least-Laxity-First) Scheduling algorithm.

2. Related Works

concepts of the least slack time scheduling algo for real time system. In [8-9], Abbott R. and Dertouzos ML, et al. suggested the Least Slack First algorithm, which assigns a priority to a task according to its urgent degree for being executed. The less the idle time of a task is, the sooner it will be executed. So, the most urgent tasks (the tasks that may not be the earliest deadline) will have the highest priority to be executed. Along with the system action, the idle time of the task is unaltered in executing state, but the idle time of the waiting task is strictly decreasing. The exigent degree of the waiting tasks will become higher when time elapses. So in the action period of the system, the task which waits long has the opportunity to get the resource of the processor at a moment. When the urgent of the tasks is close, it will author described a periodic task scheduling algorithm lead to serious frequent switches or thrashing phenomenon among the tasks.

3. Proposed Approach

Our proposed Improved Least Laxity First Scheduling Algorithm with intelligence time slice finds the time quantum by taking the greatest common divisor(GCD) of all the execution time of the processes. After every unit of time slice the laxity of each remaining process(present in the ready queue) is calculate. The loop is continued until all the processes are being executed by the cpu. Here as the GCD of execution the execution time is always greater than equal to 1 so loop will be continue for lesser no of time or same no of time i.e. when the GCD of the execution time is greater than 1 in that case scheduling algorithm will show better performance(loop will continue less no of times). Our proposed ILLF algorithm shows less context switching as compared to MLLF scheduling Algorithm.
3.1. Pseudo Code of the Proposed Algorithm

```
1. I/P : Process(Pi), Execution Time (Ei), Dead Line (Di);
   O/P: Context Switch (CS), number of times laxity calculated (N)
2. Initialize Ready Queue=0;
goto 3;
3. /*Calculation of the Time Quantum using GCD of the execution time of all the processes*/
   \[ \text{TQ}=\text{GCD}(E_1\text{to }E_n) \] //where i varies from 1 to n
goto 4;
4. if \( E_i=0 \)
   { Remove P_i; n=n-1; i=i-1;
     if \( n=0 \)
     { Stop and Exit; }
     Else
     goto 4;
   else
   L_i=Di-E_i;}
goto 5;
5. if(i<n)
   { i=i+1; goto 4;}
else
goto 6;
6. Sort P_1 to P_n according to laxity in ascending order
goto 7;
7. if \( L_j=L_{j+1} \)
   goto 8;
else
   { Execute P_j for TQ time; for(k=0;k<n;k++)
     { D_k=D_k-TQ;}
     E_j=E_j-TQ; goto 4; }
8. if \( E_j\geq E_{j+1} \)
   goto 9;
else
   { Execute P_{j+1} till completion
     for(k=0;k<n;k++)
     { D_k=D_k-E_{j+1}; E_{j+1}=0;goto 4;}
   }
9. Execute P_j till completion
   for(k=0;k<n;k++)
   { D_k=D_k-E_j;
     }
   E_j=0;goto 4;
```
3.2. Flowchart of the Proposed Algorithm

```
start

Initialize processes in ready Queue
i=1, j=0, TQ=GCD(E_1 to E_n)

E_i = 0

Remove P_i

i=i+1

Li=Di-E_i

E_i =< L_i+1

n=n-1 and i=i+1

n==0

Execute P_j till completion
E_j = E_j - TQ;
For k = 0 to n
D_k = D_k - TQ;

No

L_j = L_j+1

execute P_j for TQ time
E_j = E_j - TQ;
For k = 0 to n
D_k = D_k - TQ;

yes

n==0

Stop

execute P_{j+1} till completion
E_j = E_j - TQ;
For k = 0 to n
D_k = D_k - E_j;
E_j = 0;

execute P_j till completion
For k = 0 to n
D_k = D_k - E_j;
E_j = 0;

i < n

no

yes

Stop

execute P_j till completion
E_j = E_j - TQ;
For k = 0 to n
D_k = D_k - E_j;
E_j = 0;

no

yes

no

L_j = L_j+1

Execute P_j for TQ time
E_j = E_j - TQ;
For k = 0 to n
D_k = D_k - TQ;

execute P_{j+1} till completion
E_j = E_j - TQ;
For k = 0 to n
D_k = D_k - E_j;
E_j = 0;

n=n-1 and i=i+1

n==0

Stop

execute P_j till completion
For k = 0 to n
D_k = D_k - E_j;
E_j = 0;
```

H.S Behera et al. / International Journal of Engineering Science and Technology (IJEST)

ISSN: 0975-5462

Vol. 4 No.04 April 2012

1315
4. Experimental Analysis of the Proposed Algorithm

Let there are four periodic tasks namely $P_1, P_2, P_3$ and $P_4$. Suppose that they arrive at the same time, their execution time is 2, 4, 8, 12 respectively and deadline is 5, 6, 26, 28 respectively.

<table>
<thead>
<tr>
<th>Processes</th>
<th>Remaining Execution Time</th>
<th>Deadline</th>
<th>Laxity</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_1$</td>
<td>2</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>$P_2$</td>
<td>4</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>$P_3$</td>
<td>8</td>
<td>26</td>
<td>18</td>
</tr>
<tr>
<td>$P_4$</td>
<td>12</td>
<td>28</td>
<td>16</td>
</tr>
</tbody>
</table>

Gantt Chart of MLLF algorithm is described below

Here the Context Switching point is indicated by downward arrows. Five Context Switching occur in MLLF Scheduling Algorithm as shown in the above figure 1. Here the time quantum is 1milli sec

Gantt Chart of Proposed ILLF Scheduling Algorithm is described below

Here the Context Switching point is indicated by downward arrows. Five Context Switching occur in ILLF Scheduling Algorithm as shown in the above figure 1. Here the time quantum is 1milli sec

ISSN : 0975-5462 Vol. 4 No.04 April 2012 1316
Here the Context Switching point is indicated by downward arrow. Four Context Switching occur in ILLF Scheduling algorithm as shown in figure 2.

5. Performance Evaluation

In this section, MLLF and ILLF scheduling algorithm are tested and its performance is calculated. The result shows that the number of context switching in MLLF Scheduling Algorithm is more than our proposed ILLF Scheduling Algorithm. As context switching decreases the performance of the processor, hence our proposed ILLF scheduling algorithm shows better performance over MMLF scheduling algorithm.

![Comparison of number of context switches in both the algorithms](image1)

![Comparison of the ratio of number of context switches with tasks](image2)
In figure 3 and 4 the performance based upon context switches is analyzed. This shows that our proposed algorithm has less number of context switching and the ratio of the context switching of MMLF algorithm to our proposed ILLF algorithm is always less than equal to one. For that reason our proposed ILLF scheduling algorithm is better compared to that of MLLF Scheduling Algorithm. 

Our proposed algorithm is based on an intelligence time slice which is the Greatest Common Divisor of the execution time of all processes.

Hence our proposed ILLF scheduling algorithm is more efficient than Modified Least Laxity First (MLLF) Scheduling algorithm.

6. Conclusion

In this research, we proposed An Improved Least Laxity First (ILLF) Scheduling Algorithm with intelligence time slice that removes the drawbacks of the Least Laxity First(LLF) Scheduling Algorithm and Modified Least Laxity First (MLLF) Scheduling Algorithm. Here the performance evaluation shows that in case of MLLF Scheduling Algorithm the no of Switching is more as compared to our proposed ILLF Scheduling Algorithm, which is a overhead to the system and reduces the system performance. Thus this is an efficient dynamic priority based scheduling Algorithm for Real-Time systems.

References

AUTHORS BIOGRAPHY

1. Dr H.S. Behera is currently working as faculty in Dept. of Computer Science and Engineering in Veer Surendra Sai University of Technology (VSSUT), Burla, Odisha, India. His areas of interest include Operating Systems, Data Mining, Softcomputing and Distributed Systems.

2. Satyajit Khuntia and Soumyashree Nayak are the Final Year B.Tech. IT students in Veer Surendra Sai University of Technology (VSSUT), Burla, Odisha, India.