CHAPTER 4

SCHEDULING USING NON-UNIFORM LAXITY AS A SLACK DISTRIBUTION MEASURE

While scheduling tasks using the modified proportionate fairness strategy, tasks are scheduled using minimum resource, leaving a large amount of resource unutilized. This unutilized resource is cumulated and stored as slack. Slack is the additional resource available which can be utilized to schedule tasks that arrive at a later point of time with inadequate resource. This slack may be distributed among the needy tasks based on a suitable slack distribution measure. The second scheduling approach proposed in this thesis is a non-uniform laxity based slack distribution measure to ensure that all tasks get slack in proportion to their actual requirement.

The chapter is organized as follows: Section 4.1 explains how non-uniform laxity is obtained. Section 4.2 explains the algorithm with an example. Section 4.3 discusses the simulation results using Cheddar and SESC. Section 4.4 concludes this chapter.

4.1 NON-UNIFORM LAXITY

The unutilized portion of resources after scheduling tasks by the modified proportionate fairness strategy is cumulated as slack. The slack is distributed among tasks based on their requirement using a slack distribution measure. In this work, a new slack distribution measure, namely, non-uniform laxity has been proposed. The laxity of a task is made proportionate to the share of resource actually utilized by the task using a weight measure.

The weight of the task is given by

$$\mathrm{Wt(task)} = \left(\frac{\mathrm{exec\,min(task)}}{\mathrm{WCET(task)}} \right) \mathrm{xcore\,cos\,t} \qquad (4.1)$$

where, execmin (task) is the minimum execution time, WCET (task) is the Worst Case Execution Time and corecost is the computation cost involved to execute the task on the core. Laxity is a measure of the flexibility available for scheduling a task.

The laxity of the task is given by

$$\mathrm{Lax\ (task) = (Deadline\ (task)\text{-}(Arrivaltime\ (task) + execmin\ (task)} \qquad (4.2)$$

where Lax (task) is the laxity, Deadline (task) is the deadline and execmin (task) is the minimum execution time of the task.

Non-uniform laxity of the task is given by

$$\mathrm{nlax\ (task) = (Lax\ (task)\ x\ Wt\ (task))} \qquad (4.3)$$

The task utilization based on the non-uniform laxity is obtained using the relation

$$U_{\mathrm{nlax}}(\mathrm{task}) = \frac{\mathrm{nlax(task)}}{\mathrm{Deadline(task)}} \qquad (4.4)$$

The total task utilization (Wen et al, 2007) is computed using the equation

$$U_{tot} = 1.5 + | u_{max} - 0.5| \qquad (4.5)$$

where umax is the maximum utilization of the task set under consideration. The modified task utilization is obtained as a product of U_{nlax}(task) and U_{tot}. The slack to be granted to each task is given as a product of modified task utilization and non-uniform laxity.

The different constraints to be taken into account while carrying out the scheduling analysis is discussed in the next section.

4.1.1 Constraints for Non-Uniform Laxity

- For a system with less than two cores, complete schedulability of all task within the task sets can be obtained by evaluating the following constraint (Chao et al 2006)

  $$U_{tot} <= (z+1)/2 \tag{4.6}$$

  where, U_{tot} is the total task utilization and z is the number of cores.

- For a system with more than two cores, complete schedulability is given by the following constraint (Chao et al, 2006)

  $$U_{tot} < (z - ((z-1) \text{ x } u_{max})) \tag{4.7}$$

  where U_{tot} is the total task utilization, z is the total number of cores and u_{max} is the maximum task utilization.

- When the non-uniform laxity of a task becomes 0, the task set will be completely schedulable, if and only if the total task utilization is less than L (Chao et al, 2006). The factor L is given by

  $$L = \left(z \times \left(1 - (\frac{1}{e}) \right) \right) \tag{4.8}$$

  where, z is the number of cores and e is Euler's number.

The tasks execution status is analysed based on the following non-uniform laxity conditions.

4.1.2 Scheduling Analysis based on Non-Uniform Laxity

- When the non-uniform laxity is positive, tasks are stored in a holding queue and are not dispatched for execution.
- When the non-uniform laxity approaches zero, the tasks are prioritized in a queue for execution.
- When the non-uniform laxity becomes zero, tasks are urgently dispatched for execution whether they are in the holding queue or in the execution queue.
- When the non-uniform laxity becomes negative, tasks are discarded from the holding or execution queue, as these tasks will surely miss their deadline. The scheduling analysis eliminates the condition of non-uniform laxity turning out to be negative as far as possible, by monitoring the non-uniform laxity periodically.

The algorithm and an example to illustrate the working of the same are discussed in the next section.

4.2 ALGORITHM

The expressions used in the algorithm are given below:

- $L = \left(z \times \left(1 - \left(\frac{1}{e} \right) \right) \right)$, z is the total number of cores and e is Euler's number
- Weight of task is given by $\mathrm{Wt(task)} = \left(\left(\frac{\mathrm{execmin(task)}}{\mathrm{WCET(task)}} \right) \times \mathrm{corecost} \right)$ where execmin (task) is the minimum execution time, WCET (task) is the worst case execution time and corecost is the computation cost for the task to be executed on the core.

- Laxity of the task is given by Lax = (Deadline – (arrival time + execmin)), where execmin is the minimum execution time of the task.
- Non-uniform laxity is given by nlax = (Wt x Lax)
- Task utilization of EDF algorithm is given by U_{EDF} = (execmin / Deadline)
- Task utilization of Non-uniform laxity algorithm is given by U_{nlax} = (nlax / Deadline)
- Total task utilization is given by $U_{tot} = 1.5 + |u_{max} - 0.5|$, where u_{max} is the maximum task utilization
- Modified task utilization MU is given by = (U_{tot} x U_{nlax})
- Slack granted to tasks SL = (MU x nlax)

The algorithm is outlined below:

```
• For all cores do
    • For all tasks do
        • Start scheduling tasks using EDF
        • Define two queues namely holding (H) and execution
          queue (X) for performing operations on tasks
        • Compute factor L, original task utilization,
          modification factor, modified task utilization, weight
          of tasks, laxity and non-uniform laxity
        • If (modified task utilization <((z+1)/2)) then
            o Append task to execution queue
        fi
```

```
                • If (modified task utilization <2) then
                    o Append task to holding queue H
                  fi
                • If (modified task utilization >=2) then
                    o Append task to execution queue X
                  fi
            • Execute the tasks on the respective cores
            • If (non-uniform laxity >0) then
                o Append task to holding queue
              fi
            • If ((non-uniform laxity =0) & (modified task utilization <2))
                then
                o Urgently remove task from holding queue and
                  append to execution queue
              fi
        • If ((non-uniform laxity =0) & (modified task utilization >=2))
          then
                o Remove task from holding queue and append to
                  execution queue
              fi
        • If (modified task utilization <2) then
                o Execute task on next core
          fi
        • If ((non-uniform laxity<0) & (modified task utilization > L))
              o Declare task has missed its deadline
              o Discard the task from the holding queue H
          fi
      od
    od
• End the procedure
```

4.2.1 Example

A task set comprising of six tasks is assumed to be scheduled on four cores. The core speeds of all the cores are assumed to be 1 GHz. The task parameters are indicated in Tables 4.1 and 4.2.

Table 4.1 Task parameters

Task	Arrival time	Minimum execution time	Worst Case Execution time	Dead line (d)
T1	5	10	20	35
T2	26	11	22	53
T3	49	20	40	91
T4	7	30	60	76
T5	68	25	50	126
T6	20	28	56	100

The total task utilization of the task set is given by $U_{tot} = 1.5 + |u_{max} - 0.5|$, where $u_{max} = 0.4$ (as indicated in table 4.2), the maximum task utilization of the task set. Hence, $U_{tot} = 1.5 + 0.1 = 1.6$.

Table 4.2 Calculated parameters

Task	Weight	Laxity	Non-uniform laxity	Task utilization	Task utilization due to nlax (Unlax)	Modified task utilization	Slack granted to tasks SL
T1	0.5	20	10.0	0.3	0.3	0.5	4.6
T2	0.5	16	8.0	0.2	0.2	0.2	1.9
T3	0.5	22	11.0	0.2	0.1	0.2	2.1
T4	0.5	39	19.5	0.4	0.3	0.4	8.0
T5	0.5	33	16.5	0.2	0.1	0.2	3.5
T6	0.5	52	26.0	0.3	0.3	0.4	10.8

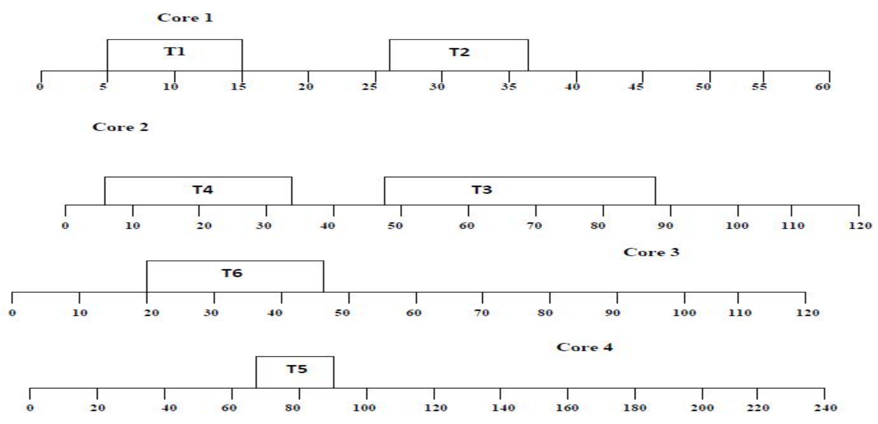

Figure 4.1 Conventional EDF Schedule without slack time measures

In Figure 4.1, tasks are scheduled based on conventional EDF in increasing order of deadlines. Task utilization of individual cores can be computed based on the entire allocated time. The sum of the task utilizations on any core is less than 1. The sum of task utilizations on core 1, core 2, core 3 and core 4 are 0.5, 0.6, 0.3 and 0.2 respectively, which is less than 1.

The EDF schedule is modified by incorporating the non-uniform laxity component. This is accomplished by modifying the task utilizations.

The modified task utilizations on core 1, core 2, core 3 and core 4 is 0.7, 0.6, 0.4 and 0.2 respectively, as shown in Figure 4.2. The maximum task utilization is 0.39 when tasks are scheduled with minimum resource allocated. Hence, the total task utilization is 1.5 + |0.4 – 0.5| = 1.6. For task 1, the laxity is (35 – (5+10)) = 20 as shown in Table 4.2. The weight of task 1 is given by $\left(\frac{10}{20}\right) \times 0.33 = 0.2$. The modified task utilization of task 1 is 0.2. The slack to be granted to task 1 is 0.5. Similarly, slack for all other tasks are computed.

For all tasks to be schedulable, the factor L is given by $\left(4\times\left(1-\left(\frac{1}{2.7}\right)\right)\right)=2.5$. The modified task utilizations on core 1, core 2, core 3 and core 4 are 0.7, 0.6, 0.4 and 0.2 respectively. As the modified task utilizations is less than L, all tasks in the task set are schedulable.

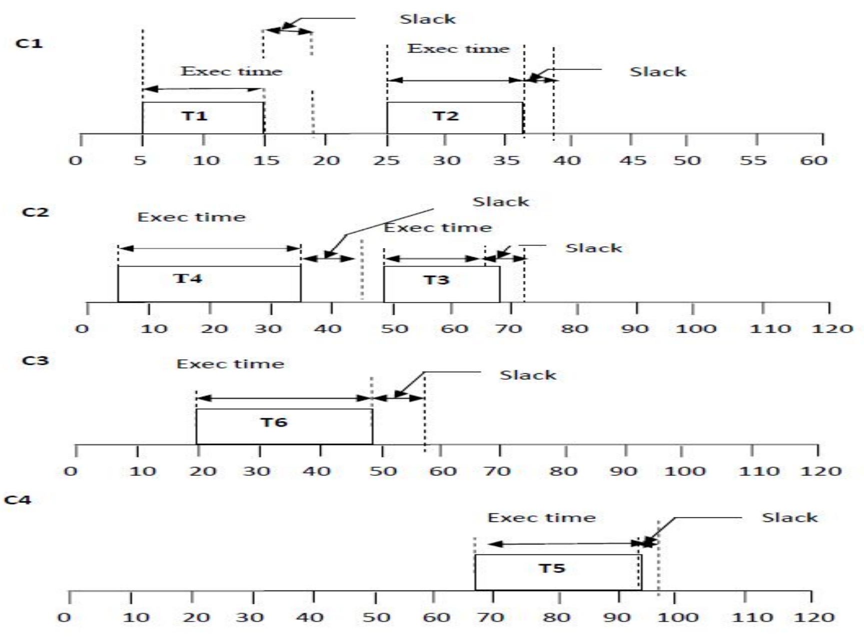

Figure 4.2 Modified EDF Schedule incorporating non-uniform laxity

4.3 SIMULATION RESULTS

The algorithm has been simulated by varying the number of cores up to 100 and tasks up to 150. The algorithm has been simulated using Cheddar, a real time scheduling tool and SESC, an architectural simulator. The performance of the algorithm is evaluated based on the resource utilization and task schedulability. The simulations have been carried out by comparing the EDF based non-uniform laxity approach with conventional EDF approach. The non-uniform laxity based approach has been compared against existing laxity based approaches, namely, Least Laxity First, Zero laxity and Uniform laxity approach (Cirinei and Baker 2007).

4.3.1 Resource Utilization

The EDF based non-uniform laxity approach has been compared against existing scheduling approaches like EDF, Least laxity, Zero laxity and Uniform laxity based approach using Cheddar. Cheddar gives the schedule for the tasks. Based on the schedule arrived using Cheddar, the algorithm is simulated on SESC to observe the execution of tasks on cores. Table 4.3 compares EDF, Least laxity, Zero laxity and Uniform laxity based approach using Cheddar. For example, when 6 tasks are scheduled on 4 cores, the conventional EDF based non-uniform laxity approach utilises 69.0 units, whereas the EDF, Least laxity, Zero laxity and Uniform laxity approaches utilise 56.0 units, 51.0, 48.0 and 51.0 units respectively . It can be observed from table 4.3, that the EDF based non-uniform laxity approach gives better results compared to the other laxity based approaches. The EDF based non-uniform laxity approach is then simulated on SESC. The EDF based non-uniform laxity approach improves resource utilization, compared to EDF, Least laxity, Zero laxity based approach and Uniform laxity based approach by 31%, 44%, 34% and 37% respectively. The results are also illustrated in Figure 4.3.

Table 4.3 Improvement in resource utilization for EDF with non-uniform laxity approach Vs EDF, Least laxity first, Zero laxity and Uniform laxity based approach-using Cheddar

Number of cores	Number of tasks	Resource utilization					Improvement of Non-uniform laxity approach Vs EDF	Improvement of Non-uniform laxity approach Vs Least laxity	Improvement of Non-uniform laxity approach Vs Zero laxity	Improvement of Non-uniform laxity approach Vs Uniform laxity
		EDF (EDF)	Least lax (LL)	Zero laxity (ZL)	Uniform laxity (Ulax)	Non-uniform laxity (nlax)				
4	6	56	51	48	51	69	23.2	35.3	43.8	35.3
8	12	51	44	43	47	62	21.6	40.9	44.2	31.9
12	18	37	38	39	41	58	56.8	52.6	48.7	41.5
16	24	32	27	34	31	40	25.0	48.1	17.6	29.0
32	48	27	24	29	25	35	29.6	45.8	20.7	40.0
64	96	26	21	23	21	30	15.4	42.9	30.4	42.9
100	150	17	17.2	19	18	25	47.1	45.3	31.6	38.9
Average		**35.1**	**31.7**	**33.6**	**33.4**	**45.6**	**31.2**	**44.4**	**33.9**	**37.1**

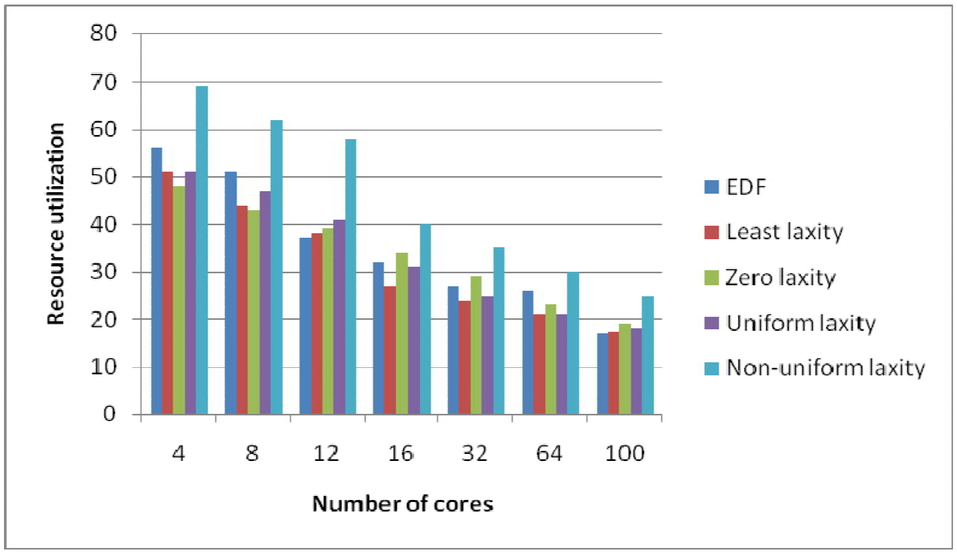

Figure 4.3 Resource utilization of EDF based non-uniform laxity Vs EDF, Least laxity, Zero laxity and Uniform laxity based approach-using Cheddar

The EDF based non-uniform laxity approach has been compared against conventional EDF approach using SESC. The results are tabulated in Table 4.4. For example, when 6 tasks are scheduled on 4 cores, the conventional EDF based non-uniform laxity approach utilises 71.0 units, whereas the conventional EDF approach utilises 56.0 units. The EDF based non-uniform laxity approach improves resource utilization, compared to conventional EDF approach by 33%. The results are also illustrated in Figure 4.4.

Table 4.4 Improvement in resource utilization for EDF based non-uniform laxity approach Vs conventional EDF approach-using SESC

Number of cores	Number of tasks	Resource utilization		Improvement nlax EDF over EDF (%)
		EDF (EDF)	EDF based non-uniform laxity (nlax EDF)	
4	6	56	71	26.8
8	12	51	65	27.5
12	18	37	46	24.3
16	24	32	46	43.8
32	48	27	36	33.3
64	96	26	36	38.5
100	150	17	23	35.3
Average		**35.1**	**46.1**	**32.8**

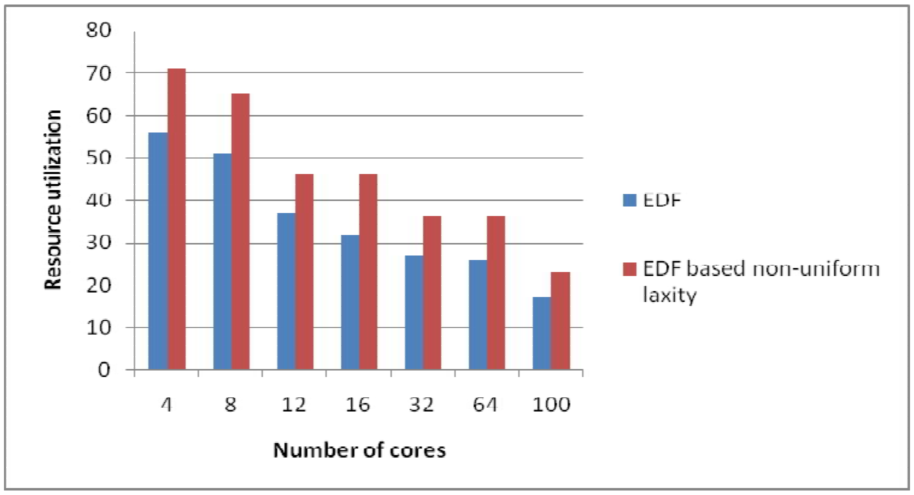

Figure 4.4 Improvement in resource utilization of EDF based non-uniform laxity Vs conventional EDF approach-using SESC

4.3.2 Task Schedulability

The EDF based non-uniform laxity approach has been compared against existing scheduling approaches like EDF, Least laxity, Zero laxity and Uniform laxity based approach using Cheddar. Cheddar gives the schedule for the tasks. Based on the schedule arrived, the algorithm is simulated on SESC to observe the execution of tasks on cores. Table 4.5 compares EDF, Least laxity, Zero laxity and Uniform laxity based approach using Cheddar. For example, when 6 tasks are scheduled on 4 cores, the EDF based non-uniform laxity approach schedules 5 tasks, whereas the EDF, Least laxity, Zero laxity and Uniform laxity approaches schedule 3, 4, 3 and 4 tasks respectively . The EDF based non-uniform laxity approach is then simulated on SESC. The EDF based non-uniform laxity approach improves task schedulability, compared to EDF, Least laxity, Zero laxity based approach and Uniform laxity based approach by 59%, 42%, 44% and 21% respectively. The results are also illustrated in Figure 4.5.

Table 4.5 Improvement in task schedulability - EDF based non-uniform laxity approach Vs EDF, Least laxity, Zero laxity and Uniform laxity based approaches – using Cheddar

No. of tasks	Task schedulability										Improvement of NLAX over EDF (%)	Improvement of NLAX over LL (%)	Improvement of NLAX over ZL (%)	Improvement of NLAX over UL (%)
	EDF		Least laxity (LL)		Zero laxity (ZL)		Uniform laxity (UL)		Non-uniform laxity (NLAX)					
	Tasks scheduled	Tasks missed	Tasks scheduled	Tasks missed	Tasks scheduled	Tasks missed	Tasks scheduled	Tasks missed	Tasks scheduled	Tasks missed				
6	3	3	4	2	3	3	4	2	5	1	66.7	25.0	66.7	25.0
12	7	5	9	3	7	5	9	3	11	1	57.1	22.2	57.1	22.2
18	10	8	12	6	13	5	14	4	17	1	70.0	41.7	30.8	21.4
24	14	10	15	9	16	8	18	6	22	2	57.1	46.7	37.5	22.2
48	29	19	27	21	28	20	39	9	46	2	58.6	70.4	64.3	17.9
96	49	47	53	43	67	29	71	25	87	9	77.6	64.2	29.9	22.5
150	116	34	121	29	125	25	130	20	149	1	28.4	23.1	19.2	14.6
Average	**33**		**34**		**37**		**41**		**48**		**59.4**	**41.9**	**43.6**	**20.9**

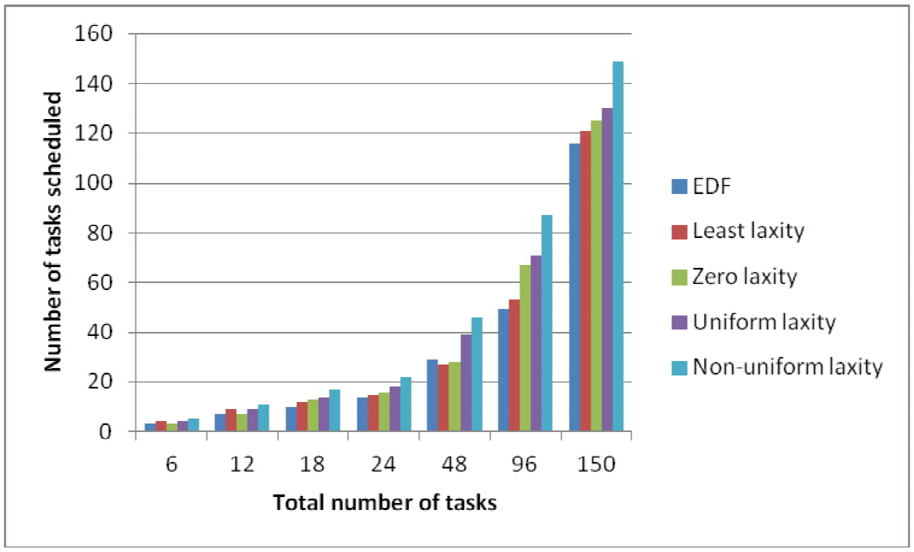

Figure 4.5 Improvement in task schedulability - EDF based non-uniform laxity approach Vs EDF, Least laxity first, Zero laxity and Uniform laxity based approach – using Cheddar

The EDF based non-uniform laxity approach has been compared against conventional EDF approach using SESC. The results are tabulated in Table 4.6. For example, when 6 tasks are to be scheduled, the EDF based non-uniform laxity approach schedules 5 tasks, whereas the conventional EDF approach schedules only 3 tasks. The EDF based non-uniform laxity approach improves task schedulability by 58%, compared to conventional EDF approach. The results are also illustrated in Figure 4.6.

Table 4.6 Improvement in task schedulability - EDF based non-uniform laxity approach Vs conventional EDF - using SESC

Number of cores	Number of tasks	Task schedulability				Improvement of EDF-nlax over EDF (%)
		EDF		EDF modified with nlax		
		Tasks scheduled	Tasks missed	Tasks scheduled	Tasks missed	
4	6	3	3	5	1	66.7
8	12	7	5	10	2	42.9
12	18	10	8	17	1	70.0
16	24	14	10	21	3	50.0
32	48	29	19	46	2	58.6
64	96	49	47	94	2	91.8
100	150	116	34	146	4	25.9
Average		**33**		**48**		**58.0**

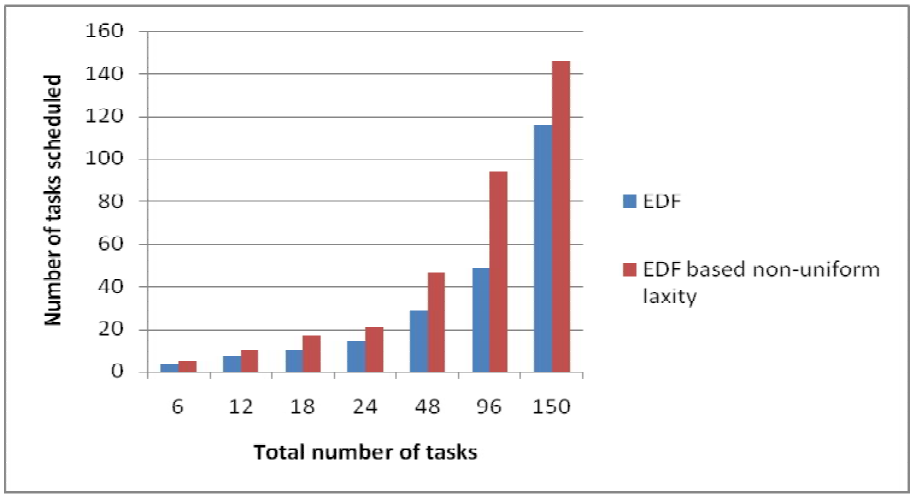

Figure 4.6 Improvement in task schedulability - EDF based non-uniform laxity approach Vs conventional EDF-using SESC

4.4 SUMMARY

This chapter has proposed a new laxity based approach to distribute slack resource proportionate to the requirement and priority, thus preventing wastage of resources. The simulation results show that non-uniform laxity based approach improves resource utilization and task schedulability.

The non-uniform laxity discussed in this chapter is effective in slack distribution as long as the scheduling parameters can be estimated precisely. However, in real time situations, where there is lack of precise knowledge of these parameters, the scheduler still has to distribute slack, so as to ensure that no task misses its deadline and all tasks get their entitled share of slack resource. These issues are addressed in the next chapter.