Fixed-Priority Multiprocessor Scheduling with Liu & Layland’s Utilization Bound

Nan Guan, Martin Stigge, Wang Yi

Uppsala University, Sweden
Scheduling of Multi-task System

- multi-rate real-time task system

- each task

Utilization: $C_i / T_i$
Liu and Layland’s Utilization Bound

- Liu and Layland’s utilization bound for single-processor scheduling [Liu1973] (the 19th most cited paper in computer science)

\[ \Theta(N) = N \left( 2^{\frac{1}{N}} - 1 \right) \]

- \( N \): the number of tasks, \( N \to \infty \), \( \Theta(N) \approx 69.3\% \)
- optimal

\[ \sum C_i/T_i \leq N \left( 2^{1/N} - 1 \right) \]

\( \Rightarrow \) the task set is schedulable
Multiprocessor Scheduling

Significantly more difficult

- Bin-packing problem
- Hard to identify the worst-case scenario
- Suffer from timing anomalies
- May lead to arbitrarily low utilization
Open Problem

- find a multiprocessor scheduling algorithm that can achieve Liu and Layland’s utilization bound.

\[
\frac{\sum C_i/T_i}{M} \leq N(2^{1/N} - 1)
\]

\[\Rightarrow\text{ the task set is schedulable}\]
Multiprocessor Scheduling

Global Scheduling

Partitioned Scheduling

Partitioned Scheduling with Task Splitting

new task

waiting queue

cpu 1

cpu 2

cpu 3
Best Known Results

Lehoczky et al. CMU ECRTS 2009
Best Known Results

Our New Result

Liu and Layland’s Utilization Bound

- [OPODIS’08]: 38%
- [TPDS’05]: 50%
- [ECRTS’03]: 50%
- [RTSS’04]: 50%
- [RTCSA’06]: 66%

Fixed Priority
Dynamic Priority
Fixed Priority
Dynamic Priority
Fixed Priority
Dynamic Priority

Global
Partitioned

Task Splitting

Multiprocessor Scheduling
Lehoczky’s Algorithm [ECRTS’09]

- sort all tasks in decreasing order of utilization

Diagram:
- lowest utilization: 8, 7, 6, 5, 4, 3
- highest utilization: 2, 1
Lehoczky’s Algorithm [ECRTS’09]

- pick up one processor, and assign as many tasks as possible

```
8 7 6 5 4 3 2 1
```

P1

Lowest utilization

Highest utilization
Lehoczky’s Algorithm [ECRTS’09]

- pick up one processor, and assign as many tasks as possible

Diagram:
- Lowest utilization:
  - 7
  - 6
  - 5
  - 4
  - 3
  - 2
  - 1
- Highest utilization:
  - P1
  - 8
Lehoczky’s Algorithm [ECRTS’09]

- pick up one processor, and assign as many tasks as possible
Lehoczky’s Algorithm [ECRTS’09]

- pick up one processor, and assign as many tasks as possible
Lehoczky’s Algorithm [ECRTS’09]

- pick up one processor, and assign as many tasks as possible

![Diagram of task assignments to processors P1 and P2.](image)
Lehoczky’s Algorithm [ECRTS’09]

- pick up one processor, and assign as many tasks as possible

```
P1
6
7
8

P2
6
```

lowest utilization

highest utilization
Lehoczky’s Algorithm [ECRTS’09]

- pick up one processor, and assign as many tasks as possible

P1

\[
\begin{array}{c}
6^1 \\
7 \\
8 \\
\end{array}
\]

P2

\[
\begin{array}{c}
5 \\
6^2 \\
\end{array}
\]

lowest utilization

highest utilization
Lehoczky’s Algorithm [ECRTS’09]

- pick up one processor, and assign as many tasks as possible

Lowest utilization:
- P1: 1
- P2: 4

Highest utilization:
- P1: 2
- P2: 6

Tasks assigned:
- P1: 1, 7, 8
- P2: 4, 5, 6
Lehoczky’s Algorithm [ECRTS’09]

- pick up one processor, and assign as many tasks as possible
Lehoczky’s Algorithm [ECRTS’09]

- pick up one processor, and assign as many tasks as possible

Lowest utilization:

- P1:
  - 6
  - 7
  - 8

- P2:
  - 2
  - 3
  - 4
  - 5
  - 6

Highest utilization:

- P1:
  - 2
  - 1
Lehoczky’s Algorithm [ECRTS’09]

- pick up one processor, and assign as many tasks as possible

Diagram:
- Lowest utilization:
  - P1: 6¹, 7, 8
  - P2: 3, 4, 5, 6²
  - P3: 2²

- Highest utilization:
  - 1
Lehoczky’s Algorithm [ECRTS’09]

- pick up one processor, and assign as many tasks as possible

![Diagram showing task assignments and utilization]

- P1: Tasks 6, 7, 8
- P2: Tasks 2, 3, 4, 5
- P3: Tasks 1, 2

The diagram illustrates the assignment of tasks (1 to 8) to processors P1, P2, and P3, with tasks ordered from highest to lowest utilization.
Lehoczky’s Algorithm [ECRTS’09]

- pick up one processor, and assign as many tasks as possible

key feature:
depth-first partitioning with decreasing utilization order
Lehoczky’s Algorithm [ECRTS’09]

- pick up one processor, and assign as many tasks as possible

Utilization Bound: 65%
Our Algorithm

width-first partitioning
with increasing priority order
Our Algorithm

- sort all tasks in increasing priority order

<table>
<thead>
<tr>
<th>Priority Level</th>
<th>Task Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Highest Priority</td>
<td>1, 2, 3, 4, 5, 6, 7</td>
</tr>
<tr>
<td>Lowest Priority</td>
<td></td>
</tr>
</tbody>
</table>

The tasks are sorted from highest priority (1) to lowest priority (7).
Our Algorithm

- select the processor on which the assigned utilization is the **lowest**
Our Algorithm

- select the processor on which the assigned utilization is the **lowest**

![Diagram showing processor utilization and priority]

- highest priority
- lowest priority

- P1
- P2
- P3
Our Algorithm

- select the processor on which the assigned utilization is the lowest
Our Algorithm

- select the processor on which the assigned utilization is the lowest

Priority Levels:
- **Highest Priority**
  - P1
  - P2
  - P3

- **Lowest Priority**
  - P1
  - P2
  - P3

Processor Utilization:
- P1: 7
- P2: 6
- P3: 5

Tasks:
- Task 1
- Task 2
- Task 3
- Task 4
- Task 3
- Task 2
- Task 1
Our Algorithm

- select the processor on which the assigned utilization is the lowest

P1

- P2
- P3

1
2
3

4
5
6
7

highest priority
lowest priority
Our Algorithm

- select the processor on which the assigned utilization is the lowest
Our Algorithm

- select the processor on which the assigned utilization is the lowest
Our Algorithm

- select the processor on which the assigned utilization is the lowest

![Diagram showing processors and their priority levels]

- lowest priority
- highest priority
Our Algorithm

- select the processor on which the assigned utilization is the lowest
Our Algorithm

- select the processor on which the assigned utilization is the lowest

<table>
<thead>
<tr>
<th></th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>2\textsuperscript{1}</td>
<td>1\textsuperscript{1}</td>
<td>1\textsuperscript{2}</td>
</tr>
<tr>
<td></td>
<td>4\textsuperscript{4}</td>
<td>2\textsuperscript{2}</td>
<td>3\textsuperscript{3}</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>6</td>
<td>5</td>
</tr>
</tbody>
</table>

highest priority

lowest priority
Our Algorithm

- select the processor on which the assigned utilization is the lowest

key feature: width-first partitioning with increasing priority order
Comparison

- maximal number of task splitting
  both are $M-1$

Ours: width-first
(increasing priority order)

Lehoczky’s: depth-first
(decreasing utilization order)
Comparison

- why is our algorithm better?

Ours: width-first
(increasing priority order)

Lehoczky’s: depth-first
(decreasing utilization order)
Comparison

key point:
by our algorithm, split tasks generally have high priorities on each processor

Ours: width-first
(increasing priority order)

Lehoczky’s: depth-first
(decreasing utilization order)

<table>
<thead>
<tr>
<th>P1</th>
<th>P2</th>
<th>P3</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>1^2</td>
<td>1^1</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>P1</th>
<th>P2</th>
<th>P3</th>
</tr>
</thead>
<tbody>
<tr>
<td>6^1</td>
<td>3^1</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>5</td>
<td>3^2</td>
</tr>
<tr>
<td>6^2</td>
<td>3^2</td>
<td></td>
</tr>
</tbody>
</table>
Split Task

Subtasks should be executed in the correct order.
Split Task

original utilization: \( U_i^k = \frac{c_i^k}{T_i} \)

synthetic utilization: \( V_i^k = \frac{c_i^k}{\tau_i^k} \)

split tasks cause “utilization increase”
Our Algorithm

- intuitions
  - high priority tasks have better chance to meet their deadlines
Our Algorithm

- intuition
  - an extreme scenario:
    - each subtask has the highest priority on each processor
    - can meet their deadlines anyway
    - no “utilization increase”
Theorem

for a task set in with each task \( T_i \) satisfies

\[
U_i \leq \frac{\Theta(N)}{1 + \Theta(N)}
\]

we have

\[
\sum \frac{C_i}{T_i} \leq N \left(2^{1/N} - 1\right)
\]

\[ \Rightarrow \text{ the task set is schedulable} \]

\[
\Theta(N) = N \left(2^{\frac{1}{N}} - 1\right)
\]

\[
\frac{\Theta(N)}{1 + \Theta(N)} \approx 0.41
\]

reasonable constraint in real-life systems
Our Algorithm

- problem of heavy tasks

- lowest priority

- highest priority

P1  P2  P3
Our Algorithm

- problem of heavy tasks

Highest priority:
- 1

Lowest priority:
- 7
- 6
- 5
- 4
- 3
- 2

Process:
- P1
- P2
- P3

Task:
- 8
Our Algorithm

- problem of heavy tasks

lowest priority: 6, 5, 4, 3, 2, 1

highest priority: 8, 7, 1

P1: [highest priority]
P2: [second highest priority]
P3: [lowest priority]
Our Algorithm

- problem of heavy tasks

1. highest priority
   - 1
2. 2
3. 3
4. 4
5. 5

- P1
  - 6
  - 8
- P2
  - 7
- P3
  - 6

lowest priority
Our Algorithm

- problem of heavy tasks

- highest priority:
  - P1: 5^2
  - P2: 5^1
  - P3: 1

- lowest priority:
  - P1: 4
  - P2: 3
  - P3: 2
Our Algorithm

- problem of heavy tasks

![Diagram showing the algorithm with priorities and tasks]

- highest priority
  - 1
- medium priority
  - 2, 3, 4
- lowest priority
  - 5, 6, 7, 8
Our Algorithm

- problem of heavy tasks

```
<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>51</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>52</td>
<td>6</td>
<td>52</td>
</tr>
</tbody>
</table>

P1: highest priority
P2: middle priority
P3: lowest priority
```
Our Algorithm

- problem of heavy tasks

P1

\[ \begin{array}{c}
5^1 \\
8
\end{array} \]

P2

\[ \begin{array}{c}
3^2 \\
7
\end{array} \]

P3

\[ \begin{array}{c}
4 \\
6
\end{array} \]

highest priority

lowest priority
Our Algorithm

- problem of heavy tasks

- lowest priority

- highest priority

- P1
  - 51
  - 8

- P2
  - 3
  - 52
  - 7

- P3
  - 2
  - 4
  - 6
Our Algorithm

- problem of heavy tasks

<table>
<thead>
<tr>
<th>P1</th>
<th>P2</th>
<th>P3</th>
</tr>
</thead>
<tbody>
<tr>
<td>5¹</td>
<td>3²</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>7</td>
<td>6</td>
</tr>
</tbody>
</table>

lowest priority: 5¹, 3², 8

highest priority: 5², 7, 6
The idea works for all tasks!

☐ To get rid of the constraint

\[ U_i \leq \frac{\Theta(N)}{1 + \Theta(N)} \]

■ pre-assign tasks with high utilization
The idea works for all tasks!

lowest priority

5
4
3
2
1

highest priority

P1
P2
P3
The idea works for all tasks!

highest priority

1
2
3
4
5

lowest priority

P1

P2

P3

pre-assign the heavy task
The idea works for all tasks!
By introducing the pre-assignment to the algorithm, we have

\[ \frac{\sum C_i/T_i}{M} \leq N(2^{1/N} - 1) \]

\[ \Rightarrow \text{ the task set is schedulable} \]
Conclusion

- Proposed multiprocessor scheduling algorithms with Liu and Layland’s utilization bound
  - works on “light” task sets with a simple width-first algorithm
  - works on any task set with a hybrid algorithm
    - pre-assigning
Conclusion

Liu and Layland’s Utilization Bound

[OPODIS’08] [TPDS’05] [ECRTS’03] [RTSS’04]

Fixed Priority Dynamic Priority Fixed Priority Dynamic Priority Fixed Priority Dynamic Priority

Global Partitioned

Task Splitting

Multiprocessor Scheduling
THANKS!

† [Andersson03ECRTS] Bjorn Andersson, Jan Jonsson: The Utilization Bounds of partitioned and Pfair Static-Priority Scheduling on multiprocessors are 50%. ECRTS 2003: 33-40

† [Andersson08OPODIS] Bjorn Andersson: Global Static-Priority Preemptive Multiprocessor Scheduling with Utilization Bound 38%. OPODIS 2008: 73-88

† [Andersson06RTCSA] Bjorn Andersson, Eduardo Tovar: Multiprocessor Scheduling with Few preemption. RTCSA 2006: 322-334


† [Lakshmanan09ECRTS] Karthik Lakshmanan, Ragunathan Rajkumar, John Lehoczky Partitioned Fixed-Priority Preemptive Scheduling for Multi-core Processors. ECRTS 20006
