Fair Lateness Scheduling: Reducing Maximum Lateness in G-EDF-like Scheduling

Jeremy P. Erickson
James H. Anderson
Basic Idea

- We will be examining a scheduler that is similar to global earliest-deadline-first (G-EDF).
- Upshot: for soft real-time, we can do better than G-EDF by making some small changes.
- Instead of going into proof details, will provide some intuition.
Background

- System with $m$ identical cores/processors.
- Arbitrary-deadline sporadic task model:
  - Worst-Case Execution Time $C_i$
  - Minimum Separation Time $T_i$
  - Relative Deadline $D_i$
  - Utilization $U_i = C_i / T_i$

\[
\begin{align*}
C_i &= 2 \\
T_i &= 3 \\
D_i &= 2 \\
U_i &= 2/3
\end{align*}
\]
Intuition - Uniprocessor Scheduling

• To gain intuition, we'll think about the *implicit deadline* case, where $D_i = T_i$.

• On a uniprocessor, can schedule using earliest-deadline-first as long as $\sum U_i \leq 1$. 
Intuition – Uniprocessor Scheduling

- $\tau_1$: $C_1 = 2$, $T_1 = D_1 = 4$, $U_1 = 0.5$
- $\tau_2$: $C_2 = 4$, $T_2 = D_2 = 8$, $U_2 = 0.5$
- Observe how schedule works:

![Diagram showing the schedule with release, deadline, and completion points for tasks $\tau_1$ and $\tau_2$.]
Intuition - EDF on Multiprocessors

- $\tau_1: C_1 = 2, T_1 = D_1 = 4, U_1 = 0.5$
- $\tau_2: C_2 = 2, T_2 = D_2 = 4, U_2 = 0.5$
- $\tau_3: C_3 = 8, T_3 = D_3 = 8, U_3 = 1.0$

CPU 1

CPU 2

Release  Deadline  Deadline Miss  Completion
Intuition - EDF on Multiprocessors

- $\tau_1$: $C_1 = 2$, $T_1 = D_1 = 4$, $U_1 = 0.5$
- $\tau_2$: $C_2 = 2$, $T_2 = D_2 = 4$, $U_2 = 0.5$
- $\tau_3$: $C_3 = 8$, $T_3 = D_3 = 8$, $U_3 = 1.0$
Intuition - EDF on Multiprocessors

- $\tau_1$: $C_1 = 2$, $T_1 = D_1 = 4$, $U_1 = 0.5$
- $\tau_2$: $C_2 = 2$, $T_2 = D_2 = 4$, $U_2 = 0.5$
- $\tau_3$: $C_3 = 8$, $T_3 = D_3 = 8$, $U_3 = 1.0$
Intuition - EDF on Multiprocessors

- $\tau_1$: $C_1 = 2$, $T_1 = D_1 = 4$, $U_1 = 0.5$
- $\tau_2$: $C_2 = 2$, $T_2 = D_2 = 4$, $U_2 = 0.5$
- $\tau_3$: $C_3 = 8$, $T_3 = D_3 = 8$, $U_3 = 1.0$

Incorrect decision - $\tau_3$ not running but has utilization 1!
Intuition - EDF on Multiprocessors

- $\tau_1$: $C_1 = 2$, $T_1 = D_1 = 4$, $U_1 = 0.5$
- $\tau_2$: $C_2 = 2$, $T_2 = D_2 = 4$, $U_2 = 0.5$
- $\tau_3$: $C_3 = 8$, $T_3 = D_3 = 8$, $U_3 = 1.0$
Intuition - EDF on Multiprocessors

- $\tau_1$: $C_1 = 2$, $T_1 = D_1 = 4$, $U_1 = 0.5$
- $\tau_2$: $C_2 = 2$, $T_2 = D_2 = 4$, $U_2 = 0.5$
- $\tau_3$: $C_3 = 8$, $T_3 = D_3 = 8$, $U_3 = 1.0$

Wasted CPU capacity - $\tau_3$ can't run parallel with itself!
Intuition - EDF on Multiprocessors

- $\tau_1$: $C_1 = 2$, $T_1 = D_1 = 4$, $U_1 = 0.5$
- $\tau_2$: $C_2 = 2$, $T_2 = D_2 = 4$, $U_2 = 0.5$
- $\tau_3$: $C_3 = 8$, $T_3 = D_3 = 8$, $U_3 = 1.0$
Intuition - EDF on Multiprocessors

- $\tau_1$: $C_1 = 2$, $T_1 = D_1 = 4$, $U_1 = 0.5$
- $\tau_2$: $C_2 = 2$, $T_2 = D_2 = 4$, $U_2 = 0.5$
- $\tau_3$: $C_3 = 8$, $T_3 = D_3 = 8$, $U_3 = 1.0$
Intuition - EDF on Multiprocessors

- \( \tau_1: C_1 = 2, T_1 = D_1 = 4, U_1 = 0.5 \)
- \( \tau_2: C_2 = 2, T_2 = D_2 = 4, U_2 = 0.5 \)
- \( \tau_3: C_3 = 8, T_3 = D_3 = 8, U_3 = 1.0 \)

Incorrect decision - \( \tau_3 \) not running but has utilization 1!
Intuition - EDF on Multiprocessors

- $\tau_1$: $C_1 = 2$, $T_1 = D_1 = 4$, $U_1 = 0.5$
- $\tau_2$: $C_2 = 2$, $T_2 = D_2 = 4$, $U_2 = 0.5$
- $\tau_3$: $C_3 = 8$, $T_3 = D_3 = 8$, $U_3 = 1.0$
Intuition - EDF on Multiprocessors

- $\tau_1$: $C_1 = 2$, $T_1 = D_1 = 4$, $U_1 = 0.5$
- $\tau_2$: $C_2 = 2$, $T_2 = D_2 = 4$, $U_2 = 0.5$
- $\tau_3$: $C_3 = 8$, $T_3 = D_3 = 8$, $U_3 = 1.0$

Wasted CPU capacity - $\tau_3$ can't run parallel with itself!
Intuition - EDF on Multiprocessors

- $\tau_1$: $C_1 = 2$, $T_1 = D_1 = 4$, $U_1 = 0.5$
- $\tau_2$: $C_2 = 2$, $T_2 = D_2 = 4$, $U_2 = 0.5$
- $\tau_3$: $C_3 = 8$, $T_3 = D_3 = 8$, $U_3 = 1.0$
Intuition - EDF on Multiprocessors

- \( \tau_1: C_1 = 2, T_1 = D_1 = 4, U_1 = 0.5 \)
- \( \tau_2: C_2 = 2, T_2 = D_2 = 4, U_2 = 0.5 \)
- \( \tau_3: C_3 = 8, T_3 = D_3 = 8, U_3 = 1.0 \)
Intuition - EDF on Multiprocessors

- $\tau_1$: $C_1 = 2$, $T_1 = D_1 = 4$, $U_1 = 0.5$
- $\tau_2$: $C_2 = 2$, $T_2 = D_2 = 4$, $U_2 = 0.5$
- $\tau_3$: $C_3 = 8$, $T_3 = D_3 = 8$, $U_3 = 1.0$
Intuition - EDF on Multiprocessors

- $\tau_1$: $C_1 = 2$, $T_1 = D_1 = 4$, $U_1 = 0.5$
- $\tau_2$: $C_2 = 2$, $T_2 = D_2 = 4$, $U_2 = 0.5$
- $\tau_3$: $C_3 = 8$, $T_3 = D_3 = 8$, $U_3 = 1.0$
Intuition - EDF on Multiprocessors

- $\tau_1$: $C_1 = 2$, $T_1 = D_1 = 4$, $U_1 = 0.5$
- $\tau_2$: $C_2 = 2$, $T_2 = D_2 = 4$, $U_2 = 0.5$
- $\tau_3$: $C_3 = 8$, $T_3 = D_3 = 8$, $U_3 = 1.0$

Incorrect decision - $\tau_3$ not running but has utilization 1!
Intuition - EDF on Multiprocessors

- $\tau_1$: $C_1 = 2$, $T_1 = D_1 = 4$, $U_1 = 0.5$
- $\tau_2$: $C_2 = 2$, $T_2 = D_2 = 4$, $U_2 = 0.5$
- $\tau_3$: $C_3 = 8$, $T_3 = D_3 = 8$, $U_3 = 1.0$

Wasted CPU capacity - $\tau_3$ can't run parallel with itself!
Intuition - EDF on Multiprocessors

- $\tau_1$: $C_1 = 2$, $T_1 = D_1 = 4$, $U_1 = 0.5$
- $\tau_2$: $C_2 = 2$, $T_2 = D_2 = 4$, $U_2 = 0.5$
- $\tau_3$: $C_3 = 8$, $T_3 = D_3 = 8$, $U_3 = 1.0$
Intuition - EDF on Multiprocessors

- $\tau_1$: $C_1 = 2$, $T_1 = D_1 = 4$, $U_1 = 0.5$
- $\tau_2$: $C_2 = 2$, $T_2 = D_2 = 4$, $U_2 = 0.5$
- $\tau_3$: $C_3 = 8$, $T_3 = D_3 = 8$, $U_3 = 1.0$
Intuition - EDF on Multiprocessors

- $\tau_1$: $C_1 = 2$, $T_1 = D_1 = 4$, $U_1 = 0.5$
- $\tau_2$: $C_2 = 2$, $T_2 = D_2 = 4$, $U_2 = 0.5$
- $\tau_3$: $C_3 = 8$, $T_3 = D_3 = 8$, $U_3 = 1.0$

CPU 1

CPU 2

Release

Deadline

Deadline Miss

Completion
Other Multiprocessor Schedulers

- EDZL
- Optimal Schedulers

\[ \tau_1 \]
\[ \tau_2 \]
\[ \tau_3 \]

↑ Release ↓ Deadline

Completion

CPU 1
CPU 2
Problem with Alternative Schedulers

- Can have high overheads
- May be difficult to implement
- Jobs can change priorities while running – causes problems with synchronization
Bounded Lateness

- G-EDF does provide *bounded lateness*. 
Bounded Lateness

- G-EDF does provide *bounded lateness*. 
Bounded Lateness

• G-EDF does provide *bounded lateness*.

**Schedule repeats itself.**

<table>
<thead>
<tr>
<th>τ₁</th>
<th>Release</th>
<th>Deadline</th>
<th>Deadline Miss</th>
<th>Completion</th>
</tr>
</thead>
<tbody>
<tr>
<td>τ₂</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>τ₃</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Bounded Lateness

- G-EDF does provide *bounded lateness*.

\[
\tau_1 \text{ and } \tau_2 \text{ are never late!}
\]

Schedule repeats itself.
Bounded Lateness

- G-EDF does provide \textit{bounded lateness}.

\begin{itemize}
  \item Schedule repeats itself.
  \item $\tau_1$ and $\tau_2$ are never late!
  \item $\tau_3$ never more than 6 units late!
\end{itemize}

\[ \tau_1 \quad \text{and} \quad \tau_2 \quad \text{are never late!} \]
\[ \tau_3 \quad \text{never more than 6 units late!} \]

\[ \text{Deadline} \quad \text{Miss} \quad \text{Completion} \]
Prior Work

- Can already determine *tardiness* bounds given system parameters
- Larger WCETs = larger bounds

\[
\begin{align*}
\tau_1 & \quad \tau_2 & \quad \tau_3 \\
\text{Release} & \quad \text{Deadline} & \quad \text{Deadline Miss} & \quad \text{Completion}
\end{align*}
\]
Can We Do Better?

- Obviously possible with optimal schedulers
- But can we do so without the disadvantages of those schedulers?
- Yes.
Priority Points

- Deadlines serve **both** to determine scheduler priorities **and** to specify when a job should be complete.
- We separate out these ideas (concept of **priority point** from Leontyev and Anderson 2007).
Priority Points

- Deadlines serve **both** to determine scheduler priorities **and** to specify when a job should be complete.

- We separate out these ideas (concept of priority point from Leontyev and Anderson 2007).

```
Actual deadline determines priority (old analysis)
```

```
<table>
<thead>
<tr>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Original Deadline</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Original Bound</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
```
Priority Points

- Deadlines serve both to determine scheduler priorities and to specify when a job should be complete.
- We separate out these ideas (concept of priority point from Leontyev and Anderson 2007).

Use a shorter “deadline” as priority point. Plug into old analysis.
Priority Points

- Deadlines serve **both** to determine scheduler priorities **and** to specify when a job should be complete.
- We separate out these ideas (concept of **priority point** from Leontyev and Anderson 2007).

Use that response time to compute new lateness bound.
Reducing Priority Points

- Using an earlier priority point improves the bound for that task at the expense of other tasks.
Reducing Priority Points

- Using an earlier priority point improves the bound for that task at the expense of other tasks.
Reducing Priority Points

- Using an earlier priority point improves the bound for that task at the expense of other tasks.

\[
\text{Lateness Bound} \quad \tau_1, \tau_2, \tau_3, \tau_4, \tau_5
\]
Best Assignment

• What is the “best” assignment?
• Our metric: minimize the maximum lateness bound.
• Optimal solution happens when all tasks have the same bound.
Fair Lateness

- Optimal solution = **fair lateness**.
- Scheduler = **Global Fair Lateness (G-FL)**
G-FL Implementation

- G-FL is G-EDF-like.
- Can use existing arbitrary deadline G-EDF scheduler with “fake deadlines.”

<table>
<thead>
<tr>
<th>Relative Priority Point</th>
<th>G-EDF</th>
<th>G-FL</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D_i$</td>
<td></td>
<td>$D_i - \frac{m-1}{m} C_i$</td>
</tr>
</tbody>
</table>

\[ D_i \]
G-FL Implementation

- G-FL is G-EDF-like.
- Can use existing arbitrary deadline G-EDF scheduler with “fake deadlines.”

**Relative Priority Point**

<table>
<thead>
<tr>
<th></th>
<th>G-EDF</th>
<th>G-FL</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D_i$</td>
<td></td>
<td>$D_i - \frac{m-1}{m} C_i$</td>
</tr>
</tbody>
</table>

where $m$ is the number of tasks.
Why Does it Work?

- Due to limited time, only giving intuition here.
- G-EDF this slide.
Why Does it Work?

- Due to limited time, only giving intuition here.
- G-EDF this slide.

Incorrect decision - scheduler only accounts for urgency, not length.
G-FL Schedule

- \( \tau_1 \)
- \( \tau_2 \)
- \( \tau_3 \)

- Release
- Deadline
- Completion
- Deadline Miss
- Priority Point
- CPU 1
- CPU 2

July 11, 2012
G-FL Schedule

- Release
- Completion
- Deadline
- Priority Point

- CPU 1
- CPU 2
G-FL Schedule

- τ₁: Release, Completion
- τ₂: Release, Deadline
- τ₃: Release, Deadline, Priority Point

Legend:
- CPU 1
- CPU 2

- Deadline
- Deadline Miss
- Priority Point
G-FL Schedule

\[ \tau_1 \]

\[ \tau_2 \]

\[ \tau_3 \]

Release  \quad \text{Deadline}  \quad \text{Deadline Miss}

Completion  \quad \text{Priority Point}  \quad \text{CPU 1}

\quad  \quad \text{CPU 2}
G-FL Schedule

\[ \tau_1 \]
\[ \tau_2 \]
\[ \tau_3 \]

- Release
- Deadline
- Priority Point
- Completion
- Deadline Miss

CPU 1
CPU 2
G-FL Schedule

\begin{align*}
\tau_1 & \quad \text{Release} \\
\tau_2 & \quad \text{Release} \\
\tau_3 & \quad \text{Deadline Miss}
\end{align*}

\begin{align*}
\text{Deadline} & \quad \text{Priority Point} \\
\text{Completion} & \quad \text{CPU 1} \\
\text{Deadline Miss} & \quad \text{CPU 2}
\end{align*}
G-FL Schedule

- τ₁
- τ₂
- τ₃

↑ Release    ↓ Deadline    ↓ Priority Point
↑ Completion  ↓ Deadline Miss

CPU 1
CPU 2
G-FL Schedule

- \( \tau_1 \)
- \( \tau_2 \)
- \( \tau_3 \)

- Release
- Completion
- Deadline
- Priority Point
- Deadline Miss

CPU 1
CPU 2
G-FL Schedule

τ₁

τ₂

τ₃

Release
Deadline
Completion
Deadline Miss

Priority Point

CPU 1

CPU 2
G-FL Schedule

τ₁  τ₂  τ₃

Release  Deadline  Priority Point
Completion  Deadline Miss

CPU 1  CPU 2
Experiments - Bounds

![Bar chart showing utilization distribution improvement with different period distributions and CPU counts.](max_relative_bounds.eps)

- Utilization Distribution:
  - Light-Uniform
  - Medium-Uniform
  - Heavy-Uniform
  - Light-Bimodal
  - Medium-Bimodal
  - Heavy-Bimodal

- Improvement (%)

- Period Distribution, CPU Count:
  - Short, m=2
  - Short, m=4
  - Short, m=6
  - Moderate, m=2
  - Moderate, m=4
  - Moderate, m=6
  - Long, m=2
  - Long, m=4
  - Long, m=6
  - Long, m=8
Experiments - Bounds

![Graph showing utilization distribution improvement vs period distribution and CPU count. The graph compares different utilization distributions (Light-Uniform, Medium-Uniform, Heavy-Uniform, Light-Bimodal, Medium-Bimodal, Heavy-Bimodal) across various period distributions and CPU counts (Short, Short, Short, Moderate, Moderate, Moderate, Long, Long, Long) for different CPU counts (m=2, m=4, m=6, m=8).]
Experiments - Bounds

![Bar Chart: Utilization Distribution]

- Light-Uniform
- Medium-Uniform
- Heavy-Uniform
- Light-Bimodal
- Medium-Bimodal
- Heavy-Bimodal

**Period Distribution, CPU Count**

**Improvement (%)**

- Short, m=2
- Short, m=4
- Short, m=6
- Moderate, m=2
- Moderate, m=4
- Moderate, m=6
- Long, m=2
- Long, m=4
- Long, m=6
- Long, m=8
Experiments - Bounds

![Graph showing utilization distribution and improvement percentage]
Experiments - Bounds
Experiments - Bounds

![Graph showing Utilization Distribution with different period distributions and CPU counts]
Experiments – Computed Schedules

Utilization Distribution
- Light-Uniform
- Medium-Uniform
- Heavy-Uniform
- Light-Bimodal
- Medium-Bimodal
- Heavy-Bimodal

Improvement (%)

Period Distribution, CPU Count
- Short, m=2
- Short, m=4
- Short, m=6
- Moderate, m=2
- Moderate, m=4
- Moderate, m=6
- Long, m=2
- Long, m=4
- Long, m=6
- Long, m=8
Conclusion

- Tardiness bounds can be reduced by about 50% by switching from G-EDF to G-FL.
- Actual tardiness also likely to be lower.
- Remember: implementation still like G-EDF!
Future Work

- HRT scheduling efficiency. (Related: see Back, Chwa, and Shin, RTAS 2012)

- Other notions of “fair lateness” - e.g. same percentage of period length instead of absolute lateness.
Questions?

Thank You!