The Global Limited Preemptive Earliest Deadline First Feasibility of Sporadic Real-time Tasks

Abhilash Thekkilakattil, Sanjoy Baruah, Radu Dobrin and Sasikumar Punnekkat
Motivation

- Multi (-core) processors in real-time systems complicate the problems associated with fully preemptive schedulers
  - Complex hardware, e.g., different levels of caches
    - Difficult to perform timing analysis
  - Potentially large number of task migrations: implementation issues
    - Difficult to demonstrate predictability
    - Difficult to reason about safety

- Non-preemptive scheduling can be infeasible at arbitrarily small utilization
  - Long task problem: at least one task has execution time greater than the shortest deadline

Solution: Limit preemptions
Advantages of limiting preemptions

Combines best of preemptive and non-preemptive scheduling

- Control preemption related overheads
  - Context switch costs, cache related preemption delays, pipeline delays and bus contention costs

- Improve processor utilization
  - Reduce preemption related costs while eliminating infeasibility due to blocking

Anecdotal evidence: “limiting preemptions improves safety and makes it easier to certify software for safety-critical applications”
## Limited preemptive scheduling landscape

<table>
<thead>
<tr>
<th>Uniprocessor</th>
<th>Limited preemptive FPS (Yao et al., RTSJ’11)</th>
<th>Limited preemptive EDF (Baruah, ECRTS’05)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiprocessor</td>
<td>Global limited preemptive FPS (Marinho et al., RTSS’13)</td>
<td>?</td>
</tr>
</tbody>
</table>
G-LP-EDF scheduling model

Processor 1

Processor 2

Low priority job executions

High priority job executions

high priority job release

job with latest deadline

$t$

$t'$
Main contributions

1. Schedulability analysis for **Global Limited Preemptive Earliest Deadline First** (G-LP-EDF) scheduling of **sporadic** real-time tasks

2. Analysis of the effects of increasing the **processor speed** on G-LP-EDF feasibility
Methodology overview

A necessary schedulability condition:

Upper-bound on the work generated under G-LP-EDF ≤ Lower-bound on the work executed under any work conserving algorithm
Methodology overview

A sufficient schedulability condition:

Upper-bound on the work generated under G-LP-EDF

≤

Lower-bound on the work executed under any work conserving algorithm
Lower bound on the work done

\[
\text{work done}(t_i, t_{i-1}) \geq m(t_i, t, x_i) + x_i
\]
Lower bound on the work done

Unschedulability scenario:

\[ x_i < (t_{i+1} - t_i)s \]

\[ x_1 < (t_0 - t_1 - L)s \]

work done\((t_i, t_{i-1})\) ≥ \(m(t_{i+1} - t_i - x_i) + x_i\)

work done\((t_k, t_0)\) > \((m (m - 1)s(t_0 - t_k - L) + mL\)

- Low priority job executions (blocking)
- High priority job executions
Methodology overview

A sufficient schedulability condition:

\[ \text{Upper-bound on the work generated under G-LP-EDF} \leq \text{Lower-bound on the work executed under any work conserving algorithm} \]
Upper bound on the work generated

….. under G-LP-EDF

In \([t_k, t_0)\), we consider the duration for which:

a. Low priority tasks block high priority tasks
b. Higher priority tasks execute
Maximum duration of blocking

blocking(t_k, t_0) : mL

- Low priority job executions (blocking)
- High priority job executions
Interference from higher priority tasks

Work done\((t_k, t_0)\) \leq \sum_{i} FF DBF_i(t_0 - t_k, s) + mL

Executes on a speed 's' processor

Task i

\[
FF DBF_i(t, s) = head_i + tail_i
\]

(Forced-Forward Demand Bound Function)

\[
\frac{t}{T_i} C_i \quad R \quad D_i \quad D \quad \frac{C_i}{s} \quad R < D_i \quad otherwise
\]

\[
C_i \quad C_i \quad (D_i \quad R) s \quad 0
\]
Methodology overview

A sufficient schedulability condition:

\[
FF \ DBF_i(t_0, t_k, s) + mL \leq (m - (m - 1)s)(t_0 - t_k - L) + mL
\]

Upper-bound on the work generated under G-LP-EDF

Lower-bound on the work executed under any work conserving algorithm
1. Schedulability analysis for Global Limited Preemptive Earliest Deadline First (G-LP-EDF) scheduling of sporadic real-time tasks

2. Analysis of the effects of increasing the processor speed on G-LP-EDF feasibility
1. Schedulability analysis for Global Limited Preemptive Earliest Deadline First (G-LP-EDF) scheduling of sporadic real-time tasks

2. Analysis of the effects of increasing the processor speed on G-LP-EDF feasibility
Feasibility bucket

- Task sets feasible at speed $s=1$
- Task sets feasible at speed $s=x$
- Task sets feasible at speed $s=1$

Speed (faster)

Speed (slower)
Long task problem

Non-preemptive infeasibility arising from at least one task having WCET greater than shortest deadline

A solution: code-refactoring at task level

Our speed-up factor quantifies the extent to which code-refactoring must be done to enable non-preemptive feasibility

(Short, 2010, “The case for non-preemptive, deadline-driven scheduling in real-time embedded systems”)
1. Schedulability analysis for Global Limited Preemptive Earliest Deadline First (G-LP-EDF) scheduling of sporadic real-time tasks

2. Analyze the effects of increasing the processor speed on G-LP-EDF feasibility
Conclusions

• Global limited preemptive EDF feasibility analysis
  – To control preemption related overheads
  – Enables better reasoning about predictability of multi (-core) processor real-time systems

• Processor speed vs. preemptive behavior
  – Quantifies the extent to which code-refactoring must be performed to address the long task problem
  – Sub-optimality of G-NP-EDF
Future work

- Compare G-LP-EDF and G-P-EDF in presence of overheads
- Perform trade-offs: number of extra processors vs. speed-up
- Partition tasks comprising of non-preemptive chunks
- Accounting for suspensions
Thank you!

Questions?