Multiprocessor Fixed Priority Scheduling with Limited Preemptions

Abhilash Thekkilakattil, Rob Davis, Radu Dobrin, Sasikumar Punnekkat and Marko Bertogna
Motivation

- Preemptive scheduling on multi (-core) processors introduces new challenges
  - Complex hardware, e.g., different levels of caches
    - Difficult to perform timing analysis
  - Potentially large number of task migrations
    - 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

One solution: limit preemptions
System Model

Identical multiprocessor platform with $m$ processors

<table>
<thead>
<tr>
<th>Job 1</th>
<th>Job 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Release time</td>
<td>Fixed Preemption Points</td>
</tr>
<tr>
<td>Minimum inter-arrival time (period)</td>
<td>Non-Preemptive Region (NPR) = WCET</td>
</tr>
</tbody>
</table>

Relative Deadline

$\sum$
Limited Preemptive Scheduling

Combines **best of** preemptive and non-preemptive scheduling

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

| Uniprocessor | Limited preemptive FPS  
(Burns’94, Bril et al., RTSJ’09, Yao et al., RTSJ’11) | Limited preemptive EDF  
(Baruah, ECRTS’05) |
|--------------|---------------------------------------------------|-----------------|
| Multiprocessor | **Global limited preemptive FPS**  
(Block et al., RTCSA’07, Marinho et al., RTSS’13, Davis et al., TECS’15) | **Global limited preemptive EDF**  
(Block et al., RTCSA’07, Thekkilakattil et al., ECRRTS’14, Chattopadhyay and Baruah, RTNS’14) |

… of course the references are by no way exhaustive!
Managing Preemptions in Global Limited Preemptive Scheduling

Lazy Preemption Approach

Processor 1
- High priority
- Medium priority

Processor 2
- High priority
- Low priority
Managing Preemptions in Global Limited Preemptive Scheduling

Processor 1

High priority

Medium priority

blocking

Processor 2

Low priority

Eager Preemption Approach
Global Limited Preemptive FPS with Fixed Preemption Points

<table>
<thead>
<tr>
<th>Lazy Preemption Approach</th>
<th>Block <em>et al.</em>, RTCSA’07: Link Based Scheduling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eager Preemption Approach</td>
<td></td>
</tr>
</tbody>
</table>
Lazy Preemption Approach: Link Based Scheduling

- Developed in the context of resource sharing by Block et al., RTCSA’07
  - Applicable to limited preemptive scheduling

- Implements lazy preemption approach

- Higher priority tasks blocked on a processor is linked to that processor

- Analyzable using a simple and generic inflation based test (Brandenburg and Anderson, MPI-Tech Report’14)
  1) Inflate WCET with largest blocking factor
  2) Determine schedulability using any standard test e.g., response time analysis for global preemptive FPS
### Global Limited Preemptive FPS with Fixed Preemption Points

<table>
<thead>
<tr>
<th><strong>Lazy Preemption Approach</strong></th>
<th>Block <em>et al.</em>, RTCSA’07: Link Based Scheduling</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Eager Preemption Approach</strong></td>
<td>No significant work!</td>
</tr>
</tbody>
</table>

How can we perform **schedulability** analysis of tasks scheduled using G-LP-FPS with eager preemptions?
Schedulability Analysis under G-LP-FPS with Eager Preemptions

Interference (higher and lower priority)

Task $i$
Schedulability Analysis under G-LP-FPS with Eager Preemptions

Interference (higher and lower priority)

• Case 1: no “push through” blocking
• Case 2: presence of “push through” blocking
Schedulability Analysis under G-LP-FPS with Eager Preemptions

Interference
(higher and lower priority)

• Case 1: no “push through” blocking
• Case 2: presence of “push through” blocking
Lower Priority Interference before Task Start Time

Case 1: no push through blocking

Processor 1
Task \( i \) (high)
Medium priority

Processor 2
Low priority

blocking = sum of \( m \) largest (\{lower priority NPRs\})
Schedulability Analysis under G-LP-FPS with Eager Preemptions

- Case 1: no “push through” blocking
- Case 2: presence of “push through” blocking
Lower Priority Interference before Task Start Time

Case 2: presence of push through blocking

blocking = \text{sum of } m \text{ largest } \{\text{lower priority NPRs, final NPR of } i\}$
Schedulability Analysis under G-LP-FPS with Eager Preemptions

Interference (higher and lower priority)
Schedulability Analysis under G-LP-FPS with Eager Preemptions

Interference (higher and lower priority)

Task $i$
**Lower Priority Interference after Task Start Time**

**Diagram Description:**

Processor 1:
- High priority task
- Task \( i \) blocked
- Number of processors executing a lower priority NPR \( \leq (m-1) \)

Processor 2:
- Low priority

**Equation:**

blocking = sum of \((m-1)\) largest (\{lower priority NPRs\})
Schedulability Analysis under G-LP-FPS with Eager Preemptions

\[ R_i = \text{Interference (higher and lower priority)} + \text{Interference (higher and lower priority)} + \text{Interference (higher and lower priority)} + \text{Interference (higher and lower priority)} + C_i \]

Of course, preemption may not occur at all preemption points
- No. of preemptions as a function of response time to reduce pessimism
- Details in the paper
Experiments

Which among eager and lazy preemption approaches is better for Global Limited Preemptive FPS (G-LP-FPS)?

• Compared schedulability under **eager preemptions** and **lazy preemptions**

  • Test for lazy preemptions: test for **link-based scheduling** that implements lazy preemptions
    - Inflate task execution time with largest blocking time
    - Perform response time analysis for G-P-FPS
Overview of Experiments

- Task utilizations generated using UUnifastDiscard
- Periods in the range 50 to 500
- Taskset utilization in the range 2.4 to m

- We investigated how weighted schedulability varies with:
  1. Varying number of tasks
  2. Varying number of processors
  3. Varying NPR lengths
     a. relatively large NPR w.r.t task WCETs
     b. relatively small NPR w.r.t task WCETs
Weighted Schedulability

- Weighs schedulability with utilization (Bastoni et al., OSPERT’10)

\[ W(p) = \frac{\sum_{\forall \Gamma} U(\Gamma)S(\Gamma, p)}{\sum_{\forall \Gamma} U(\Gamma)} \]
Weighted Schedulability

- Weighs schedulability with utilization (Bastoni et al., OSPERT’10)

\[ W(p) = \frac{\sum_{\forall \Gamma} U(\Gamma) S(\Gamma, p)}{\sum_{\forall \Gamma} U(\Gamma)} \]

Schedulability of taskset \( \Gamma \) w.r.t parameter \( p \)
Weighted Schedulability

- Weighs schedulability with utilization (Bastoni et al., OSPERT’10)

\[
W(p) = \frac{\sum_{\forall \Gamma} U(\Gamma) S(\Gamma, p)}{\sum_{\forall \Gamma} U(\Gamma)}
\]

Utilization of taskset $\Gamma$
Weighted Schedulability

• Weighs schedulability with utilization (Bastoni et al., OSPERT’10)

\[ W(p) = \frac{\sum_{\forall \Gamma} U(\Gamma)S(\Gamma, p)}{\sum_{\forall \Gamma} U(\Gamma)} \]

• Enables investigation of schedulability w.r.t a second parameter in addition to utilization

• Higher weighted schedulability implies a better algorithm with respect to scheduling high utilization tasksets (and thus better algorithm w.r.t efficiency)
Experiments

We investigated how *weighted schedulability* varies with:

1. Varying number of tasks
2. Varying number of processors
3. Varying NPR lengths
   a. relatively large NPR *w.r.t* task WCETs
   b. relatively small NPR *w.r.t* task WCETs
Varying Number of Tasks

m=4 and NPR=5%

Eager approach outperforms lazy approach for larger number of tasks
Experiments

We investigated how \textit{weighted schedulability} varied with:

1. Varying number of tasks
2. Varying number of processors
3. Varying NPR lengths
   a. relatively large NPR \textit{w.r.t} task WCETs
   b. relatively small NPR \textit{w.r.t} task WCETs
Varying Number of Processors

n=30 and NPR=5%

Higher utilization and fixed $n$ $\Rightarrow$ large execution times $\Rightarrow$ large NPRs

$\Rightarrow$ more blocking after start time

Lazy preemptions

Eager preemptions

Weighted Schedulability

No. of processors
Experiments

We investigated how *weighted schedulability* varied with:

1. Varying number of tasks
2. Varying number of processors
3. Varying NPR lengths
   a. relatively large NPR *w.r.t* task WCETs
   b. relatively small NPR *w.r.t* task WCETs
Varying Lengths of NPRs (large)

$n=30$ and $m=4$

- G-P-FPS
- EPA
- LPA
- EPA Only
- G-NP-FPS

Number of preemptions:
- Number of preemptions = 3
- Number of preemptions = 2
- Number of preemptions = 1
- Number of preemptions ≈ 1
- Number of preemptions = 0

Eager preemptions

Lazy preemptions

Largest NPR length as % of WCET
Experiments

We investigated how *weighted schedulability* varied with:

1. Varying number of tasks
2. Varying number of processors
3. Varying NPR lengths
   a. relatively large NPR *w.r.t* task WCETs
   b. relatively small NPR *w.r.t* task WCETs
Varying Lengths of NPRs (small)

n=30 and m=4

Lazy approach outperforms eager approach for smaller NPR lengths

Small NPR lengths ➔ many preemption points ➔ more blocking

Lazy preemptions

Eager preemptions

Weighted Schedulability

Largest NPR length as % of WCET
Conclusions

• Presented a schedulability test for global LP FPS with eager preemptions

• Compared eager and lazy approaches using synthetically generated tasksets
  – Eager approach outperforms lazy approach

• Eager preemption is beneficial if high priority tasks have short deadlines relative to their WCETs
  – Need to schedule them ASAP

• Lazy preemption is beneficial if tasks have many preemptions points
  – Need to reduce blocking occurring after tasks start their execution
Future Work

- Evaluation of runtime preemptive behaviors of eager and lazy approaches under global EDF and FPS
  - LP scheduling with eager approach generates more runtime preemptions compared to preemptive scheduling (under submission to RTAS’16)

- Evaluation on a real hardware
  - Context Switch Overheads
  - Cache related preemptions delays

- Efficient preemption point placement strategies for multiprocessor systems
Thank you!

Questions?