Limited Preemptive Real-time Scheduling on Multiprocessors

Abhilash Thekkilakattil
Mälardalen University - Sweden

Västerås ("Westeros")
Stockholm
Västerås
Why am I here?
Why am I here?
Limited Preemptive Real-time Scheduling on Multiprocessors

Abhilash Thekkilakattil

MÄLARDALEN UNIVERSITY SWEDEN
Limited Preemptive Real-time Scheduling on Multiprocessors

Abhilash Thekkilakattil
We do Scheduling Everyday!

- **Priority based scheduling**
  - Call from school: sick kid
  - Drop kid at home
  - Work
  - Work

- **Deadline based scheduling**
  - Go to tax office
  - Shopping
  - Closing time of tax office
  - Closing time of grocery store
Limited Preemptive Real-time Scheduling on Multiprocessors

Abhilash Thekkil
Real-time Systems for the Uninitiated

Computer programs called “tasks”

Detect the crash  Evaluate its intensity  Inflate the airbags
Real-time Systems for the Uninitiated

Correctness of the system depends on both:

1. Correctness of the decision to inflate the airbags
2. Correctness of the time at which the decision is implemented
Real-time Systems for the Uninitiated

• Fewer computers than the number of tasks!
• Need to schedule the tasks such that deadlines are not missed
• For example tasks are typically executed highest priority first or earliest deadline first
• Schedulability test determines if a taskset meets deadlines under a given scheduling algorithm
Real-time Tasks

Periodic and sporadic event occurrences

Timing characteristics of the physical processes

Dependability requirements

int task_braking() {
    ....
    ....
}

Real-time Tasks

Preemptive scheduling

Non-Preemptive scheduling
Limited Preemptive Real-time Scheduling on Multiprocessors

Abhilash Thekkilakattil

Mälardalen University
Sweden
Limited Preemptive Scheduling

Fully Preemptive Scheduling

3 preemptions (no. of high priority releases)

Fixed Preemption Points Scheduling

No more than 2 preemptions (no. of PPs)
Faster Computers vs. Power Demand

Multicore Revolution!

Increase the number of processors instead of increasing processor speeds.

Power density (W/cm²)

- 1
- 10
- 100
- 1000

- Nuclear reactor


- 486
- 386
- 286
- 8085
- 8080
- 8008
- 4004

Intel cancelled Pentium Thejas!

(adapted from Marko Bertogna’s PhD thesis)
Processor to Memory Performance Gap

Hardware features to bridge this performance gap
e.g., caches

Predictability challenge!
Processor to Memory Performance Gap

Task `detect_crash`

Wait for the data (overhead)

Fetch data request → Data → Memory
Processor to Memory Performance Gap

- Smaller overhead $\rightarrow$ shorter execution time

**Task**: detect_crash

**Fetch data request**

**Memory**
Bridging the Processor to Memory Performance Gap

**Task brake**

**Task detect Crash**

Wait for the data is **negligible**

**Cache memory**

**Alternative 1**: build predictable hardware e.g., PRETS

**Alternative 2**: efficient resource allocation & overhead accounting
Preemption Related Overheads

- Cache related preemption and migration delays
  - Overhead involved in reloading cache lines
  - Vary with the point of preemption and increase bus contention

- Context switch related overheads
  - Overhead involved in saving and retrieving the context of the preempted task

- Pipeline related overheads
  - Clear the pipeline upon preemption
  - Refill the pipeline upon resumption
Preemption Related Overheads

- Cache related preemption and migration delays
  - Overhead involved in reloading cache lines
  - Vary with the point of preemption and increase bus contention

- Context switch related overheads
  - Overhead involved in saving and retrieving the context of the preempted task

- Pipeline related overheads
  - Clear the pipeline upon preemption
  - Refill the pipeline upon resumption

Alternative: don’t allow preemptions!

Hi! Hello Donald, I will call you after I finish this book!
Motivation

Preemptive Scheduling

- 🎁 Zero blocking: high schedulable utilization
- 😞 High runtime overhead: preemption costs
- 😞 Difficult to perform timing analysis
- 😞 Difficult to demonstrate predictability and safety: multicores

Non-Preemptive Scheduling

- 🎁 Low runtime overhead: zero preemption costs
- 🎁 Relatively easier to perform timing analysis
- 🎁 Preferred by many safety standards
- 😞 Increased blocking: low schedulable utilization

Solution: Limited-Preemptive Scheduling

- 🎁 Best of preemptive and non-preemptive: preempt only when necessary
Limited Preemptive Scheduling Models

- Preemption Threshold Scheduling (ThreadX, Wang and Saksena, 1999)
- Fixed Preemption Point Scheduling (Burns, 1994, Bril et al., 2009, Yao et al., 2011)
- Floating Non-Preemptive Region Scheduling (Baruah, 2005)

Our focus
Fixed Preemption Points Scheduling

- Minimum inter-arrival time (period)
- Relative Deadline
- Non-Preemptive Region (NPR) = WCET

Hi Donald, give me some time. Let me just finish this chapter!
Floating Non-Preemptive Regions Scheduling

Task $j$ (high priority)

Floating NPR (time duration)

Task $i$ (low priority)

Hi! Donald, give me some time. Let me just finish 5 more pages!

Hi!
## Limited Preemptive Scheduling Landscape

<table>
<thead>
<tr>
<th></th>
<th>Fixed priority based scheduling</th>
<th>Deadline based scheduling</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Uniprocessor</strong></td>
<td>Limited Preemptive FPS</td>
<td>Limited Preemptive EDF</td>
</tr>
<tr>
<td><strong>Multiprocessor</strong></td>
<td>Global Limited Preemptive FPS</td>
<td>Global Limited Preemptive EDF</td>
</tr>
</tbody>
</table>
Limited Preemptive Scheduling Landscape

<table>
<thead>
<tr>
<th></th>
<th>Fixed priority based scheduling</th>
<th>Deadline based scheduling</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Uniprocessor</strong></td>
<td>Limited Preemptive FPS</td>
<td>Limited Preemptive EDF</td>
</tr>
<tr>
<td><strong>Multiprocessor</strong></td>
<td>Global Limited Preemptive FPS</td>
<td>Global Limited Preemptive EDF</td>
</tr>
</tbody>
</table>

Handling preemptions in global LP scheduling:

1. **Eager** Preemption Approach
2. **Lazy** Preemption Approach
Global Limited Preemptive Scheduling

Lazy Preemption Approach: wait to preempt the lowest priority task
Global Limited Preemptive Scheduling

**Eager Preemption Approach:** preempt the *lower* priority task that first finishes its NPR (not necessarily the lowest)

Processor 1

Processor 2
Incomparability of Eager and Lazy Preemption Approaches

There are tasksets schedulable under an eager approach that are unschedulable under the lazy approach and vice-versa!
## Limited Preemptive Scheduling Landscape

<table>
<thead>
<tr>
<th></th>
<th>Fixed priority based scheduling</th>
<th>Deadline based scheduling</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Uniprocessor</strong></td>
<td>Limited Preemptive FPS Burns’94, Bril et al., RTSJ’09, Yao et al., RTSJ’11</td>
<td>Limited Preemptive EDF Baruah, ECRTS’05</td>
</tr>
<tr>
<td><strong>Multiprocessor</strong></td>
<td>Global Limited Preemptive FPS Marinho et al., RTSS’13, Block et al., RTCSA’07</td>
<td>Global Limited Preemptive EDF Block et al., RTCSA’07</td>
</tr>
</tbody>
</table>

Limited preemptive scheduling models:

1. Fixed Preemption Points
2. Floating Non-Preemptive Regions

Disclaimer: references are not exhaustive
Global Limited Preemptive EDF

<table>
<thead>
<tr>
<th>Fixed Preemption Points</th>
<th>Block et al., RTCSA’07 (link based scheduling)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Floating Non-Preemptive Regions</td>
<td>No significant work!</td>
</tr>
</tbody>
</table>

Question 1: How to determine the schedulability of real-time tasks under G-LP-EDF with floating NPRs?
G-LP-EDF Scheduling Model

Floating NPRs and Lazy Preemption Approach

Processor 1

high priority job release

Processor 2

Low priority job executions

High priority job executions

job with latest deadline

35
Methodology Overview

A necessary unschedulability 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

\[ \sum_{\forall \tau_i} FF - DBF_i(t_0 - t_k, s) + mL \leq (m-(m-1)s)(t_0 - t_k - L) + mL \]

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

\[ \sum_{\forall \tau_i} FF - DBF_i(t_0 - t_k, s) \leq (m-(m-1)s)(t_0 - t_k - L) \]
Limited Preemptive Scheduling Landscape

<table>
<thead>
<tr>
<th></th>
<th>Fixed priority based scheduling</th>
<th>Deadline based scheduling</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Uniprocessor</strong></td>
<td>Limited Preemptive FPS</td>
<td>Limited Preemptive EDF</td>
</tr>
<tr>
<td></td>
<td>Burns’94, Bril et al., RTSJ’09,</td>
<td>Baruah, ECRTS’05</td>
</tr>
<tr>
<td></td>
<td>Yao et al., RTSJ’11</td>
<td></td>
</tr>
<tr>
<td><strong>Multiprocessor</strong></td>
<td>Global Limited Preemptive FPS</td>
<td>Global Limited Preemptive EDF</td>
</tr>
<tr>
<td></td>
<td>Marinho et al., RTSS’13, Block</td>
<td>Block et al., RTCSA’07</td>
</tr>
<tr>
<td></td>
<td>et al., RTCSA’07</td>
<td></td>
</tr>
</tbody>
</table>

Preemption approaches:

1. Eager Preemption Approach
2. Lazy Preemption Approach
Global Limited Preemptive FPS with Fixed Preemption Points

<table>
<thead>
<tr>
<th>Lazy Preemption Approach</th>
<th>Block et al., RTCSA'07 and Marinho et al., RTSS’13</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eager Preemption Approach</td>
<td>No significant work!</td>
</tr>
</tbody>
</table>

Question 2: How to determine the schedulability of real-time tasks under G-LP-FPS with eager preemptions?
G-LP-FPS Scheduling Model

Fixed Preemption Points and Eager Preemption Approach

Processor 1

High priority

Low priority task is executing even though the medium priority task is ready to execute

Processor 2

High priority

Medium priority

Low priority
We propose a method to calculate response time considering the lower priority interference before and after a task starts executing.

Method to calculate higher priority interference exists.
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 thesis
Experiments

• Compared schedulability under eager and lazy preemption approaches
  − Our test vs. existing test for link-based scheduling that implements lazy preemption approach

• 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
Schedulability under Large NPRs

- Schedulability varies with the number of preemptions.
  - Variations in blocking due to preemptions.

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

Eager preemptions:
- Number of preemptions = 3
- Number of preemptions = 2
- Number of preemptions = 1

Lazy preemptions:
- Number of preemptions = 0

Diagram shows weighted schedulability as a function of NPR length as a percentage of WCET.
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
Schedulability under Small NPRs

Lazy approach outperforms eager approach for smaller NPR lengths

Small NPR lengths → more preemption points → more blocking

Weighted Schedulability

Eager preemptions

Lazy preemptions

NPR length as % of WCET

0.5 1 1.5 2 2.5 3 3.5 4 4.5 5

0 0.03 0.06 0.09 0.12 0.15 0.18 0.21 0.24 0.27 0.3 0.33
## Limited Preemptive Scheduling Landscape

<table>
<thead>
<tr>
<th></th>
<th>Fixed priority based scheduling</th>
<th>Deadline based scheduling</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Uniprocessor</strong></td>
<td>Limited Preemptive FPS</td>
<td>Limited Preemptive EDF</td>
</tr>
<tr>
<td></td>
<td>Burns’94, Bril et al., RTSJ’09, Yao et al., RTSJ’11</td>
<td>(Baruah, ECRTS’05)</td>
</tr>
<tr>
<td><strong>Multiprocessor</strong></td>
<td>Global Limited Preemptive FPS</td>
<td>Global Limited Preemptive EDF</td>
</tr>
<tr>
<td></td>
<td>Marinho et al., RTSS’13, Block et al., RTCSA’07</td>
<td>Block et al., RTCSA’07</td>
</tr>
<tr>
<td></td>
<td>This talk: schedulability test for G-LP-FPS with eager preemption</td>
<td>This talk: schedulability test for G-LP-EDF with floating NPRs</td>
</tr>
</tbody>
</table>
### Question 3: No. of Preemptions

How does the no. of preemptions vary with the scheduling algorithm (EDF vs. FPS) and approach to preemption (eager vs. lazy) on multiprocessors?
RM vs EDF: Rise of the Multiprocessors

[Graph showing comparison between G-P-FPS and G-P-EDF for Weighted no. of preemptions vs NPR (% of WCET)]
RM vs EDF: Rise of the Multiprocessors

![Graph showing the comparison between G-P-FPS, EPA-FPS, and G-P-EDF in the context of weighted number of preemptions vs NPR (% of WCET).]
RM vs EDF: Rise of the Multiprocessors

![Graph showing weighted number of preemptions vs NPR (NPR = % of WCET)](image)

Legend:
- G-P-FPS
- EPA-FPS
- G-P-EDF
- EPA-EDF

NPR = % of WCET

51
RM vs EDF: Rise of the Multiprocessors

Similar Trends

Trends reverse
A Better Priority Assignment than RM?

RMPO : Higher frequency of release → higher the priority
D-C MPO : Smaller the value of D-C → higher the priority
A Better Priority Assignment than RM?

Weighted no. of preemptions

NPR Lengths as % of WCET

P-FPS (DM)
A Better Priority Assignment than RM?
A Better Priority Assignment than RM?

- P-FPS (DM)
- Eager FPS (DM)
- P-FPS (D-C)

Weighted no. of preemptions vs. NPR Lengths as % of WCET.
A Better Priority Assignment than RM?

- P-FPS (DM)
- Eager FPS (DM)
- P-FPS (D-C)
- Eager FPS (D-C)

Weighted no. of preemptions vs NPR Lengths as % of WCET graph.
A Better Priority Assignment than RM?

- P-FPS (DM)
- Eager FPS (DM)
- Lazy FPS (DM)
- P-FPS (D-C)
- Eager FPS (D-C)

Weighted no. of preemptions vs NPR Lengths as % of WCET
A Better Priority Assignment than RM?
Summary of Lessons Learned

1. Limited preemptive scheduling on multiprocessors may not reduce the number of preemptions
   • Number of preemptions larger than fully preemptive scheduling under the eager approach

2. The preemptive behavior of EDF on uniprocessors generalizes to multiprocessors
   • G-P-EDF generates fewer preemptions than G-P-FPS

3. This, however, does not generalize to global limited preemptive scheduling
   • G-LP-EDF generates more preemptions than G-LP-FPS

4. Schedulability gains from better priority assignment could be offset by increased number of preemptions
Limited Preemptive Scheduling Landscape

<table>
<thead>
<tr>
<th>Fixed priority based scheduling</th>
<th>Deadline based scheduling</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Uniprocessor</strong></td>
<td><strong>Limited Preemptive FPS</strong></td>
</tr>
<tr>
<td>Burns’94, Bril et al., RTSJ’09, Yao et al., RTSJ’11</td>
<td>Marinho et al., RTSS’13, Block et al., RTCSA’07</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Multiprocessor</strong></th>
<th><strong>Limited Preemptive EDF</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Block et al., RTCSA’07</td>
<td>Block et al., RTCSA’07</td>
</tr>
</tbody>
</table>

This talk: identifies runtime anomalies in the preemptive behavior under global limited preemptive scheduling
Sub-optimality of Uniprocessor Scheduling Algorithms

Question 4: How good is non-preemptive/limited preemptive scheduling when compared to uniprocessor optimal (preemptive) scheduling algorithms?
Processor speed vs. LP-EDF feasibility

Bounds on Uniprocessors

\[ 2 \max \left( 1, \frac{C_{\text{max}}}{D_{\text{min}}} \right) \left( 2 - \frac{1}{m} \right) \text{ G-NP-EDF feasibility} \]

\[ 2 \max \left( 1, \frac{L}{D_{\text{min}}} \right) \left( 2 - \frac{1}{m} \right) \text{ G-LP-EDF feasibility} \]

\[ \max \left( 1, \frac{L}{D_{\text{min}}} \right) \text{ uniprocessor LP-EDF feasibility} \]

Bounds on Multiprocessors

\[ 2 \max \left( 1, \frac{C_{\text{max}}}{D_{\text{min}}} \right) \left( 2 - \frac{1}{m} \right) \text{ G-NP-EDF feasibility} \]

\[ 2 \max \left( 1, \frac{L}{D_{\text{min}}} \right) \left( 2 - \frac{1}{m} \right) \text{ G-LP-EDF feasibility} \]

\[ \max \left( 1, \frac{L}{D_{\text{min}}} \right) \text{ uniprocessor LP-EDF feasibility} \]

\[ \left( 2 - \frac{1}{m} \right) \text{ G-P-EDF Feasibility (Phillips et al., 2002)} \]

1 \( m \) (and uni) processor feasibility
Advances in Sub-optimality of Uniprocessor Scheduling Algorithms

P-FPS feasible tasks (uniprocessor optimal)

NP-EDF feasible tasks

NP-FPS feasible tasks

Davis et al., RTSS’15

Davis et al., RTSS’15

NP-EDF feasible tasks

Davis et al., RTNS’10

Davis et al., RTSJ’15

Davis et al., RTNS’09

Liu and Layland, JACM’73

Davis et al., RTSJ’09

Thekkilakattil et al., ECRTS’13

Thekkilakattil et al., RTSJ’15

Abugchem et al., ESL’15

Nasri et al., RTNS’14

Nasri et al., RTNS’14

Davis et al., RTSS’15

Bruggen et al., ECRTS’15

Davis et al., RTNS’09

Davis et al., RTSJ’15

Davis et al., RTSS’15

Davis et al., RTNS’10

Davis et al., RTSJ’15
### Limited Preemptive Scheduling Landscape

<table>
<thead>
<tr>
<th>Uniprocessor</th>
<th>Multiprocessor</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Limited Preemptive FPS</strong></td>
<td><strong>Global Limited Preemptive FPS</strong></td>
</tr>
<tr>
<td>Burns’94, Bril et al., RTSJ’09, Yao et al., RTSJ’11</td>
<td>Marinho et al., RTSS’13, Block et al., RTCSA’07</td>
</tr>
<tr>
<td><strong>Limited Preemptive EDF</strong></td>
<td><strong>Global Limited Preemptive EDF</strong></td>
</tr>
<tr>
<td>(Baruah, ECRTS’05)</td>
<td>Block et al., RTCSA’07</td>
</tr>
</tbody>
</table>

This talk: the sub-optimality of LP scheduling
Conclusions

1) Limited preemptive scheduling on multiprocessors brings forth new **challenges**
   - Large design space
   - Counter-intuitive behaviors

2) Limited preemptive scheduling on multiprocessors brings forth new **opportunities**
   - Reduced overheads
   - Improved schedulability
   - Better overhead accounting strategies
Future Work

• Hybrid approach to manage preemptions on multicores to get best of eager and lazy approaches: integration with preemption thresholds

• Probabilistic schedulability analysis for tasks with fixed preemption points

• Improved preemption overhead accounting for tasks with fixed preemption points

• ….
Thank you!

Questions/Comments?