Gang FTP scheduling of periodic and parallel rigid real-time tasks

Vandy BERTEN    Joël GOOSENS

Université libre de Bruxelles

RTNS 2010
| 1 | Introduction               |
| 2 | Predictability            |
| 3 | Schedulability test       |
| 4 | Conclusions               |
Table of contents I

1 Introduction
   • Model
   • Predictability Problem

2 Predictability

3 Schedulability test

4 Conclusions
Introduction

Our objectives:

- Explore the theory of *parallel tasks*, and *Gang scheduling*
- Provide *schedulability tests* for various kinds of Gang schedulers

Why?

- Parallel tasks are coming on real-time/embedded systems (energy efficiency)
- Very few results in the literature
Task model

- \( n \) (rigid) parallel (\( v \)), periodic (\( T \)), constrained deadline (\( D \leq T \)) tasks
- \( m \) identical processors
Task model

- Preemptive
- Migrations
- Rigid (not malleable/moldable)
Gang FJP scheduler

1. Pick the highest priority job
2. If it fits, start it now
3. Start again with the remaining jobs
Schedulability test

Predictability

Schedulable for WCET $\Rightarrow$ Schedulable

+ Feasibility interval $[A, B]$

Feasibility Interval

Schedulable in $[A, B] \Rightarrow$ Schedulable forever

= Schedulability test

Schedulability test

Simulate the system in $[A, B]$ with WCET
Deadline miss

⇒ Gang FJP not predictable!

One of the problems: priority inversion (slack introduces new preemptions)
# Table of contents I

1. Introduction

2. Predictability
   - Possibilities
   - Parallel Monotonic
   - Idling FJP Scheduler
   - Limited Gang FJP Scheduler
   - Limited Slack Reclaiming

3. Schedulability test

4. Conclusions
Making the system predictable

We propose several solutions making the system predictable:

- Avoiding priority inversion
- Not using the slack
- Using the slack “smartly”

Two ways of doing so:

- Constraint the task system
- Constraint the scheduler
Parallel monotonic

Parallel monotonic FJP assignment

Larger job $\Rightarrow$ Lower priority

$\rightarrow$ High priority to small jobs
Parallel monotonic

We avoid priority inversion!

Theorem

Parallel Monotonic systems are predictable
Idling FJP scheduler

Just don’t use the slack!

- Still priority inversions, but same behavior as in the WCET case
- Not work conserving!

Theorem

Idling FJP schedulers are predictable
Limited Gang FJP scheduler

1. Pick the highest priority job
2. If it fits, start it now
3. *If it fitted in step 2*, start again with the remaining jobs

We avoid priority inversion!

**Theorem**

Limited Gang FJP schedulers are predictable
Limited Gang FJP scheduler

Limited Gang scheduler less efficient than “normal” Gang scheduler
Limited slack reclaiming

Gang FJP scheduler with *limited slack reclaiming*

1. While there is no slack, behave as for Gang FJP scheduler
2. Use the slack to run ahead jobs *narrower than the slack*

Still priority inversions, but no “problematic preemptions”

Theorem

Gang schedulers with limited slack reclaiming are predictable
Table of contents I

1 Introduction

2 Predictability

3 Schedulability test
   - Periodicity
   - Feasibility interval
   - Exact Schedulability Test

4 Conclusions
Periodicity of Gang FTP

Theorem

(Whatever) Gang FTP schedulers are periodic (using WCET):

- With a period $P \overset{\text{def}}{=} \text{lcm}\{T_1, \ldots, T_n\}$
- Starting from $S_n$, where:
  - $S_1 \overset{\text{def}}{=} O_1$
  - $S_i \overset{\text{def}}{=} \max\left\{O_i, O_i + \left\lceil \frac{S_{i-1} - O_i}{T_i} \right\rceil T_i \right\}$, $\forall i \in \{2, 3, \ldots, n\}$.

$\Rightarrow$ Same as non-Gang systems!

Going from *sequential* to *parallel jobs* did not change the periodicity
Periodicity

\[ O_1 = S_1 \]

\[ O_2 \]

\[ O_3 \]

\[ O_4 \]

\[ S_2 \]

\[ S_3 \]

\[ S_4 \]
Feasibility interval

Theorem

For any Gang FTP system (Parallel Monotonic, Idling scheduler, Limited Gang scheduler, Limited Slack reclaiming scheduler), we can use the following Feasibility interval:

\[ [0, S_n + P] \]
Exact schedulability test

Predictability

OK for Parallel Monotonic, Idling-, Limited Gang- and Limited Slack reclaiming scheduler

+ Feasibility Interval

Feasibility interval

\[ [0, S_n + P] \]

= Schedulability test

Schedulability test

Simulate the system in \([0, S_n + P]\) with WCET
Table of contents I

1. Introduction
2. Predictability
3. Schedulability test
4. Conclusions
   - Future work
   - Questions
Conclusions

- We strictly defined *rigid, moldable* and *maleable* recurent tasks
- We provided (and proved) an *exact schedulability test* for several kinds of FTP Gang schedulers
- We studied the *predictability* of those schedulers
Future work

- Extends our results to *moldable* and *maleable* tasks
- Sufficient RM-schedulability test for *sporadic* Gang scheduling
- ...
Questions?