9. Sustainable multiprocessor scheduling of sporadic task systems

(or: how an inexact schedulability test might be better)

Ted Baker
Florida State University, USA
Sanjoy Baruah
University of North Carolina, USA
@ ECRTS 2009

Overview

- Sustainability
  - What is it?
  - Why do I want it?
  - How do I get it?
- Sustainability of FTP (a.k.a FPS), EDF, EDZL applied to global multi-processor scheduling
- Sustainability of some global EDF tests

Schedulability test

- Will a workload W meet all deadlines if scheduled by algorithm A on platform P ?
- Exact test : "yes" $\Leftrightarrow$ W is schedulable
- Sufficient test : "yes" $\Rightarrow$ W is schedulable
- Goal
  - Validate timing properties on which correctness, reliability, safety depend

Weakness

- Real systems differ from models
  - Clock and timer inaccuracies
  - Blocking / non-preemption
  - Execution time variation
- Differences must not invalidate test (**!**)
- Solution
  - Use pessimistic assumptions

What is pessimistic?

- Should cover full range of possibilities
- How to validate over such a range?
- People tend to assume that system behavior is monotonic
  - So that they can rely on *boundary testing*
- This can be dangerous (**!**)
  - For example: does WCET of individual jobs give worst-case behavior of the entire system?

Goals

- Our notion of what is "pessimistic" should be consistent with the analysis that underlies the schedulability test
- The test should be monotonic with respect to any workload parameters that are considered to be bounds on actual values

Predictability [Ha & Liu, 1994]

- For arbitrary job sets on multiprocessors, if the scheduling algorithm is *work-conserving* [1], preemptive, global (with migration), with fixed job priorities
  - Job completion times are monotonically related to job execution times
- Hence it is safe to consider only upper bounds for job execution times in schedulability tests
- This is not true for non-preemptive scheduling

1) A scheduling algorithm is *work conserving* if processors are not idle while tasks eligible for execution are not able to execute on other processors

Sustainability [Baruah & Burns, 2006]

- Extends the notion of predictability to wider range of relaxations of workload parameters
  - Shorter execution times
  - Longer periods
  - Less release-time jitter
  - Later deadlines
- Relaxation should preserve schedulability
  - Much like predictability does for increase
- Cited work considers single-processor systems
- Here we consider multi-processors

Why to relax?

- Safety
  - Run-time behavior may fall short of pessimistic model parameters
- Modular design
  - Start with pessimistic parameters, relax as design evolves
- Efficiency
  - Revise estimates at job arrival, schedule more precisely

How?

- Actual/nominal?
  - Actual maximum job execution time < C_i, or
  - Value C_i used by scheduler is reduced
- Per job/task?
  - C_i reduced differently for each job, or
  - C_i reduced for all jobs of task
- Use in scheduling decisions, or not?
  - These may or may not make a difference in sustainability, depending on context

Getting specific

- Sustainability depends on
  - The platform
  - The workload model
  - The scheduling algorithm
- Platform : SMP
- Workload model : sporadic task set
- Scheduling algorithms : global FTP, EDF, EDZL

SMP platform

- M identical processors
- Bounded cost for migration, context switch

Sporadic [task workload] model

- Fixed set of tasks
- Each generates a sequence of jobs
- Each job of a task has parameters
  - Execution time
    - C_i is an upper bound
  - Inter-arrival time
    - T_i is a lower bound
  - Deadline
    - D_i, relative to arrival time

Where is pessimism?

- T_i : lower bound
- C_i : upper bound
- D_i : lower bound?
- Sustainability means allowing for the following relaxations
  - Longer inter-arrival times
  - Shorter execution times
  - Longer deadlines

Global scheduling

- Fixed Task Priority (FTP), a.k.a. FPS
  - *M* tasks with highest task priority allowed to execute
- EDF
  - *M* jobs with earliest deadlines allowed to execute
- EDZL
  - Same, except that jobs with zero laxity preempt
  - *Laxity = remaining-time-to-deadline – remaining-WCET*
- Common property
  - Limit of 1 (or 2, for EDZL) preemption per job allows us to bundle worst-case preemption/migration overhead into WCET of jobs

Longer inter-arrival times

- Sustainability is built into the sporadic workload model
  - Per-job or per-task
  - Actual, nominal too for algorithms that do not make use of T_i in scheduling decisions
- *Global FTP, EDF, EDZL are sustainable w.r.t. longer inter-arrival times*

Shorter execution times

- Depends on scheduling algorithm
- Studied by Ha & Liu (1994)
- Non-preemptive scheduling not sustainable
- Preemptive work-conserving fixed-job-priority SMP scheduling with migration is sustainable
  - *FTP and EDF are sustainable*
- Proved for job sets, covers sporadic tasks
  - Actual/nominal, per job/task

EDZL & shorter execution time

- Difference from FTP & EDF
  - Job priority can change
    - Breaks Ha & Liu assumption
- *EDZL is sustainable*
  - Piao, Han, Kim et al., 2006 (new proof in this work!)
    - Per job/task
    - Nominal and actual, or just actual

Longer deadlines?

- FTP is sustainable for sporadic tasks
- EDF & EDZL *sustainability open* for sporadic tasks
  - Not sustainable for job sets*
  - Exact test (if one exists) might be risky
  - Or scheduler should ignore deadline changes
  - Sustainable tests are of interest

* Apparently also not sustainable for sporadic task sets, with respect to relaxation of a single job deadline (see appendix). However, it might be sustainable w.r.t. uniform relaxation of deadline over all jobs of a task

Sustainable schedulability test

- If the test says a task set is schedulable, then it remains schedulable if we reduce pessimism in our assumptions, *e.g.,*
  - Decrease task execution times
  - Increase task periods
  - Reduce release-time jitter
  - Lengthen job deadlines

Relationships

- Algorithm sustainable ⇒ Test sustainable
- Algorithm <u>not</u> sustainable ⇒ Exact test <u>not</u> sustainable
- So an inexact test might be better, *if it is sustainable and the algorithm is not*

This might be true for global EDF
Do we know any sustainable tests?

Self-sustainable test (new)

- If the test says a task set is schedulable, then the test still says it is schedulable if we reduce pessimism in our assumptions, *e.g.*
  - Decrease task execution times
  - Increase task periods
  - Reduce release-time jitter
  - Lengthen job deadlines

Do we know any self-sustainable tests for global EDF?

Summary

- Sustainability depends on workload model as well as platform and scheduling algorithm
- For sporadic tasks and SMP
  - Global FTP is fully sustainable
  - Global EDF & EDZL are (only?) exec. time and inter-arrival time sustainable
- Self-sustainability is a strong form of sustainability for tests
  - Not all tests are deadline self-sustainable
  - But at least one is