Limited-Preemption Scheduling on Multiprocessors

Bipasa Chattopadhyay   Sanjoy Baruah
Department of Computer Science
The University of North Carolina at Chapel Hill

Outline

- Motivation
- Task Model
- Our Contribution
- Application to Multi-GPU systems
- Experimental Evaluation
- Summary

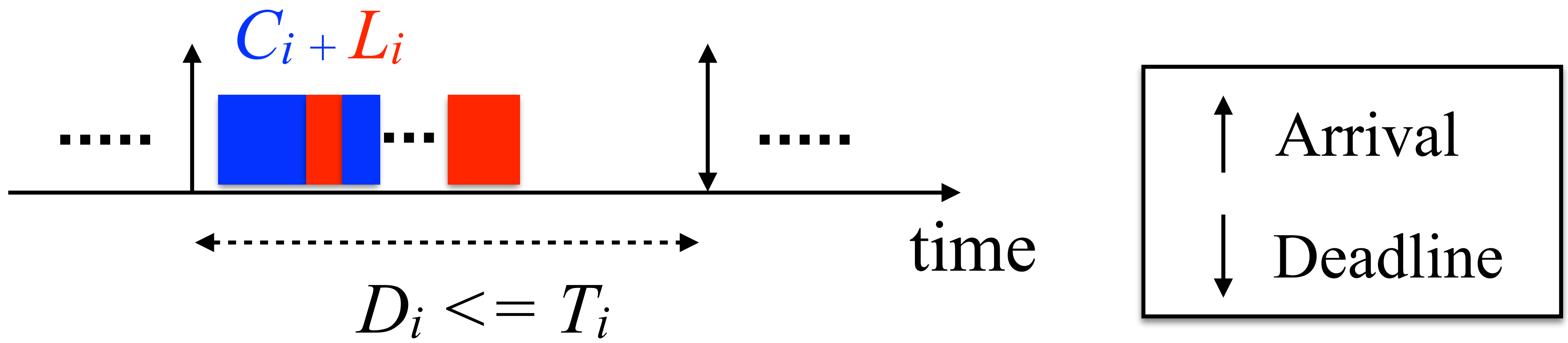
Motivation

Limited-preemption scheduling is an alternative to fully-preemptive scheduling and non-preemptive scheduling

	Fully-preemptive	Non-preemptive
Schedulability	Better schedulability	Higher priority jobs may be blocked by lower priority jobs
WCET and Run-time overheads	Larger and harder to determine	The system model is closer to the real system
Access to shared resources (multiprocessors)	Non-trivial synchronization protocols needed	Synchronization protocols are simpler to implement

In limited-preemption scheduling a job executes preemptively until it needs to execute non-preemptively

Limited-Preemption Sporadic Task Model



- Each task τ_i is characterized by four parameters:
  - C_i - preemptive worst-case execution time
  - L_i - non-preemptive worst-case execution time
  - T_i - minimum inter-arrival separation (period)
  - D_i - deadline (implicit or constrained)
- Utilization, $U_i = (C_i+L_i)/T_i$
- A task set τ, consists of *n tasks*

Schedulability Test

- In this work we propose a schedulability test for *limited-preemption sporadic task sets* on *m identical processors* under *Global Earliest Deadline First* (GEDF)

- Prior work*: A schedulability test has been proposed for fully-preemptive sporadic task sets τ, $\tau_i = \{C_i, T_i, D_i\}$, on *m identical processors* under *GEDF*

    - The analysis is based on computing the total execution demand of all jobs over a certain *interval t*

- Our Contribution: Extension to limited-preemption sporadic task sets τ, $\tau_i = \{C_i, L_i, T_i, D_i\}$

    - We compute the *maximum blocking* a job can experience over a certain *interval t* due to the non-preemptive execution of lower-priority jobs

*S. Baruah, “Techniques for Multiprocessor Global Schedulability Analysis”, RTSS 2007

Properties

- *Pseudo-polynomial time* for all task sets for which total utilization is bounded by a constant strictly less than m
- Sufficient and necessary for uniprocessors
- Sufficient for multiprocessors
- *Sustainable* with respect to all parameters; $\{C_i, L_i, T_i, D_i\}$

Application to Multi-GPU Systems

- Recent work has been done towards incorporating GPUs (Graphical Processing Units) as a shared processing unit in real-time systems

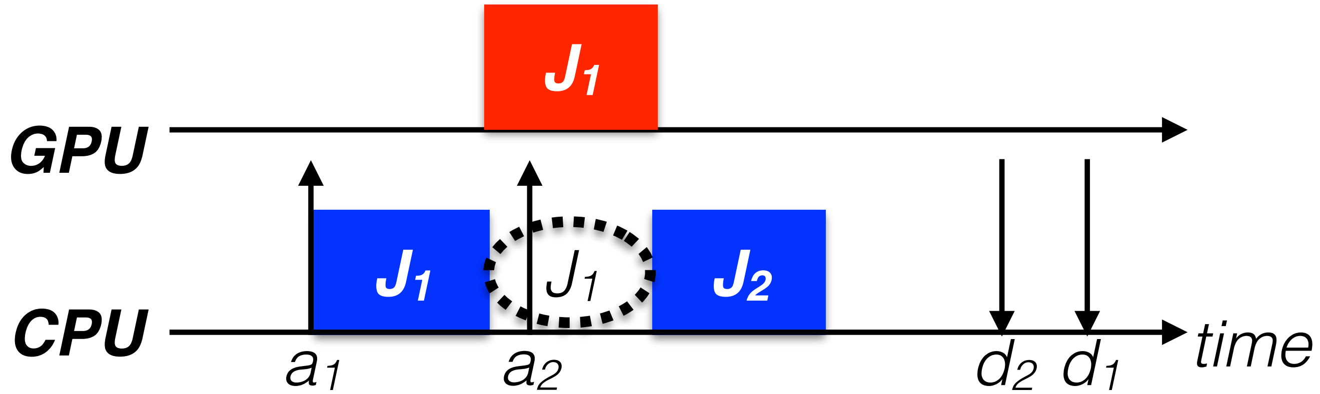
- Multi-GPU system model:

  - *m identical CPUs* and *g identical GPUs*

  - Each task τ_i may execute on the CPU and GPU. Consider that a task makes a single request to a GPU, and may execute on the GPU for a total of G_i time units

  - On GPUs *execution is non-preemptive*

- When a job executes non-preemptively on a GPU, the job *busy-waits non-preemptively* on a CPU. Other options: suspension, busy-wait preemptively



J_1 busy-waits non-preemptively on the CPU, therefore J_2 must wait to execute

- A synchronization approach* is used to access GPUs

- For the given *synchronization approach*, given *values of g and m*, and G_i, for each task τ_i, we compute L_i *(worst-case non-preemptive busy-waiting)* for use in our schedulability test

* A. Block, H. Leontyev, B. Brandenburg, and J. Anderson, "A flexible real-time locking protocol for multiprocessors," RTCSA 2007

Experimental Evaluation

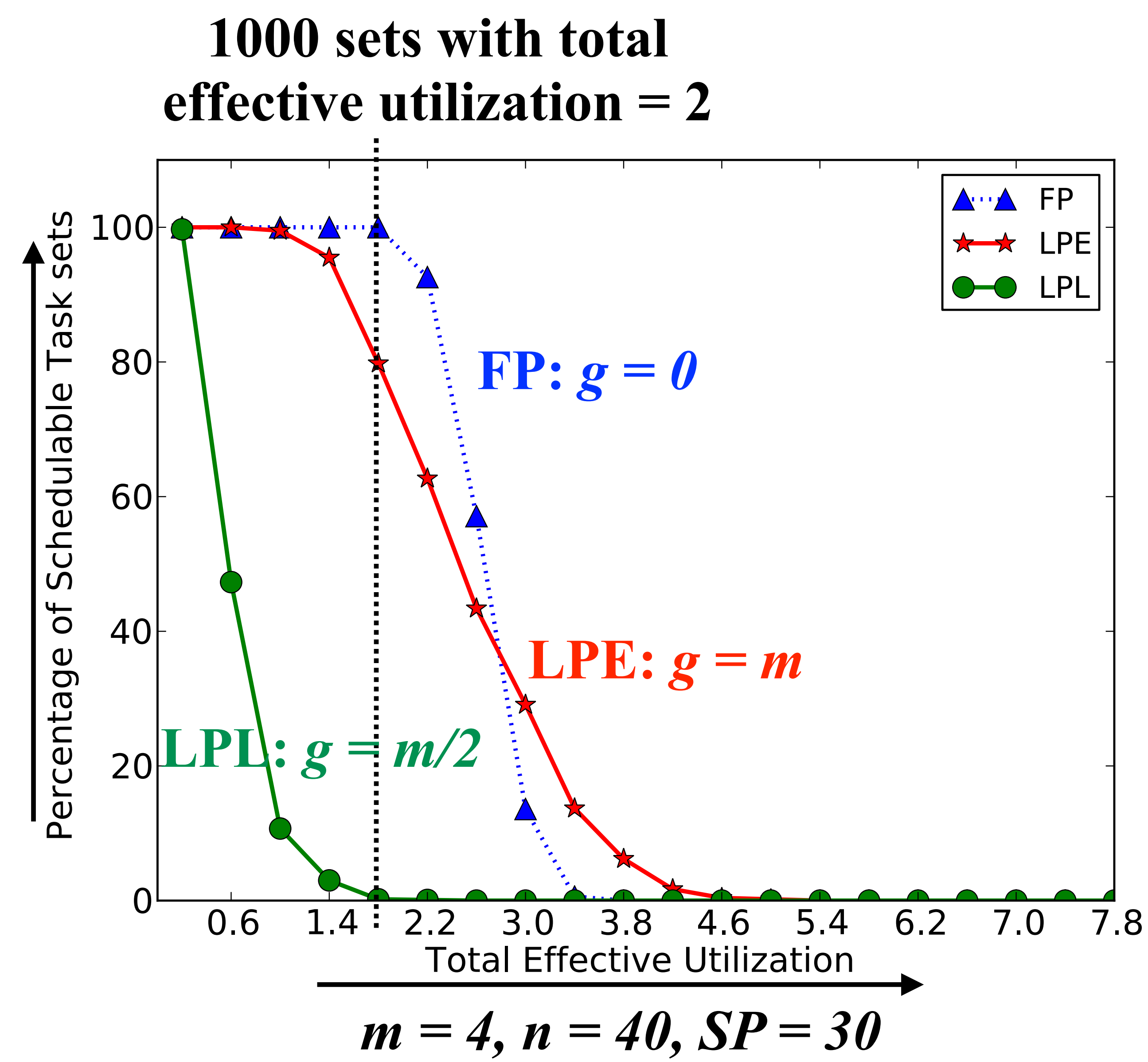
- Schedulability experiments: randomly generated task sets and determined the percentage of task sets that are schedulable under the proposed schedulability test

- Compared schedulability under different platform configurations:

  - Limited-preemption + Multi-GPU system with $g = m$ (LPE)

  - Limited-preemption + Multi-GPU system with $g = m/2$ (LPL)

  - Full-preemption with $g = 0$ (FP)



- For each *total effective utilization*,1000 sets each with n effective utilization values $\{u_1 \ldots u_n\}$, were generated using the *UUnifast-Discard algorithm*
- For a set of n effective utilization values, $\{u_1 \ldots u_n\}$, *3 corresponding task sets* were generated

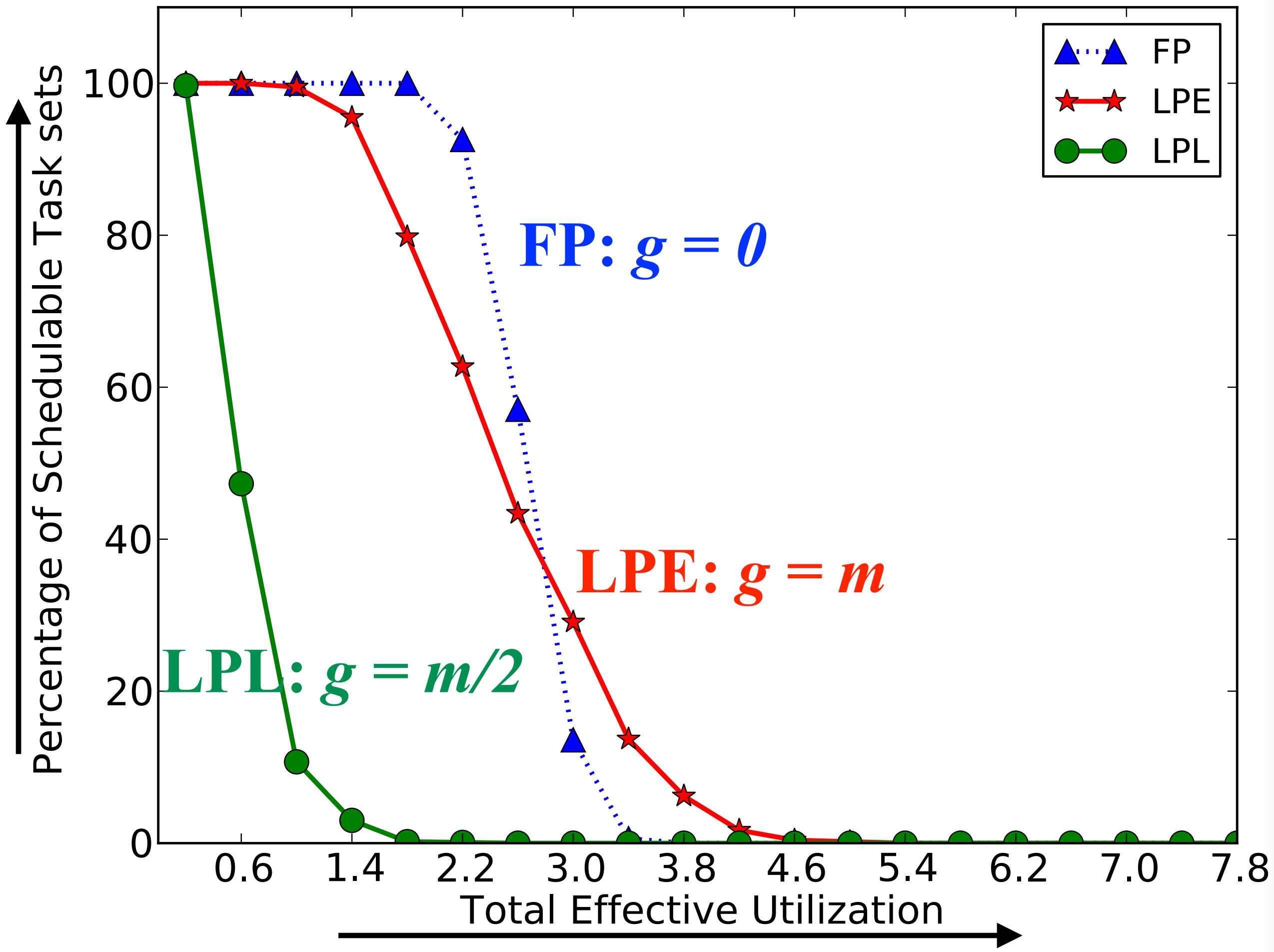
Results

- LPE has better schedulability than FP for higher values of total effective utilization
- LPE has significantly better schedulability than LPL
- For the same total effective utilization:
    - for smaller values of SP, the length of L_i increases and schedulability decreases
    - for smaller values of n, schedulability in all 3 cases decreases. However, the trends observed in the graphs are consistent

Summary

- We proposed a schedulability test for limited-preemption scheduling under GEDF
- Applied the schedulability test to a Multi-GPU system model with non-preemptive busy-waiting
- Performed schedulability experiments and compared schedulability under different platform configurations

Thank you!



$m = 4$, $n = 40$, $SP = 30$

- For a set of *n* effective utilization values, {
  sets were generated with the following task parameters:

- FP: $g = 0$
  - T_i
    1000]
  - D_i
  - G_i
    on the GPU)
  - C_i
  - L_i

- LPE: $g = m$
  - g_i
  - G_i
  - C_i
  - L_i

- LPL: $g = m/2$
  - L_i