Global Mixed-Criticality Scheduling on Multiprocessors

Haohan Li          Sanjoy Baruah
Department of Computer Science
The University of North Carolina at Chapel Hill
Introduction

- Scheduling mixed-criticality task systems
  - Sporadically releasing
  - Identical multiprocessors
  - Global scheduling
  - Two criticality levels
  - Implicit deadlines
### Mixed-criticality sporadic task model

- **Period:** $P_i$
- $P_1 = 3$, $P_2 = 6$
Mixed-criticality sporadic task model

- Period: $P_i$
  - $P_1=3$, $P_2=6$
- WCET: $C_i(LO)$, $C_i(HI)$
  - For $T_1$: $C_1(LO)=2$
  - For $T_2$: $C_2(LO)=1$, $C_2(HI)=3$
- Mixed-criticality sporadic task model

  - If every job uses up to low-criticality WCET, all deadlines must be met
- **Mixed-criticality sporadic task model**
  - If every job uses up to **low-criticality** WCET, all deadlines must be met.
Mixed-criticality sporadic task model

- If every job uses up to low-criticality WCET, all deadlines must be met.
- If some job uses more than low-criticality WCET, only high-criticality deadlines must be met.
- **Mixed-criticality sporadic task model**
  - If every job uses up to low-criticality WCET, all deadlines must be met
  - If some job uses more than low-criticality WCET, only high-criticality deadlines must be met
  - Execution times are only known at run-time
Prior Work

- **EDF-VD**
  - *Mixed-Criticality Scheduling of Sporadic Task Systems*, Baruah et al., ESA 2011
    - Earliest Deadline First with Virtual Deadlines
    - An EDF-based *mixed-criticality* scheduling algorithm for *implicit-deadline* systems on uniprocessor
    - Speed-up factor is at most $(\sqrt{5}+1)/2 \approx 1.618$
Prior Work

- **fpEDF**
    - Fixed Priority Earliest Deadline First
    - Fixed-job-priority scheduling algorithm
    - An EDF-based *global* scheduling algorithm for *implicit-deadline* systems on *multiprocessors*
    - Speed-up factor is at most 2
A global preemptive algorithm for scheduling mixed-criticality implicit-deadline systems on identical multiprocessors

- Proving that speed-up factor is at most $\sqrt{5+1} \approx 3.236$
- Experiments to show effectiveness
Scheduling Algorithm

Offline pre-processing step → Run-time dispatching
Scheduling Algorithm

Assigning virtual deadlines to high-criticality tasks

Schedulability test with given virtual deadlines

Run-time dispatching
Scheduling Algorithm

- Assigning virtual deadlines to high-criticality tasks
- Schedulability test with given virtual deadlines
- Use fpEDF to schedule tasks with virtual deadlines
- Use fpEDF to schedule original high-criticality tasks
- Criticality change
Scheduling Algorithm

Assigning virtual deadlines to high-criticality tasks

Schedulability test with given virtual deadlines

Use fpEDF to schedule tasks with virtual deadlines

Use fpEDF to schedule original high-criticality tasks

Criticality change
Assigning virtual deadlines to high-criticality tasks

- Virtual deadlines are assigned proportionally to original deadlines with factor $x \ (0 < x < 1)$
- In the example, assume that $x = 1/2$, then the virtual deadline is $xP_2 = 1/2 \times 6 = 3$
Scheduling Algorithm

Assigning virtual deadlines to high-criticality tasks

Schedulability test with given virtual deadlines

Use fpEDF to schedule tasks with virtual deadlines

Use fpEDF to schedule original high-criticality tasks

Criticality change
Scheduling Algorithm

Assigning virtual deadlines to high-criticality tasks

Test low-crit behavior
Test high-crit behavior

Use fpEDF to schedule tasks with virtual deadlines

If Criticality change,
Use fpEDF to schedule original high-criticality tasks
Schedulability test with given virtual deadlines

- Low-criticality sufficient schedulability condition
  - at low-criticality, the virtual system with virtual periods is schedulable

Assuming that in low-criticality behavior, all high-criticality tasks must be completed by their virtual deadlines
### Scheduling Algorithm

- **Schedulability test with given virtual deadlines**
  - **Low-criticality** sufficient schedulability condition
    - at low-criticality, the virtual system with *virtual periods* is schedulable
    - The virtual periods of high-crit tasks will be $xP_i$
    - Test the virtual system with fpEDF’s schedulability condition

![Diagram showing low-criticality and high-criticality tasks](image)
Scheduling Algorithm

- Schedulability test with given virtual deadlines
  - High-criticality sufficient schedulability condition
    - At high-criticality, the virtual system with remaining periods and high-crit WCETs is schedulable

Only high-crit demand is left after criticality change
Schedulability test with given virtual deadlines

- **High-criticality sufficient schedulability condition**
  - at high-criticality, the virtual system with remaining periods and high-crit WCETs is schedulable
  - The virtual periods and virtual WCET of high-crit tasks will be \((1-x)P_i\) and \(C_i(HI)\)
  - Test the virtual system with fpEDF’s schedulability condition
Assigning virtual deadlines to high-criticality tasks

Test low-crit behavior
Test high-crit behavior

Use fpEDF to schedule tasks with virtual deadlines

Use fpEDF to schedule original high-criticality tasks

Criticality change
Run-time dispatching

- Use fpEDF to schedule tasks with virtual deadlines
Run-time dispatching

- Use fpEDF to schedule tasks with virtual deadlines
- If high-crit behavior detected, drop low-crit tasks and use fpEDF to schedule high-crit tasks with original deadlines
Scheduling Algorithm

Assigning virtual deadlines to high-criticality tasks

- Test low-crit behavior
- Test high-crit behavior

Use fpEDF to schedule tasks with virtual deadlines

Use fpEDF to schedule original high-criticality tasks

Criticality change
Assigning virtual deadlines to high-criticality tasks

Test low-crit behavior

Test high-crit behavior

Use fpEDF to schedule tasks with virtual deadlines

Criticality change

Use fpEDF to schedule original high-criticality tasks
Scheduling Algorithm

- The selection of $x$
  - Computation from theoretical analysis
    - Fig. 2 and Theorem 3 in the paper
    - Does not always generate a proper $x$
    - Guarantees the speed-up factor
  - Enumeration as a pragmatic improvement
    - The possible values of $x$ are finite
    - If none of them passes the test, report unschedulable
Global mixed-criticality scheduling on multiprocessors, July 12th, 2012

Scheduling Algorithm

1. Compute, then enumerate values of $x$.
   - Test low-crit behavior
   - Test high-crit behavior

2. Use fpEDF to schedule tasks with virtual deadlines.

3. Use fpEDF to schedule original high-criticality tasks.

4. Criticality change
Scheduling Algorithm

Compute, then enumerate values of $x$

- Test low-crit behavior
- Test high-crit behavior

Use fpEDF to schedule tasks with virtual deadlines

Use fpEDF to schedule original high-criticality tasks

Criticality change
Simulation experiments

- Generate random MC implicit-deadline tasks with given utilization in low and high criticality
  - Utilization is defined as the maximum fraction of time demand in each criticality
  - $U(LO) = \frac{2}{3} + \frac{1}{6} = \frac{5}{6}$
  - $U(HI) = \frac{3}{6}$
Simulation experiments

- Generate random MC implicit-deadline tasks with given utilization in low and high criticality
  - The task generation algorithm is in Fig. 4
- Compare 3 algorithms
  - Regular-EDF: Worst-case reserving to meet all deadlines
  - GLOBAL: Using computed $x$ only
  - GLOBAL-PRAGMATIC: Using computed and enumerated $x$
Experimental results

![Chart showing performance results](chart.png)
Experimental results
Conclusion

- **Global mixed-criticality scheduling**
  - Global-EDF with virtual deadlines
  - Schedulability test on two virtual systems
  - Selected virtual deadlines
  - Speed-up factor 3.236, and experiments

- **Future work**
  - More criticality levels
  - Better scheduling strategies
Thank you