Limited Pre-emptive Global Fixed Task Priority
In a nutshell...

We studied the preemptive scheduling problem on multicores

We focused on the **limited preemptive model**, which is somewhat between fully preemptive and non-preemptive approaches
Preemption or no preemption?

![Diagram showing time and deadline with low and high priorities.](image)
Preemption or no preemption?

Diagram showing the comparison of low and high priority tasks with different deadlines. The diagram illustrates the impact of preemption on task fulfillment within the given time frames.
We studied the preemptive scheduling problem on multicores

We focused on the limited preemptive model, which is somewhat between fully preemptive and non-preemptive approaches.

We came up with two different scheduling techniques and derived schedulability analyses for each of them.

We played around with the parameters to further reduce the number of preemption and increase the schedulability.
System model & assumptions

Application model:
Sporadic constrained-deadline tasks

Minimum interarrival time
WCET Deadline WCET
Minimum interarrival time
WCET Deadline WCET

Time
Scheduler model:
Global Fixed Task Priority Scheduling
Fixed Non-Preemptive Region
Scheduler model:
Global Fixed Task Priority Scheduling
Fixed Non-Preemptive Region

Defined by our algorithm
System model & assumptions

Platform model:
Identical multiprocessor platform
m processors

Assumptions:
No migration cost,
No preemption cost
Independant tasks

Focus on the two new scheduling techniques that we propose and the impact on the schedulability
Some necessary background

A bit of history...
Background on fully-preemptive schedulability analysis

Maximum interference from HP workload in $t$ time units
Background on fully-preemptive schedulability analysis

Result 1: Maximum workload without carry-in job

By considering only jobs released within the interval
Background on fully-preemptive schedulability analysis

Result 2: Maximum workload with carry-in

- **Result 2**: Maximum workload with carry-in
- **Result 3**: Worst-case scenario
- **Result 4**: Maximum HP workload
- **Result 5**: Maximum Interference

**Result 2: Maximum workload with carry-in**

- **Max workload W/O carry-in**
- **Max workload W/ carry-in**

**Minimum interarrival time**

**Max response time**

**Interval of time of length t**

**WCET**
Result 3: Worst-case interference scenario

If m cores and HP tasks:
- m-1 tasks with carry-in
- Others without carry-in and release at the start of the window

WCET

Maximum interference from HP workload in t time units
Background on fully-preemptive schedulability analysis

Result 1: Max workload W/O carry-in
Result 2: Max workload W/ carry-in
Result 3: Worst-case scenario

Result 4: Maximum Higher Priority Workload

1. Worst-case interference scenario
2. Maximum task workload without carry-in job
3. Maximum task workload with carry-in job

Result 5: Maximum Interference

Maximum workload that can be generated within the given time window
Result 1: Max workload W/O carry-in

Result 2: Max workload W/ carry-in

Result 3: Worst-case scenario

Result 4: Maximum Interference

Result 5: Maximum HP workload

Background on fully-preemptive schedulability analysis

Result 4: Maximum interference

Max interference = max workload / m
Preemptive techniques on multicores

Definition of fully preemptive scheduling on multicores

“Policy where at any time, the m highest priority tasks execute on the m processors of the platform.”

In limited preemptive scheduling
Definition of fully preemptive scheduling on multicores

“policy where at any time, the $m$ highest priority tasks execute on the $m$ processors of the platform.”
Definition of fully preemptive scheduling on multicores

“Policy where at any time, the m highest priority tasks execute on the m processors of the platform.”

Medium priority

Low priority

Regular Deferred Scheduling (RDS): New arriving higher priority workload preempts the first preemptible lower priority workload.
Preemptive techniques on multicores

Definition of fully preemptive scheduling on multicores

“Policy where at any time, the m highest priority tasks execute on the m processors of the platform.”

Adapted Deferred Scheduling (ADS): New arriving higher priority workload preempts the lowest priority job once it gets preemptible.
What's wrong with RDS?
What's wrong with RDS

Result 3: Worst-case scenario

If m cores and HP tasks:
- m-1 tasks with carry-in
- Others without carry-in and release at the start of the window

With RDS, all the tasks, including HP and LP, can interfere with the task under analysis
And what about ADS?
Let’s focus on ADS

Result 3: Worst-case interference scenario

If m cores and HP tasks:
- m-1 HP tasks \textbf{with} carry-in
- Others HP \textbf{without} carry-in and release at the start of the window

Maximum interference from HP workload in t time units
Result 3: Worst-case interference scenario

If $m$ cores and HP tasks:
- $m-1$ HP tasks with carry-in
Let's focus on ADS

Result 3: Worst-case interference scenario

Result 3: Worst-case interference scenario

If m cores:
- m-1 HP tasks
- 1 LP job

If m cores:
- m-2 HP tasks
- 2 LP jobs

If m cores:
- m-3 HP tasks
- 3 LP jobs

If m cores:
- 0 HP tasks with carry-in
- m LP carry-in jobs

And so on...
Assume 4 carry-in jobs from LP tasks

We must compute the maximum interfering workload generated by these 4 LP carry-in jobs

To compute the exact area requires to enumerate all possible subsets of 4 tasks out of the set of LP tasks
Our solution

We propose three different techniques to upper-bound this area:
• Two of them are fairly straightforward
• The third one is more complex
Our solution

Upper-bound on the area

Maximum interfering workload from LP tasks

Maximum interference from HP and LP tasks

Schedulability test
We designed a top-down approach (starting with the highest priority task) to assign to each task the maximum non-preemptive region length such that all the HP tasks are still schedulable.

Reduce the number of preemption
Some preliminary results

Fully preemptive

ADS
Questions?

Thank you for your attention 😊