
Waqar Ali, Heechul Yun
University of Kansas
Multicore Processors

- Provide high computing performance
- Needed for intelligent safety-critical real-time systems
Parallel Real-Time Tasks

- Many emerging workloads in AI, vision, robotics are parallel real-time tasks

DNN based real-time control

Effect of parallelization on DNN control task

Effect of Co-Scheduling

- DNN control task suffers \textbf{>10X slowdown}
  - Due to interference in shared memory hierarchy

\[ \text{It can be worse!} \quad \text{[Bechtel, RTAS'19]} \]

Observations

• Interference in shared memory hierarchy
  – Can be very high and unpredictable
  – Depends on the hardware (black box)
• Constructive sharing (Good)
  – Between threads of a single parallel task
• Destructive sharing (Bad)
  – Between threads of different tasks

• Goal: analyzable and efficient parallel real-time task scheduling framework for multicore
RT-Gang

- One (parallel) real-time task---a gang---at a time
  - Eliminate inter-task interference by construction
- Schedule best-effort tasks during slacks *w/ throttling*
  - Improve utilization with bounded impacts on the RT tasks
Safe Best-Effort Task Throttling

- Throttle the best-effort core(s) if it exceeds a given bandwidth budget set by the RT task

Throttling mechanism [Yun, RTAS’13]

Virtual Gang

- Statically group RT tasks as a "virtual gang"
  - All threads of a virtual gang are scheduled together
Implementation

• Modified Linux’s RT scheduler
  – Implemented as a “feature” of SCHED_FIFO (sched/rt.c)
  – Enforce one real-time priority across all cores (invariant)
  – A high priority RT thread preempts lower priority RT threads on any cores (gang preemption)

• Best-effort task throttling
  – Based on BWLOCK++ [Ali, ECRTS’18]
  – Each RT task sets the tolerable throttling threshold
  – Enforced by the kernel-level bandwidth regulators for any co-scheduled best-effort tasks

Evaluation

• Setup
  – Linux 4.14 baseline
  – Raspberry Pi 3 (4x Cortex-A53)
  – NVIDIA Jetson TX2 (4x Cortex-A57)

• Benchmarks
  – IsolBench (synthetic RT/BE)
  – DNN control task of DeepPicar (real-world RT)
  – Parboil benchmarks (real-world BE)
Synthetic Taskset

<table>
<thead>
<tr>
<th>RT Task</th>
<th>WCET (ms)</th>
<th>Period (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \tau_1 )</td>
<td>3.5</td>
<td>20</td>
</tr>
<tr>
<td>( \tau_2 )</td>
<td>6.5</td>
<td>30</td>
</tr>
</tbody>
</table>

Deterministic timing is achieved
DNN Taskset

Deterministic timing is achieved

<table>
<thead>
<tr>
<th>Task</th>
<th>WCET (C ms)</th>
<th>Period (P ms)</th>
<th># Threads</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_{dnn}^{rt}$</td>
<td>34</td>
<td>78</td>
<td>2</td>
</tr>
<tr>
<td>$t_{bww}^{rt}$</td>
<td>47</td>
<td>100</td>
<td>4</td>
</tr>
<tr>
<td>$t_{cutcp}^{be}$</td>
<td>$\infty$</td>
<td>$N/A$</td>
<td>4</td>
</tr>
<tr>
<td>$t_{lbm}^{be}$</td>
<td>$\infty$</td>
<td>$N/A$</td>
<td>4</td>
</tr>
</tbody>
</table>
Related Work

• Gang scheduling
  – A. Melani et al., “A scheduling framework for handling integrated modular avionic systems on multicore platforms.” In *RTCSA*, 2017

• Key differences of our work
  – First gang scheduling implementation on an actual OS
  – Integrate throttling to safely co-schedule best-effort tasks
Conclusion

• Parallel real-time task scheduling
  – Hard to analyze on COTS multicore
  – Due to interference in shared memory hierarchy

• RT-Gang
  – Analyzable and efficient parallel real-time gang scheduling framework
  – Implemented in Linux

https://github.com/CSL-KU/rt-gang
Thank You!

Disclaimer:
This research is supported by NSF CNS 1718880, CNS 1815959, and NSA Science of Security initiative contract #H98230-18-D-0009.