Analyzable and Practical Real-Time Gang Scheduling on Multicore Using RT-Gang

Waqar Ali, Michael Bechtel, Heechul Yun
University of Kansas
Outline

• RT-Gang
• Tutorial
• DeepPicar Case Study
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 >10X slowdown
  - Due to interference in shared memory hierarchy

It can be worse! (> 300X slowdown)*

* Michael G. Bechtel and Heechul Yun. “Denial-of-Service Attacks on Shared Cache in Multicore: Analysis and Prevention.” In RTAS, 2019
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
  – By avoiding destructive sharing
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

---

* Yun et al., “MemGuard: Memory Bandwidth Reservation System for Efficient Performance Isolation in Multi-core Platforms.” In RTAS, 2013
* W. Ali and H. Yun., “Protecting Real-Time GPU Kernels on Integrated CPU-GPU SoC Platforms.” In ECRTS, 2018
Implementation

• Modified Linux’s RT scheduler
  – Implemented as a “feature” of SCHED_FIFO (sched/rt.c)

• Best-effort task throttling
  – A separate kernel module based on BWLOCK++ *

* W. Ali and H. Yun., “Protecting Real-Time GPU Kernels on Integrated CPU-GPU SoC Platforms.” In ECRTS, 2018
Outline

• RT-Gang
• Tutorial
• DeepPicar Case Study
Source Code Repository

- git clone https://github.com/CSL-KU/RT-Gang
Installation

- From the Linux kernel directory:
  - patch -p1 < ../RT-Gang/rtgang-v4.19.patch
  - Compile & install & restart

- To check if installed correctly:
  - sudo cat /sys/kernel/debug/sched_features | grep RT_GANG_LOCK
Enable/Disable RT-Gang

- RT-Gang is enabled/disabled through the kernel's scheduling feature

```
# Enable RT-Gang
echo RT_GANG_LOCK >> /sys/kernel/debug/sched_features

# Disable RT-Gang
echo NO_RT_GANG_LOCK >> /sys/kernel/debug/sched_features
```
Best-Effort Task Throttling

• Throttling is enabled through a kernel module
  – cd RT-Gang/throttling/kernel_module
  – make
  – sudo insmod exe/bwlockmod.ko
Best-Effort Task Throttling

- Only occurs when a real-time task is running
  - W/o real-time task
    
    ```
    pi@raspberrypi:~ $ bandwidth
    memsize=4096 KB, type=read, cpuid=0
    stop at 5
    g_nread(bytes read) = 11228151808
    elapsed = 5.00 sec ( 50000001 usec )
    CPU0: B/W = 2141.60 MB/s | CPU0: average = 28.50 ns
    ```

  - W/ real-time task
    
    ```
    pi@raspberrypi:~ $ sudo chrt -f 1 bandwidth -t 0 &> /dev/null &
    [1] 1222
    pi@raspberrypi:~ $ bandwidth
    memsize=4096 KB, type=read, cpuid=0
    stop at 5
    g_nread(bytes read) = 524288000
    elapsed = 5.00 sec ( 5000394 usec )
    CPU0: B/W = 99.99 MB/s | CPU0: average = 610.40 ns
    ```
Outline

• RT-Gang
• Tutorial
• DeepPicar Case Study
DeepPicar

- **A low cost**, small scale replication of NVIDIA’s DAVE-2
- **Uses the exact same DNN**
- Runs on a Raspberry Pi 3 in **real-time**

<table>
<thead>
<tr>
<th>Item</th>
<th>Cost ($)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raspberry Pi 3 Model B</td>
<td>35</td>
</tr>
<tr>
<td>New Bright 1:24 scale RC car</td>
<td>10</td>
</tr>
<tr>
<td>Playstation Eye camera</td>
<td>7</td>
</tr>
<tr>
<td>Pololu DRV8835 motor hat</td>
<td>8</td>
</tr>
<tr>
<td>External battery pack &amp; misc.</td>
<td>10</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>70</strong></td>
</tr>
</tbody>
</table>

* Bechtel et al. DeepPicar: A Low-cost Deep Neural Network-based Autonomous Car. In RTCSA, 2018
DNN based Real-Time Control

while True:
    # 1. read from the forward camera
    frame = camera.read()
    # 2. convert to 200x66 rgb pixels
    frame = preprocess(frame)
    # 3. perform inferencing operation
    angle = DNN_inferencing(frame)
    # 4. motor control
    steering_motor_control(angle)
    # 5. wait till next period begins
    wait_till_next_period()

• DNN Inferencing is the most compute intensive part.
• Parallelized by TensorFlow to utilize multiple cores.
Experiment Setup

- DNN control task of DeepPicar (real-world RT)
- IsolBench BwWrite benchmark (synthetic RT)
- Parboil benchmarks (real-world BE)

<table>
<thead>
<tr>
<th>Task</th>
<th>WCET (C ms)</th>
<th>Period (P ms)</th>
<th># Threads</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_{rt}^{rt_{dnn}}$</td>
<td>34</td>
<td>100</td>
<td>2</td>
</tr>
<tr>
<td>$t_{rt}^{rt_{bww}}$</td>
<td>220</td>
<td>340</td>
<td>2</td>
</tr>
<tr>
<td>$t_{be}^{cutcp}$</td>
<td>$\infty$</td>
<td>N/A</td>
<td>4</td>
</tr>
<tr>
<td>$t_{be}^{lbm}$</td>
<td>$\infty$</td>
<td>N/A</td>
<td>4</td>
</tr>
</tbody>
</table>

Parboil cutcp & lbm

DNN
- Core1
- Core2

BwWrite
- Core3
- Core4

LLC
- DRAM

RT

BE
Execution Time Distribution

What does this look like in the real world?

- RT-Gang achieves deterministic timing
CoSched (w/o RT-Gang)

https://youtu.be/Jm6KSDqlqiU
RT-Gang

https://youtu.be/pk0j063cUAs
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
  – Avoid interference by construction
    • Can protect critical real-time tasks

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.