Scheduling Techniques for Automated Driving Systems using the AUTOSAR Adaptive Platform

Yuchen Zhou, Thomas E Fuhrman, Prathap Venugopal
General Motors
Agenda

Introduction
Adaptive Platform & Scheduling Techniques
Scheduling policies
Bench experiments
Conclusion and Next steps
Background and Motivation

• Safety Mechanisms within AUTOSAR
  • AUTOSAR Classic defines functional safety mechanisms both in application level and infrastructure level
    • Memory protection, watchdog management, end-to-end protection, etc.
  • AUTOSAR Classic solution is well matured, implementations widely available
Introduction

As we develop automated driving systems, industry needs to learn how effectively use the safety mechanisms (Freedom from interference, mixed criticality integration, etc.) within the software platform.

**AUTOSAR Adaptive Platform (AP)** is one of the strong candidates providing safety and security mechanisms.

Adaptive Platform operating systems are expected to support POSIX standard. POSIX itself has inherent safety mechanisms e.g. memory protection.

In this presentation we focus on **timing protection** and **timing isolation** aspects of safety.
Scheduler and OS for AUTOSAR Adaptive R17-03

POSIX OS API and Adaptive Application (AA)
- Portable
- Sequencing, priority will be defined by application manifest
- AA managed by execution manager

OS
- POSIX API required
- POSIX scheduler
  - Real-time scheduler
- Linux: CBS scheduler
- Other OS
Scheduler for Adaptive AUTOSAR

POSIX and portability

- **Periodic real-time task** (function calls, thread) can be generated through timer and alarm based POSIX API
- Support **POSIX real-time scheduler** FIFO (SCHED_FIFO), Round Robin (SCHED_RR)

Linux specific

- RT Preempt patch
  - Kernel preemption point
  - **Reduced jitter** and **latency** of the real-time task
  - AUTOSAR Adaptive demonstrator is currently using Yocto Linux + Preempt-RT patch

- Other scheduler
  - **Dynamic scheduler**: CBS (SCHED_DEADLINE) (already merged into mainline Linux kernel)

Other commercial OS schedulers

- e.g. QNX Adaptive Partition Scheduler, Green Hills Integrity Partition Scheduler
POSIX Scheduler (available in Linux)

- **SCHED_FIFO**
  - Real time scheduling. Low priority task will be preempt by high priority task.
  - Real time priority range from 1 to 99.
  - If threads have same priority, they are executed in a FIFO manner.

- **SCHED_RR**
  - Round robin added a soft deadline, called time quantum.
  - If thread has been running for longer than the time quantum, it will be put to the end of the list for its priority.

- Default Scheduler
  - **SCHED_OTHER**
  - Priority is 0.
  - Fair execution among all threads.
PREEMPT_RT patch

Enable kernel preemption

The result has been into the Linux kernel, but still not fully integrated due to driver compatibility and strict timing requirement. Real time scheduler such as FIFO and RR scheduler is needed to actually utilize the capability.
Constant Bandwidth Server (CBS)

Since Linux kernel 3.14, constant bandwidth server and earliest deadline first (SHED_DEADLINE) based scheduler is introduced.

- Runtime, deadline, and period

- The CBS guarantees non-interference between tasks, by throttling threads that attempt to over-run their specified runtime.
- Time resolution can be set around microsecond
- The thread associated are set as the highest priority. It will preempt any other threads using other scheduling policy including FIFO and RR.
Bench experiments and Results

Evaluation of Linux scheduler on i.MX 6 Sabre Lite board
- Latency characteristic for standard Linux and RT Linux
- RT Linux and FIFO scheduler evaluation
- CBS and timing isolation

Latency - ST vs RT

Tracer of CBS scheduling in ST vs RT
Conclusion

Recommendation

– Use FIFO and RR by themselves if output jitter is important
  • CBS preempt real-time task which can leads to variation in jitter and latency

– Use CBS as a “mechanism” to cut off overrunning processes
  • Use CBS for heavy computation load processes such as a process that launches machine learning and image processing code
  • Use FIFO and RR for real-time applications which requires deterministic behavior

– Prefer low latency standard Linux over RT patched Linux
  • CBS works better with standard Linux based on experiment
  • RT patched Linux has driver limitation and may be less reliable due to its limited user base
Next Steps

Continue working on scheduling policies evaluation on AP 17-10 release
Evaluate commercial OS scheduling policies within AP
QUESTIONS?
Thank you for your attention!

Yuchen Zhou, Thomas E. Fuhrman, Prathap Venugopal

General Motors Research & Development
Mail Code: 480-106-RE2
30470 Harley Earl Blvd
Warren, MI 48092-2031
yuchen.zhou@gm.com
thomas.e.fuhrman@gm.com
prathap.venugopal@gm.com