Chain-Based Fixed-Priority Scheduling of Loosely-Dependent Tasks

Hyunjong Choi, Mohsen Karimi, Hyoseung Kim
Motivational example

I. Introduction

Perception reaction distance (A) Braking deceleration distance (B) Total stopping distance (C)

Complex information flows implemented with chains of tasks

II. Related work and contributions
Loosely-dependent task chains

- Each task executes and produces output at its own rate
  - Based on most recent input data from a preceding task
  - e.g., publisher-subscriber in ROS, read-execute-write in AUTOSAR
- Give flexibility in system design, scheduling, and information sharing

Goal: Minimize the end-to-end latency of loosely-dependent chains
Contributions

- Propose a new chain-based fixed-priority scheduler that identifies effective chain instances producing valid and updated chain outputs.
- Present an analytical method to upper-bound the end-to-end latency of chains under the proposed scheduler.
- Significantly outperforms the state-of-the art chain-unaware schedulers
  - Up to 83% reduction in end-to-end latency with a shorter update rate of valid chain output.

Prior Work:

- Chain-unaware schedulers
  - Upper bound on latency based on the WCRT
    - Abdullah et al. [DATE 2019]
    - Kloda et al. [ETFA 2018]
    - Becker et al. [RTCSA 2016]
- Limitations of DAG-based schedulings (inapplicable to tasks running asynchronously with different periods and priorities)
  - Ayan et al. [ICCPS 2019]
  - Han et al. [RTSS 2009]
System model

- Multi-core system with partitioned fixed-priority scheduling
- Task model: $\tau_i := (BC_i, WC_i, D_i, T_i, o_i, \pi_i)$
  - $BC_i$: The best-case execution time of a job of $\tau_i$
  - $WC_i$: The worst-case execution time of a job of $\tau_i$
  - $D_i$: The relative deadline of $\tau_i$ ($D_i \leq T_i$)
  - $T_i$: The period of $\tau_i$
  - $o_i$: The period of $\tau_i$
  - $\pi_i$: The priority of $\tau_i$
- Chain model: $\Gamma^c := [\tau_s, \tau_{m1}, \tau_{m2}, \ldots, \tau_e]$  
  - $\tau_s$: The start task of a chain $\Gamma^c$
  - $\tau_{m*}$: The intermediate task of a chain $\Gamma^c$
  - $\tau_e$: The end task of a chain $\Gamma^c$

< Example of chains >
Chain-based fixed-priority scheduler (1/2)

- Offline part: find effective chain instances from candidates

**Definition 1.**

An *effective instance* of a chain $\Gamma^c$ is the earliest instance producing a valid and updated final output using the *most recently updated input data*. The $i$-th effective instance of $\Gamma^c$ is denoted as $E^c[i]$.

---

III. Chain-based fixed-priority scheduling

**Step 1:** Initialize chain instance candidates

Create instances for job releases from the start task of a chain

**Step 2:** Build chain instances

Add each job of intermediate tasks to eligible chain instances

< Synthesis of chain instances and effective instances for the taskset >
Chain-based fixed-priority scheduler (2/2)

- Runtime part: *Release-and-Ready* (RNR) policy
  - Prevent unnecessarily early start of job execution
- **Two step-phases**
  - Release phase: arrival of a job according to its period, but cannot start execution
  - Ready phase: when previous jobs of the same chain instance have completed their execution

- **Rule 1. Job** $E^c[i, j]$ **in a single chain**
  - $E^c[i, j-1]$ complete, if $j \neq 1$
  - Most recent job of $E^c[i-1]$ to the same CPU, if $j = 1$

- **Rule 2. Job** $E^c[i, j]$ **in multiple chains**
  - Rule 1 is satisfied for all of its effective instances

- **Rule 3. Job** $E^c[i, j]$ **not in effective instance**
  - Default: dropped (skipped)

< 3 categories of jobs for ready phase of effective chain instance >
End-to-end latency analysis

IV. End-to-end latency analysis

- Consider self-suspension effect caused by release phase
- Interference from high priority jobs of other chains
- Iterate until converge upper- and lower-bounds

Step 1. Lower bound start-time and upper bound finish-time of a job

\[ L^c_{\text{start}} = \max_{\forall i} \bar{F}^c[i, N_c], \quad S^c[i, 1] \]

Step 2. Compute end-to-end latency of effective chain instance

\[ L^c = \max_{\forall i} \bar{F}^c[i, N_c] - S^c[i, 1] \]

Our analysis framework can also be used to analyze end-to-end latency under conventional chain-unaware fixed-priority schedulers

V. Evaluation
Evaluation

- Comparison with the state-of-the-art (single chain)
  - Abdullah et al.[2], Becker et al.[5]
  - SFA-RM: start- and finish-time based analysis under chain-unaware rate monotonic scheduling
  - CBS: proposed analysis framework under chain-based scheduler

- Use 500 tasksets with 7 tasks each for each utilization

- A chain with N tasks, left tasks are hard real-time tasks (i.e., modeled single-task chains)

Evaluation

- Comparison with the state-of-the-art (multiple chains with a mutual task)

- Utilization of 0.8 with 9 tasks that forms 2 chains
- Mutual task’s position

Chain set 1: start task, Chain set 2: end task, Chain set 3: intermediate task
VI. Conclusion

Conclusion

 New chain-based fixed-priority scheduling and analysis of end-to-end latency of chains
 The proposed scheduler outperforms the state-of-the-art with respect to end-to-end latency
 Our analysis framework can also be used for conventional chain-unaware scheduling policies

Future work

 Apply proposed scheduler to robotic platforms
 Investigate the timing unpredictability caused by shared memory resources in multi-core platforms
Thank you

Chain-Based Fixed-Priority Scheduling of Loosely-Dependent Tasks

- Hyunjong Choi, Mohsen Karimi, Hyoseung Kim

Q & A