50 YEARS OF COMPUTER SCIENCE AT YORK

www.york.ac.uk/computerscience

Resilience-aware Mixed-criticality DAG Scheduling on Multi-cores for Autonomous Systems

Authors: Jie Zou, Xiaotian Dai, John A. McDermid

Department of Computer Science, University of York

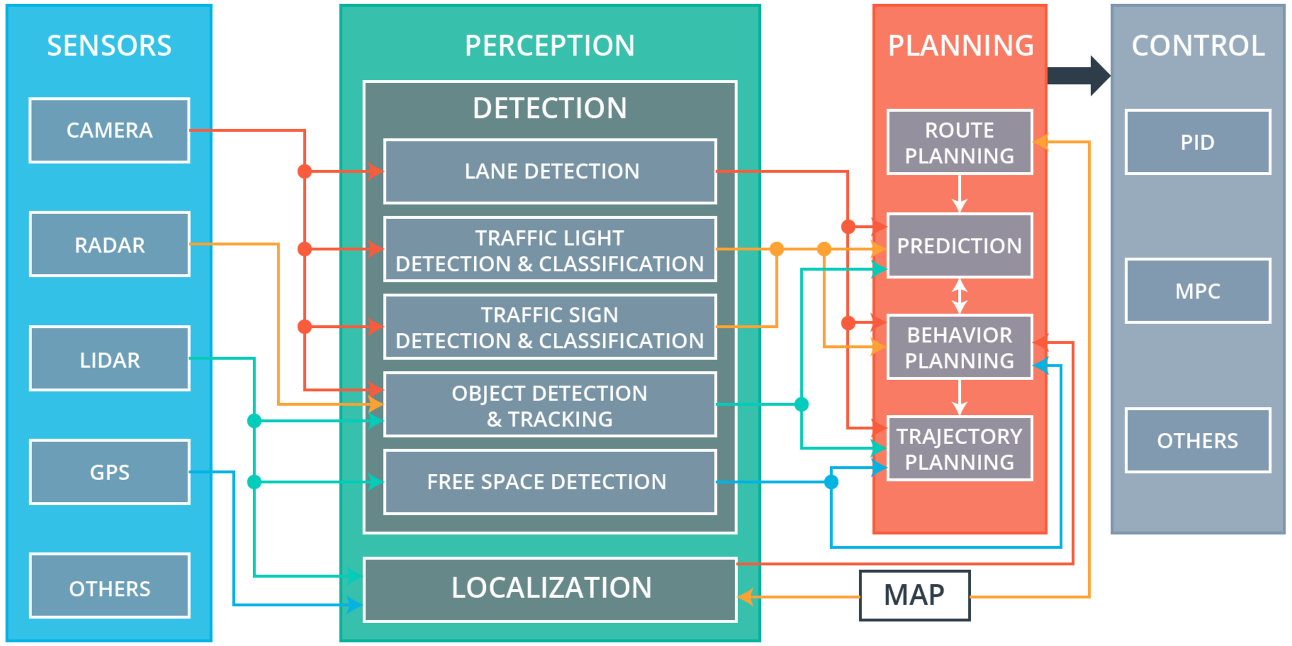
Email: jie.zou@York.ac.uk, xiaotian.dai@York.ac.uk, john.mcdermid@york.ac.uk

HISE Group

Background

Software Complexity

- Dramatic implementation of artificial intelligence (AI), including machine learning (ML).
- The applications have a high dependency degree and can involve an extremely deep processing pipeline.



https://medium.com/@macchakaran/udacity-self-driving-car-project-11-path-planning-a8266eb04515

RTS York HISE Group UNIVERSITY of York

Background

Safety Assurance Based Criticality Definition

Employing the concept of **Automotive Safety Integrity Levels (ASILs)** in the ISO26262 standard, we know the **functions in autonomous systems have different criticality requirements**.

- The **criticality level of functions** is assigned based on the **hazard and risk analysis**.
- The **three main factors** contributing to the ASIL level calculation are **severity**, **exposure**, and **controllability**.

Hardware Complexity

- The advanced hardware computational platforms consisting of several different processing units (e.g., CPU, GPU, FPGA).
- The heterogeneous attributes of computing systems can introduce unpredictable variations and thus give rise to uncertainties in system timing.

Challenges

- Autonomous system is complex and safety-critical embedded systems with strict real-time and resource constraints as well as having a deep processing pipeline with strong dependencies across functions.
- Tasks with different criticality levels are integrated on the same hardware platform. As a mixed-criticality system (MCS), the scheduling strategy should guarantee HI-criticality tasks without any risks introduced by LO-criticality tasks.
- Once a timing fault (e.g., overrun) happens in any HI criticality task, the survivability of LO criticality task should be considered to improve the robustness and resilience of MCS task scheduling.

Static scheduling is regarded as completely deterministic and is well supported in the industry. Thus, we adopt **a static scheduling method** and use **Directed Acyclic Graphs (DAGs)** to model task precedence constraints.

Problems of Existing Methods

- Most static scheduling work considering task dependencies does not consider the survivability of low criticality tasks.
- The schedules for high and low modes are different. More efforts are needed to check the safety of schedules during mode change.

Target of this work against existing problems:

- Generates one consistent schedule considering the survivability of LO tasks to realise task-level mode change and significantly improves system resilience.
- Reduces the complexity of task-level mode change, which can accelerate the recovery of specific impacted LO task.

Proposed method

Mixed-Criticality Task Model

We adopt dual-criticality system considering criticality-dependent WCET estimation (i.e., for HI critical tasks C(LO) < C(HI)). A task $\tau_i$can be defined by the tuples (T_i, D_i, C_i(HI), C_i(LO), L_i).

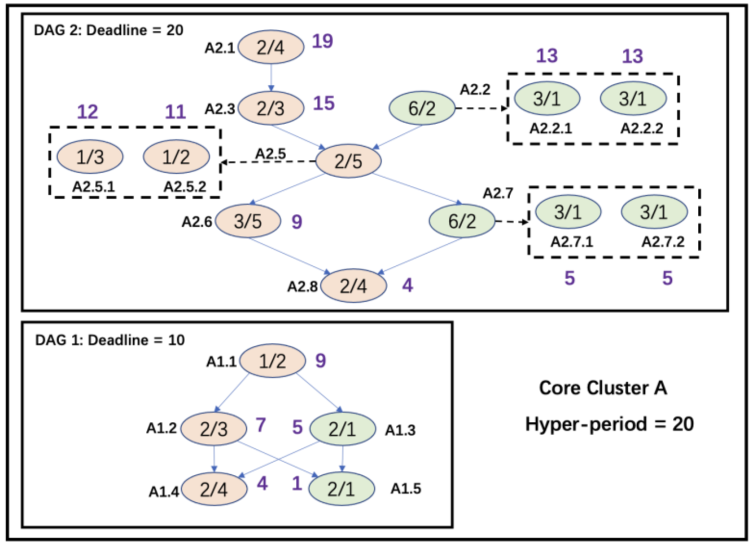


Fig. 2: Mixed-criticality DAGs example (Green nodes: LO-tasks; Red nodes: HI-tasks)

Consistent Mixed-Criticality DAGs Scheduling

Start from the last time point of the hyperperiod and the last layer of DAGs in the system. All HI critical tasks can be executed as late as possible and scheduled start time will be kept the same in both system modes.

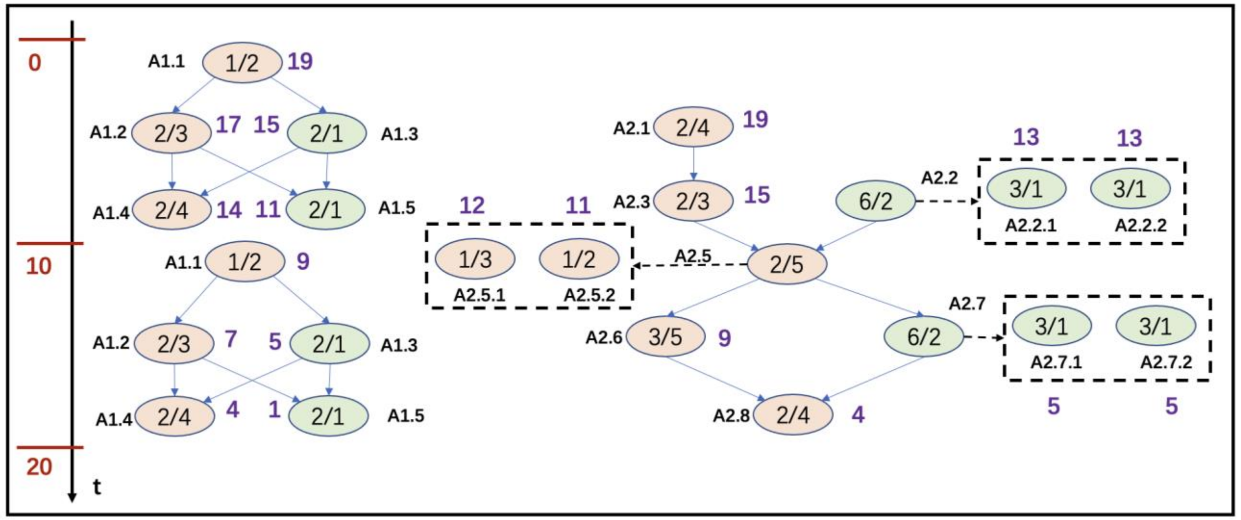


Fig. 3: The execution of DAGs

Proposed method

Schedule Calculation

Step 1. Schedule calculation in HI mode:

The schedule should be generated based on the HI mode task behaviour to guarantee the execution of HI tasks. The existence of LO criticality tasks can assist in anchoring the target schedule region of LO tasks in LO mode.

Step 2. Consistent schedule generation:

Based on the schedule calculated in step 1, all tasks are performed with LO mode behaviour in this step. The execution time of HI tasks is shortened, and more time is freed to schedule LO tasks.

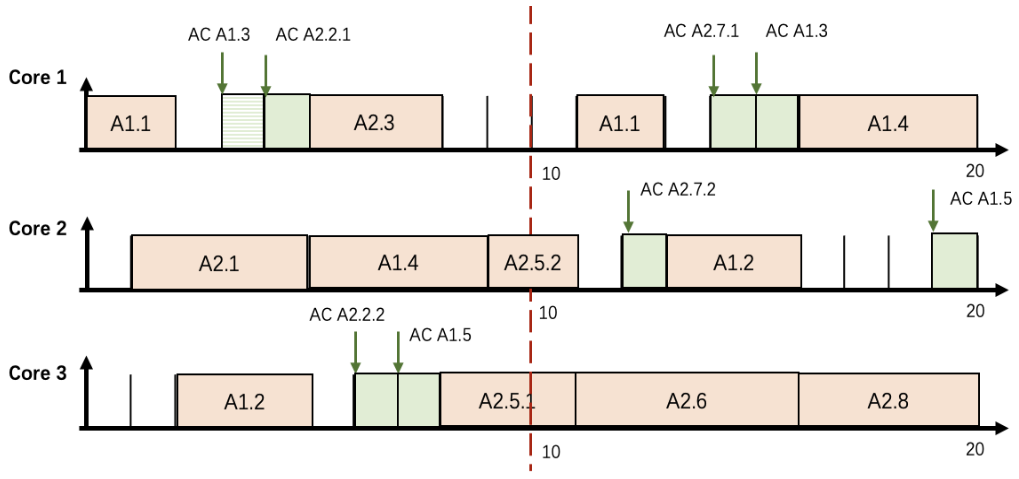


Fig. 4: The schedule generated by step 1

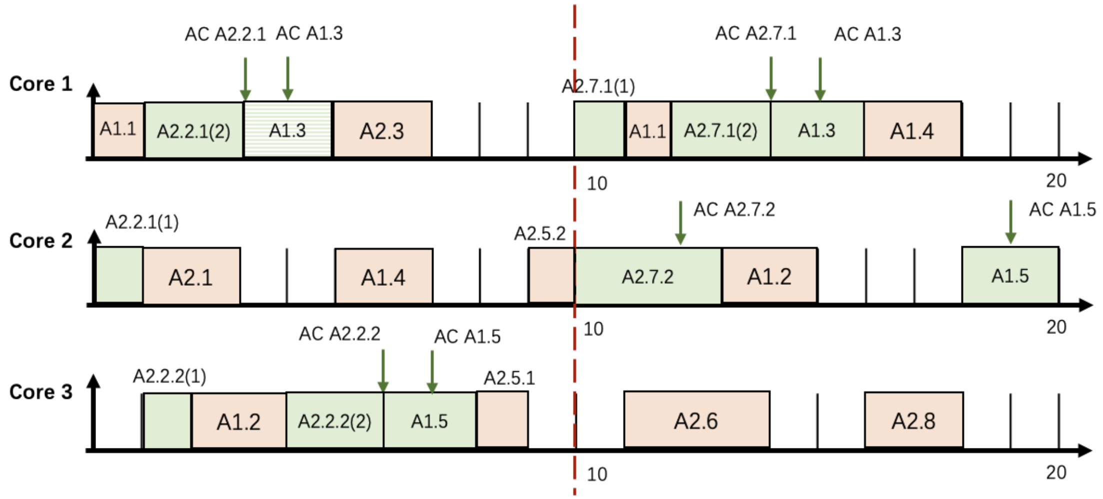


Fig. 5: The consistent schedule from step 2

RTS York HISE Group UNIVERSITY of York

Future Work Plan

The benefits brought by our method will be further verified based on large scale simulation. Furthermore, it will be applied to more realistic examples – as we are doing with a mobile delivery robot.

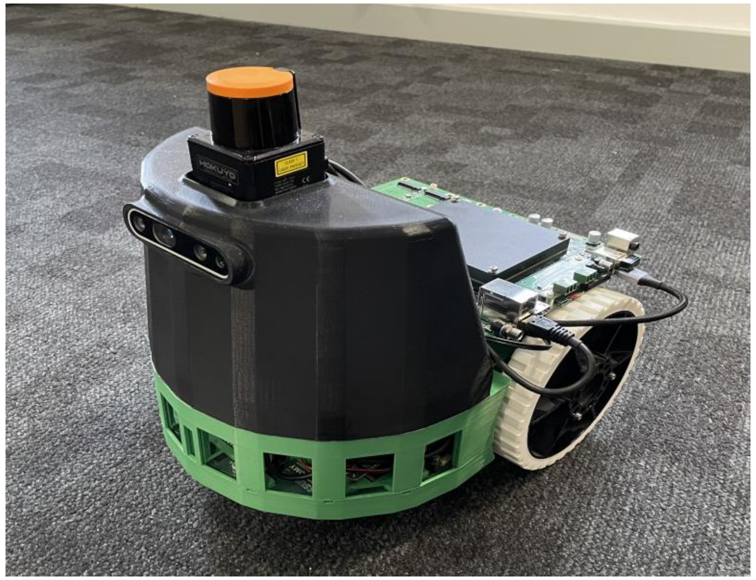
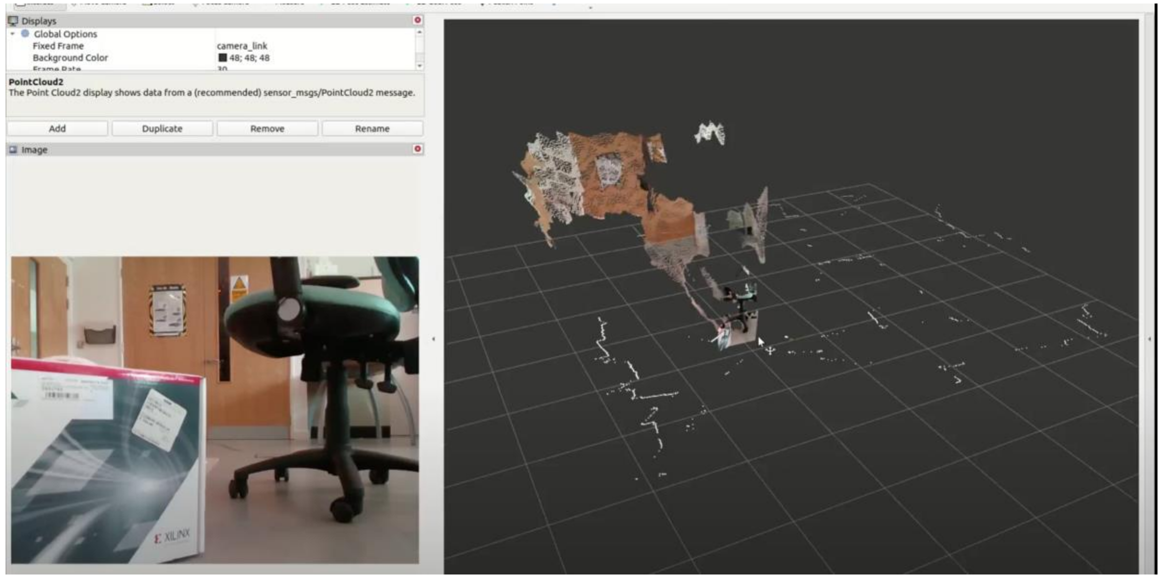
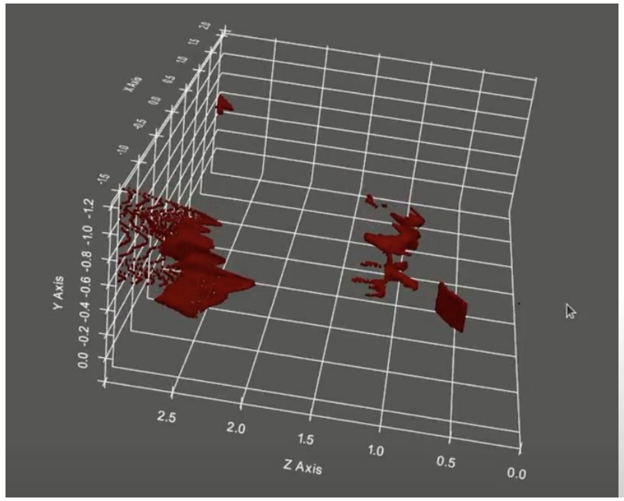


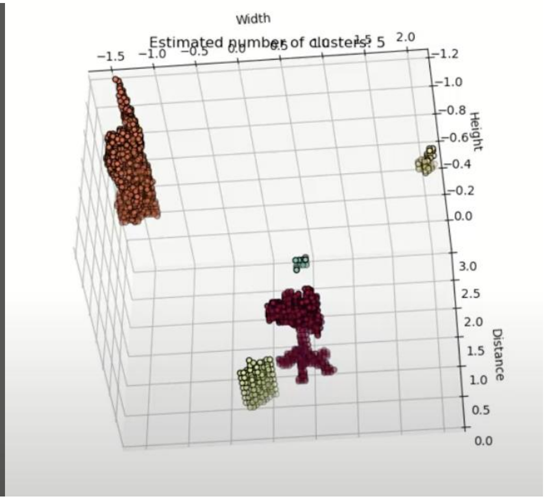
Fig. 4: ISA Robot from AAIP[1] Project

[1] **Assuring Autonomy International Programme** (**AAIP**) https://www.york.ac.uk/assuring-autonomy/about/



Displays
Global Options
Fixed Frame camera_link
Background Color 48; 48; 48
PointCloud2
The Point Cloud2 display shows data from a (recommended) sensor_msgs/PointCloud2 message.





Thank You!