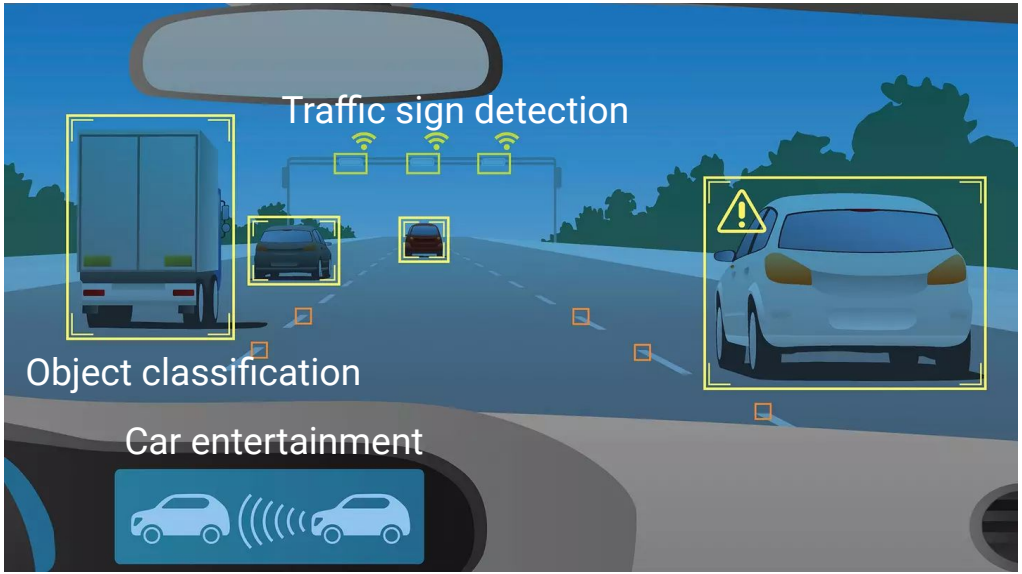
PAStime: Progress-aware Scheduling for Time-critical Computing

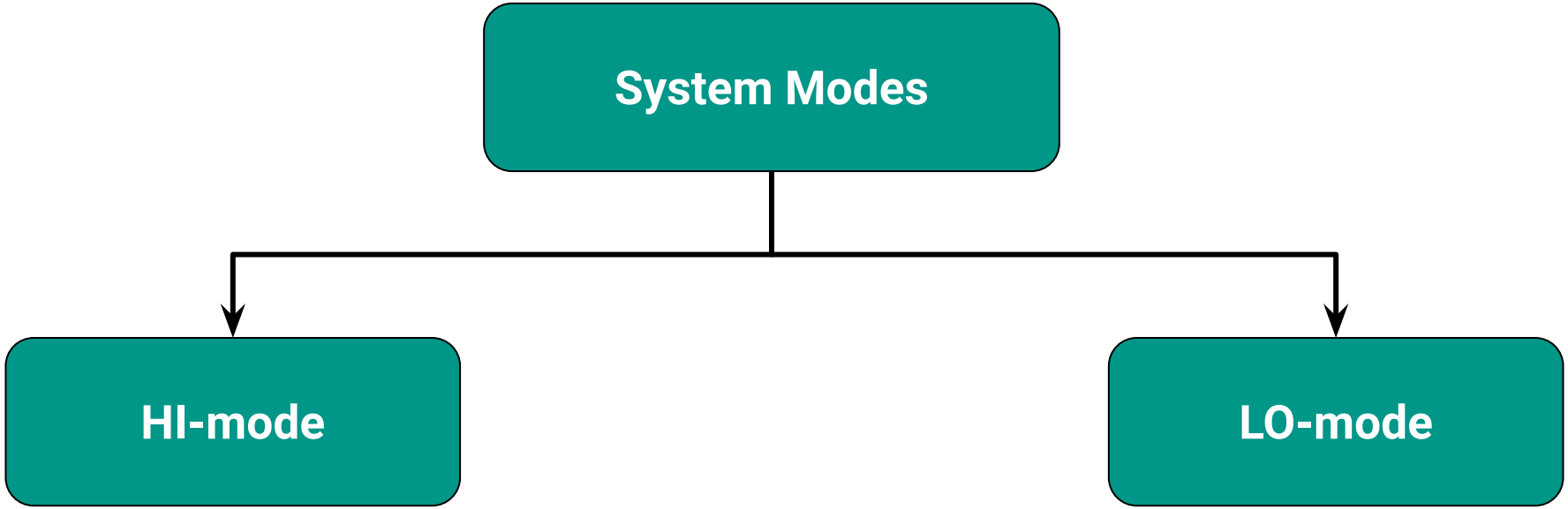
Soham Sinha, Richard West, Ahmad Golchin
Department of Computer Science, Boston University, USA

Introduction - Mixed-criticality Systems

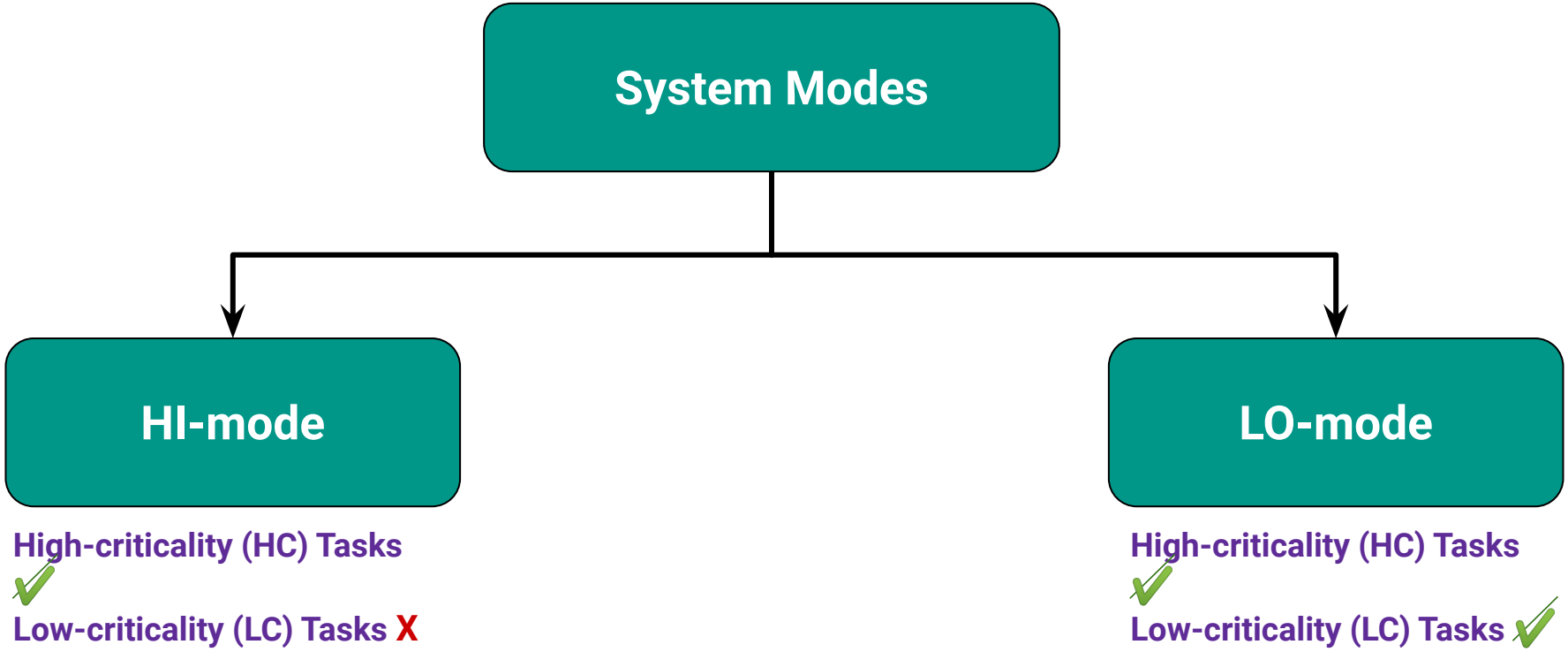
Unmanned Aerial Vehicles



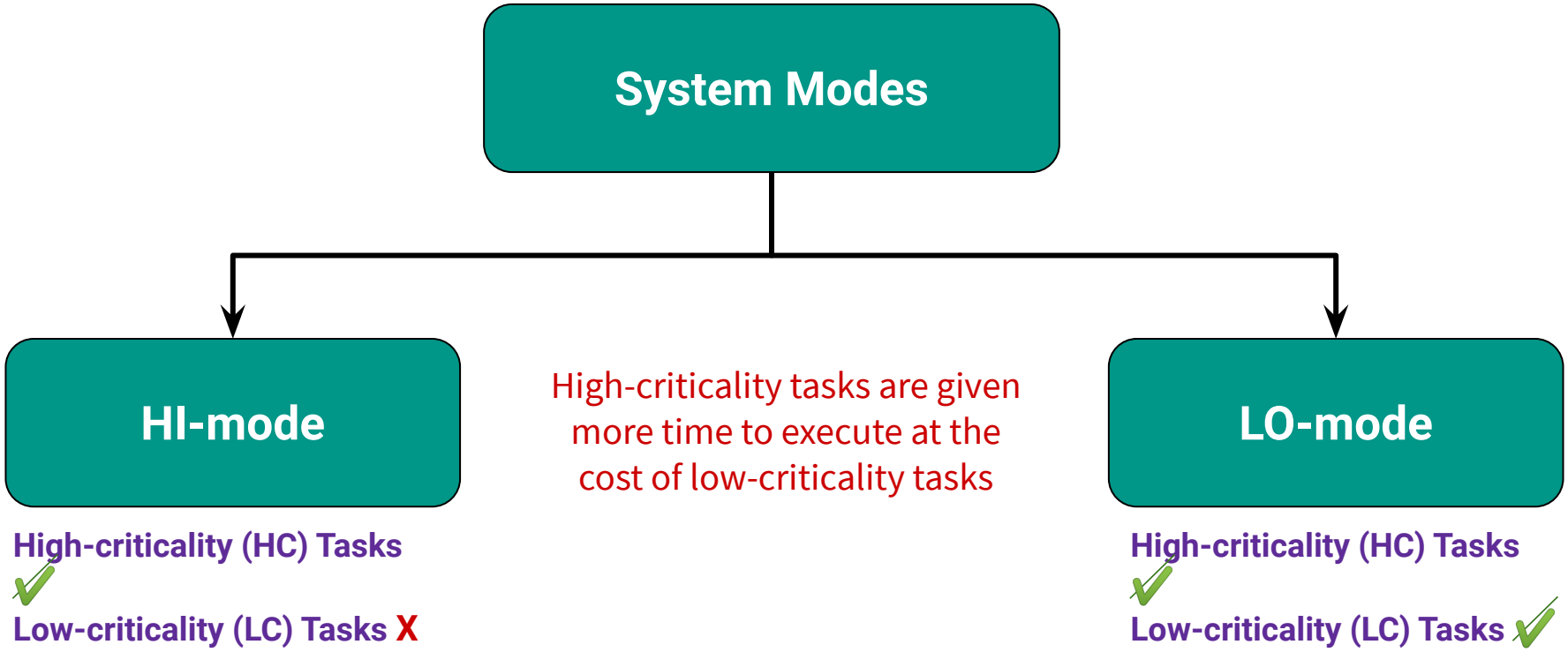
Background - MC Task Scheduling



Background - MC Task Scheduling



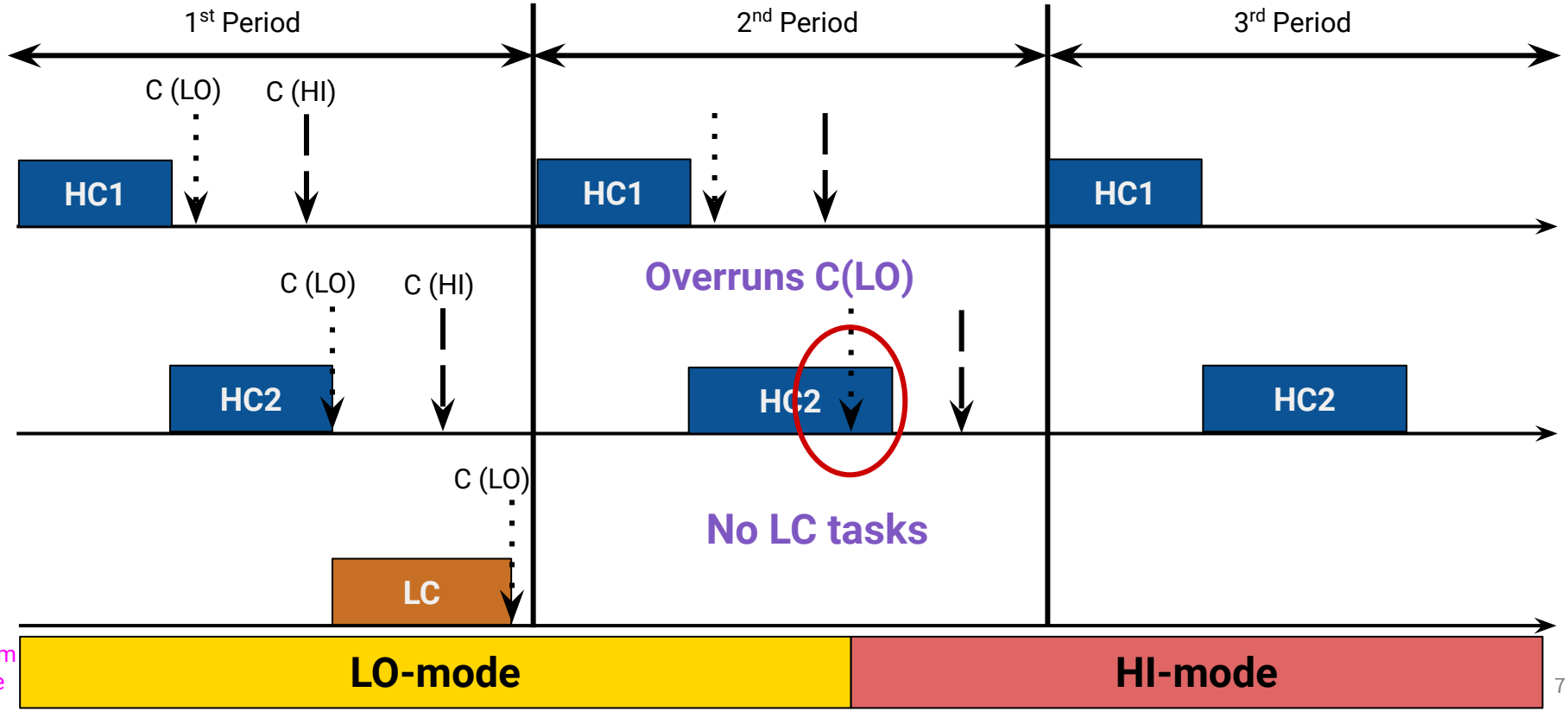
Background - MC Task Scheduling



Adaptive Mixed-criticality (AMC) Scheduling

1. The system starts in LO-mode.
    - All tasks run with their LO-mode budgets.
2. When a task overruns its LO-mode budget, system mode is switched to HI-mode.
3. In HI-mode, only high-criticality tasks get to run.

AMC Scheduling - A Simple Example



Limitations of AMC

- Although task deadlines are honored, LC tasks are dropped in HI-mode.
- A small delay in a HC task could overrun its LO-mode budget.
  - System is switched to HI-mode.
- Frequent switch to HI-mode will drop LC tasks more frequently as well.
- Quality-of-service of the LC tasks is degraded by premature or unnecessary switches to HI-mode.

Prior Solutions to improve AMC

- Stretch the period.
- Use reduced HI-mode budget for low-criticality tasks.
- Static calculation of slack.

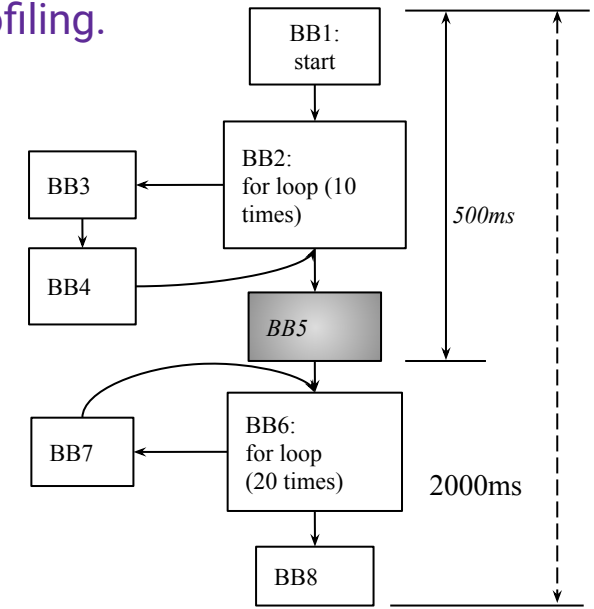
- **Improve AMC by using runtime progress.**
  - **Reducing the number of mode switches**
  - **Increasing the execution time for LC tasks**
  - **Improve QoS of LC tasks while guaranteeing HC tasks' deadlines.**

PAStime: Progress-aware Scheduling

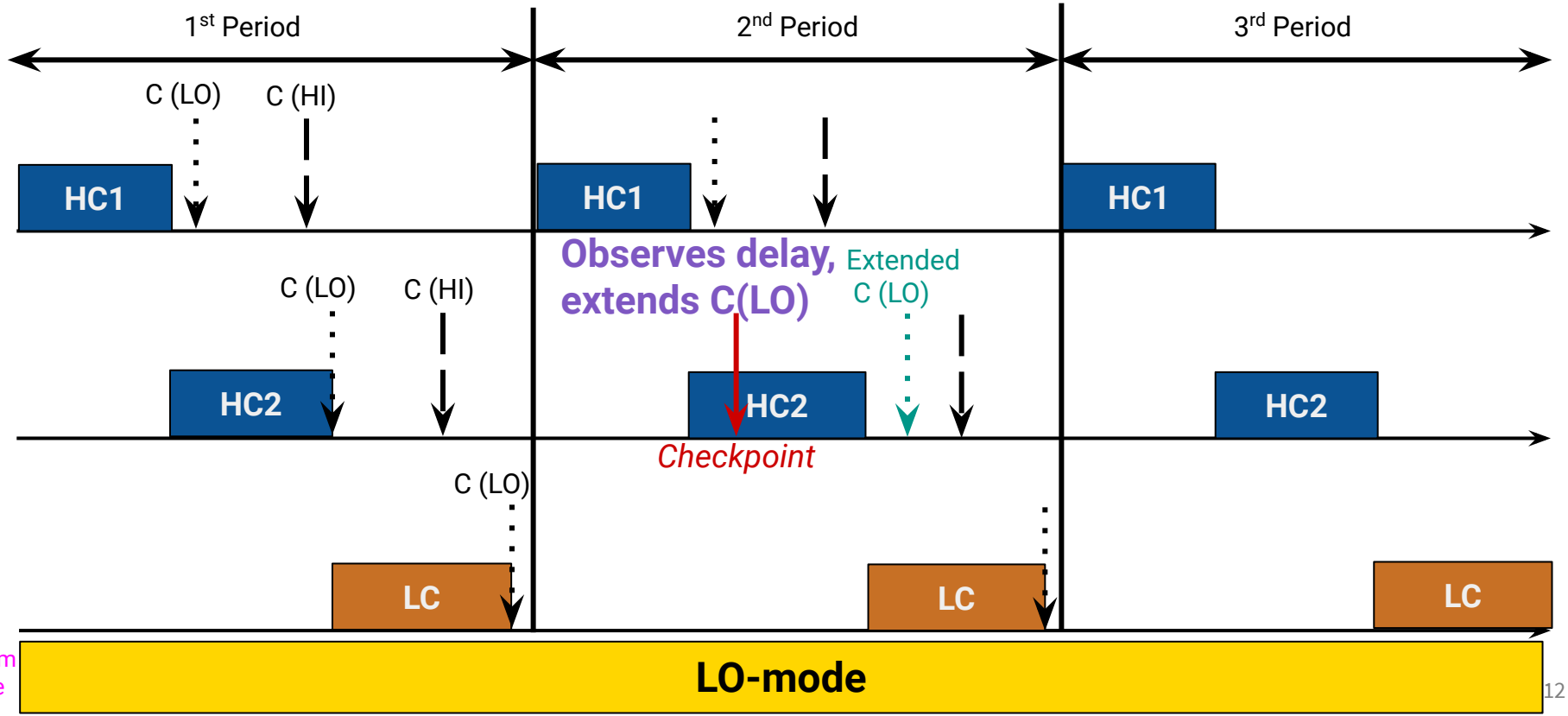
PAStime Runtime System

- Add checkpoints in a high-criticality program's source code.
- Measure progress at the checkpoints in LO-mode by profiling.

- At runtime, if a HC task is delayed at a checkpoint
  - Check if C (LO) could be extended, without breaking schedulability of other tasks.
- Keep the system in LO-mode, if the task finishes within extended C (LO)
  - Otherwise, switch to HI-mode

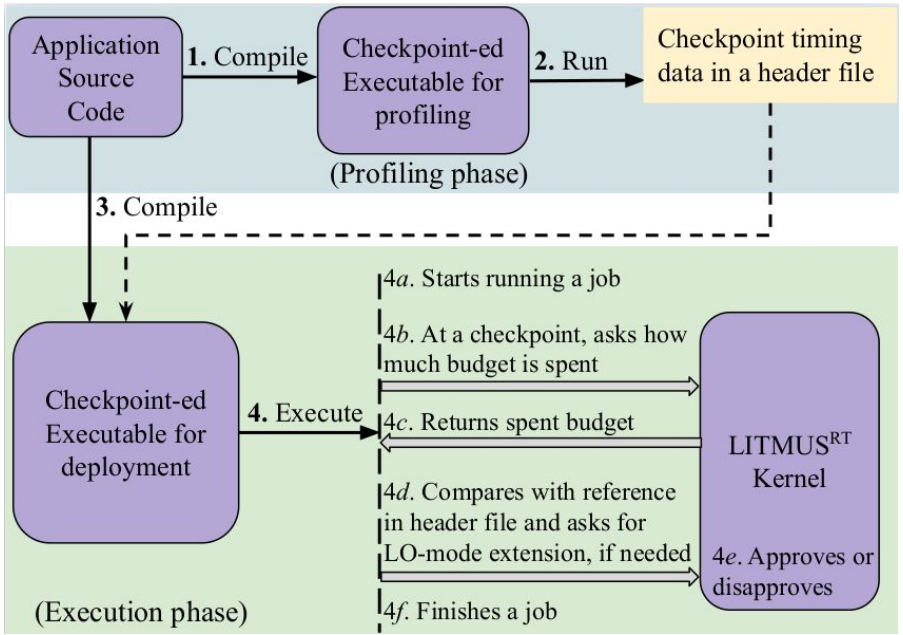


AMC-PAStime: AMC extended with PAStime



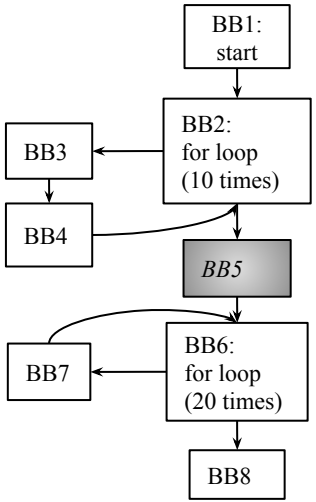
Implementation of PAStime

- Two phases
  - Profiling phase
  - Execution phase
- Runtime implementation in LITMUS^{RT}
  - First implementation of AMC in LITMUS^{RT}
  - Both AMC and AMC-PAStime In LITMUS^{RT}



Checkpoint Instrumentation

- Manual Checkpoint Instrumentation
- Automatic Checkpoint Instrumentation for Profiling phase
  - Insert checkpoint before a loop (*except the first*)
  - Implemented in LLVM

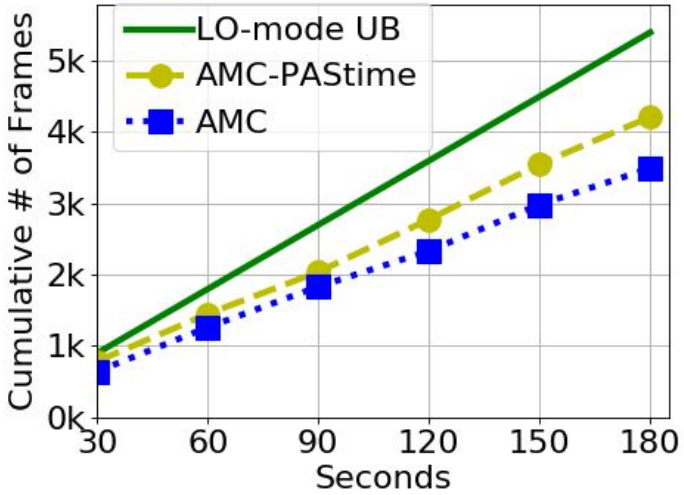


Evaluation

An Overview

- **Platform:** Intel NUC Kit (Intel Core i7-5557U 3.1 GHz)
- **Applications:** Darknet Object Classification (*HC*), dlib Object Tracking (*HC*), MPEG Video Decoder (*LC*)
- **Metrics:** QoS, Scalability (*2-20 tasks*), Flexibility in LO-mode utilization, Checkpoint location, Overheads, Prediction Models

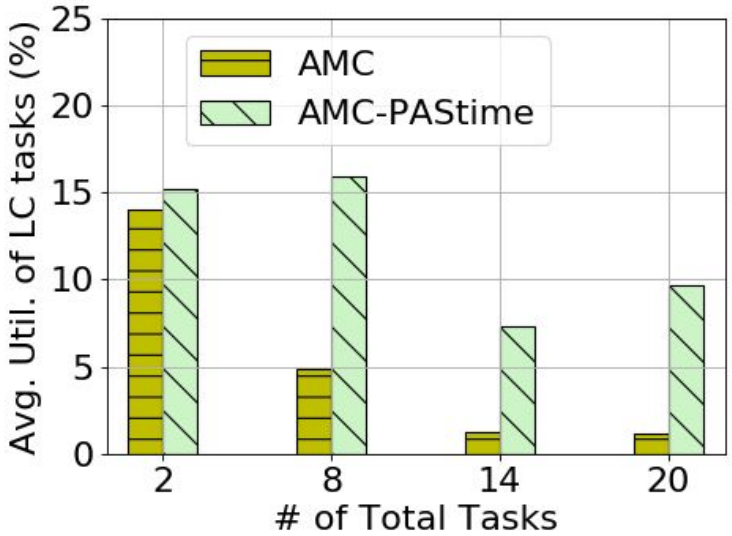
QoS of A Low-criticality Task



Two Tasks
One HC Object Classifier
One LC Video Decoder

9-21% increment in decoded frames

Scalability - 2 to 20 Tasks

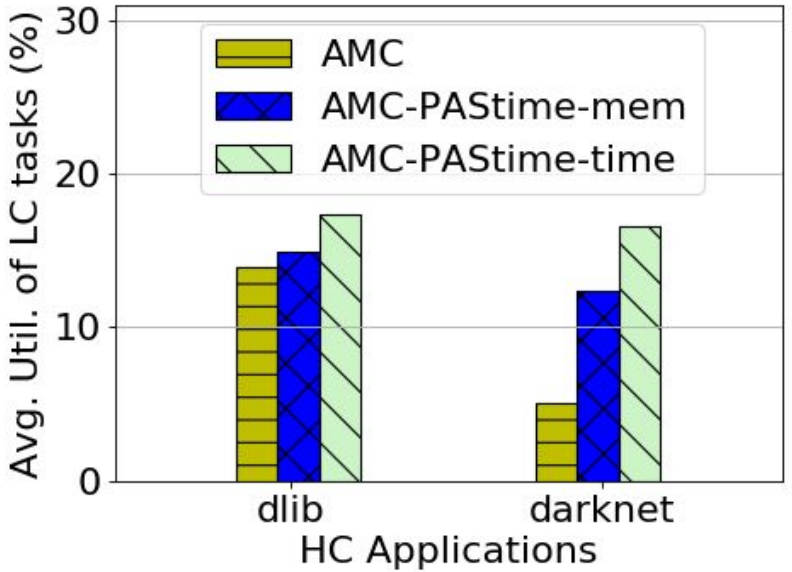


Half the task in each set are HC Object Classifier tasks and half are LC Video Decoder tasks

Utilizations of LC tasks is improved by **a factor of 3 to 9** for 8 to 20 tasks.

Two Prediction Models

- Prediction based on linear extrapolation of delay
- Prediction based on Memory Access Time



Conclusion and Future Work

PAStime is a *mixed-criticality runtime system* to extend the LO-mode based on the execution progress of the HC tasks. PAStime is implemented using **LLVM** and **LITMUS^{RT}**.

- Explore other prediction models such as the feedback-based one.
- Applications of PAStime in timing-sensitive cloud-computing applications.
- In Quest RTOS, VCPU budget could be extended based on observed delay at a checkpoint, given that RMS schedulability criteria is met.

Thanks You!

Questions?

Contact: soham1 <AT> bu.edu