Worst-case Scheduling of Software Tasks

A Constraint Optimization Model to Support Performance Testing

Stefano Di Alesio [1,2]

Shiva Nejati [2]

Lionel Briand [2]

Arnaud Gotlieb [1]

CP 2014

Lyon, 10/09/1914

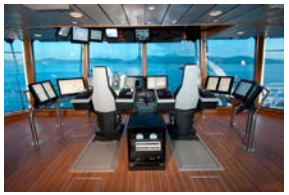
Certus
Centre for Software Verification & Validation

SnT
securityandtrust.lu

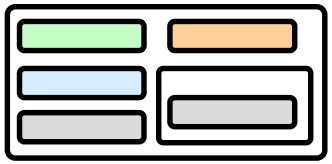
[1] Certus Centre for Software V&V
Simula Research Laboratory
Norway

[2] Interdisciplinary Centre for Reliability, Security and Trust (SnT)
University of Luxembourg
Luxembourg

We present a Constrained Optimization Model to support Performance Testing in RTES



Performance Requirements vs. Real Time Embedded Systems (RTES)



Supporting Performance Testing: A novel application for COPs

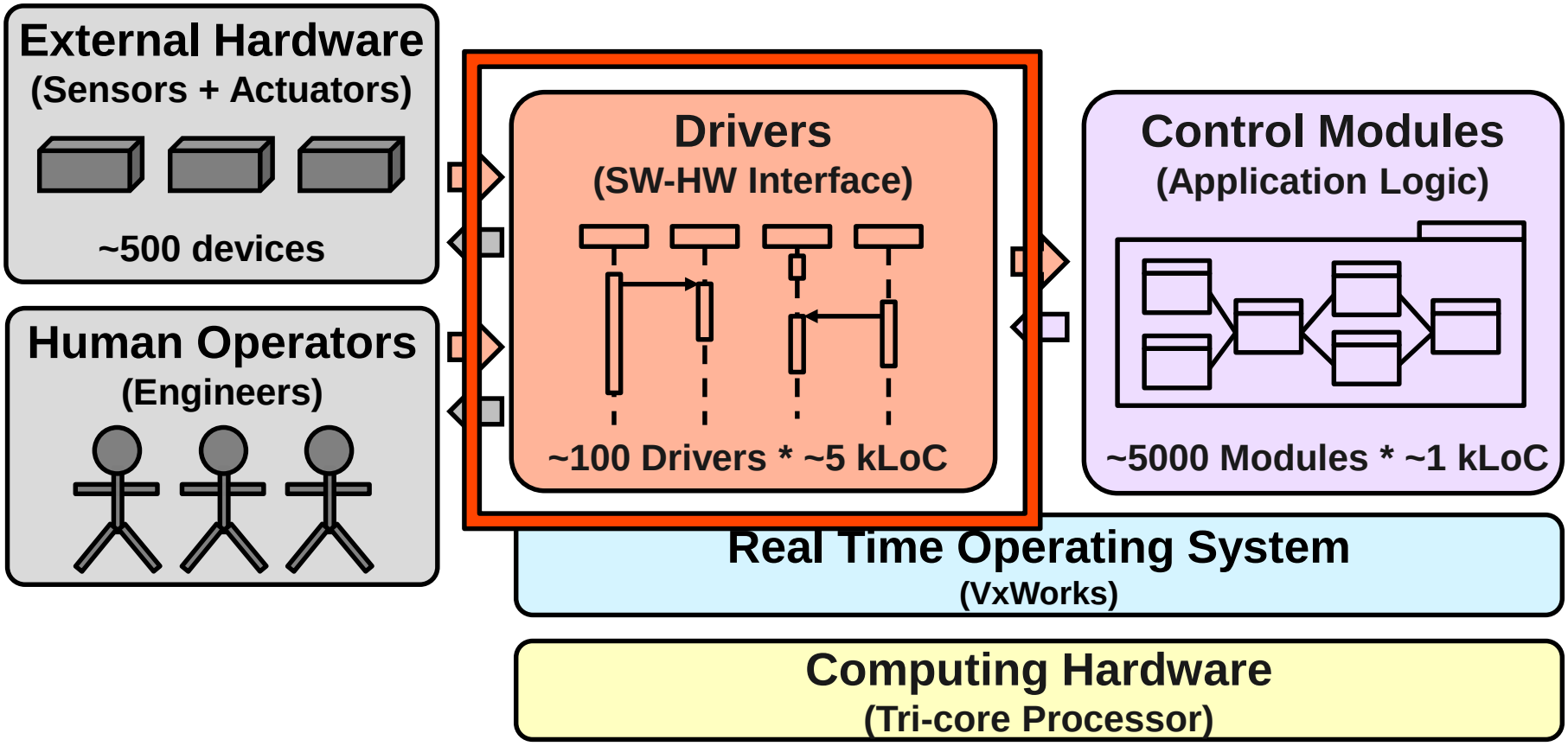
Industrial Experience: Context, Process and Results

RTES are typically safety-critical, and thus bound to meet strict Performance Requirements

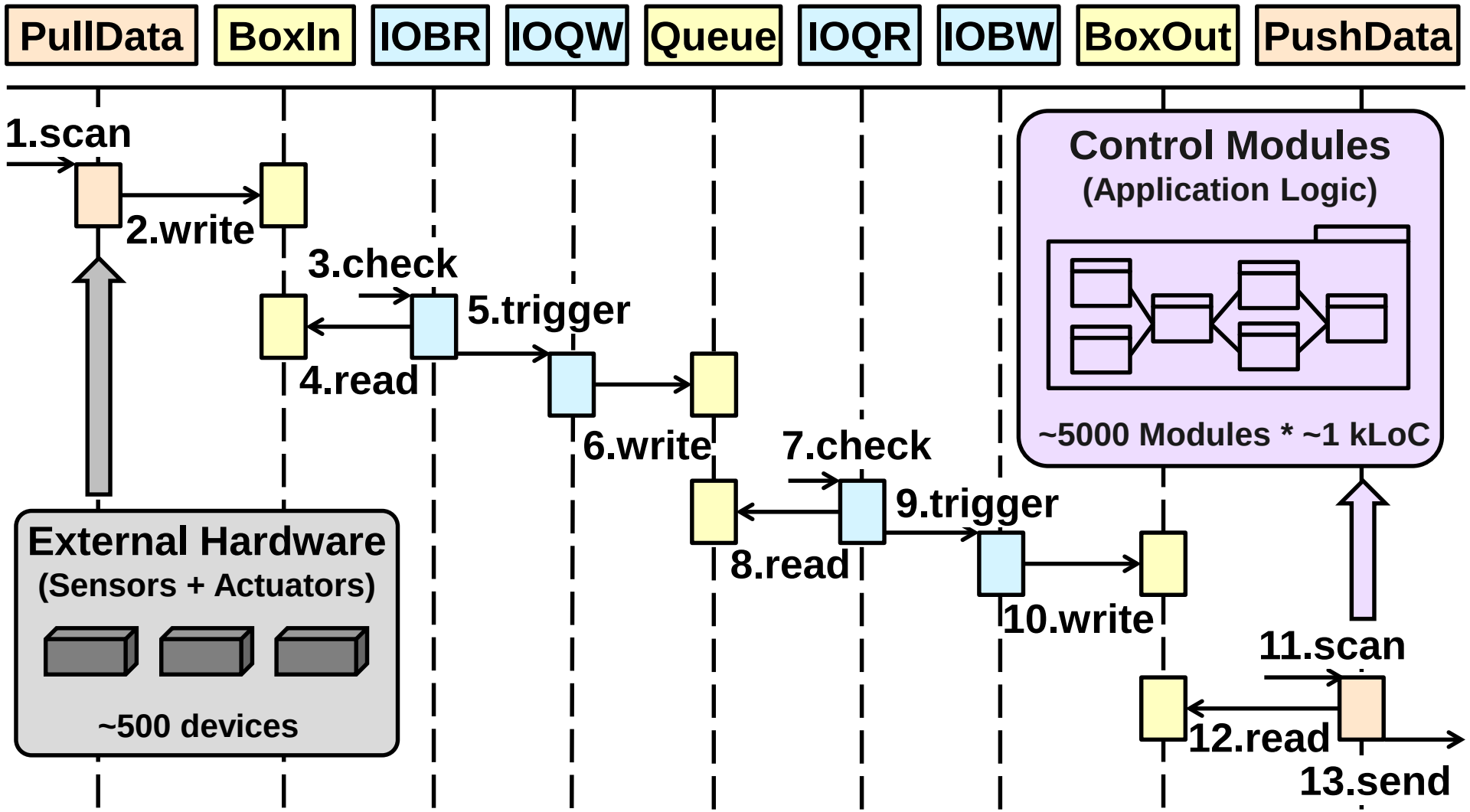
Our case study is a monitoring application for fire/gas leaks detection in offshore platforms

KONGSBERG

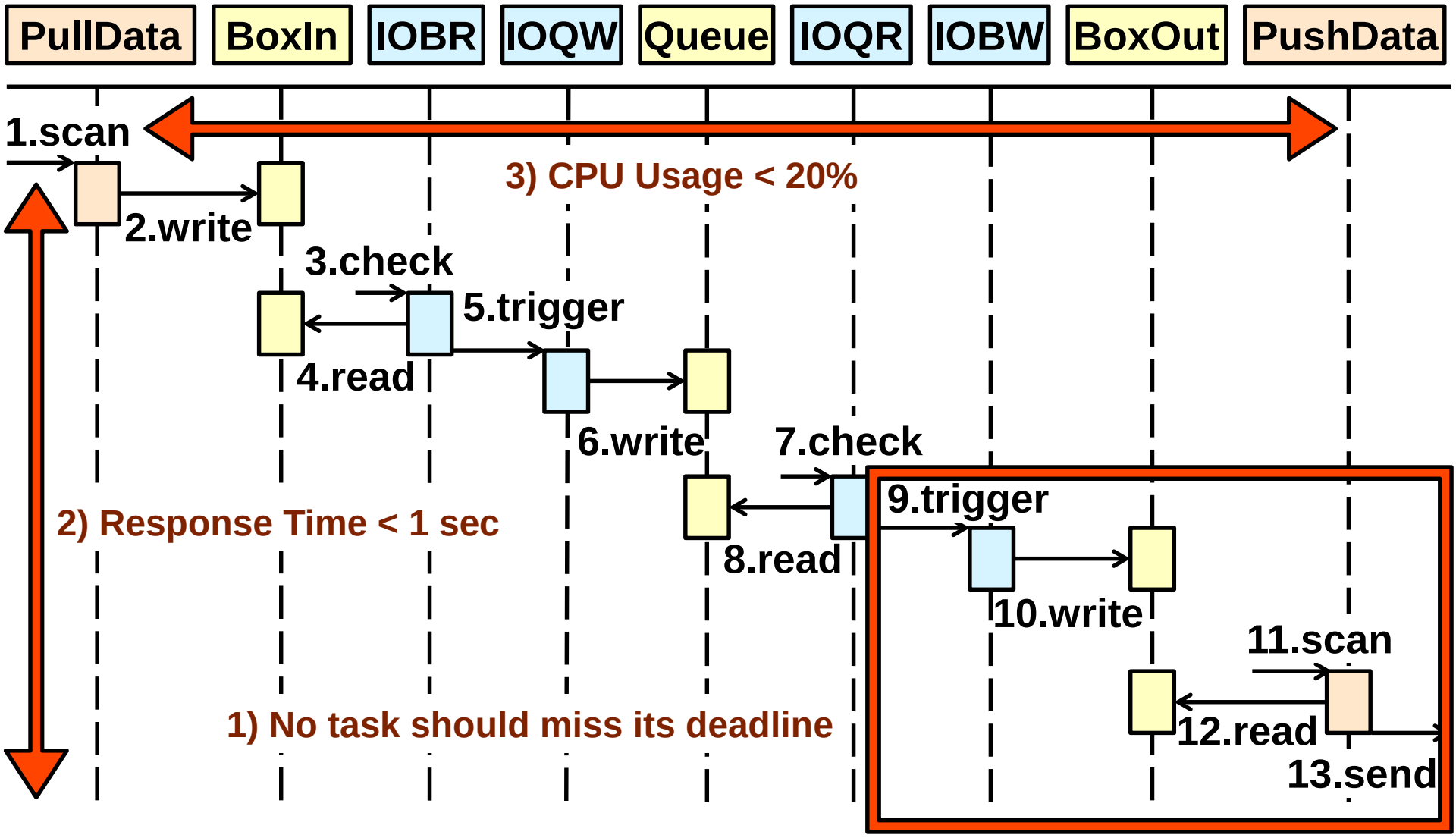
KM: Kongsberg Maritime
FMS: Fire and gas Monitoring System



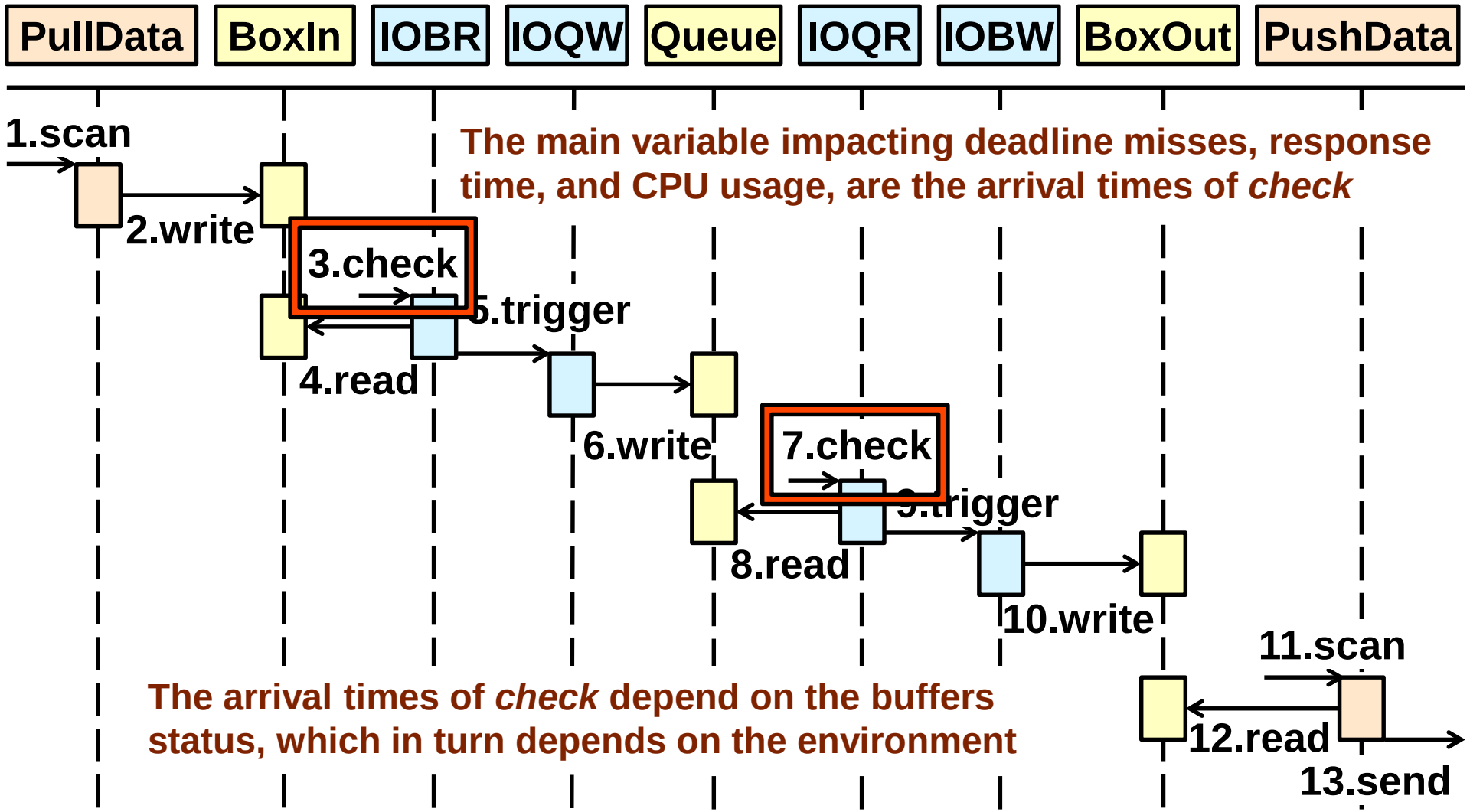
Drivers transfer data between external hardware (sensors and actuators) and control modules



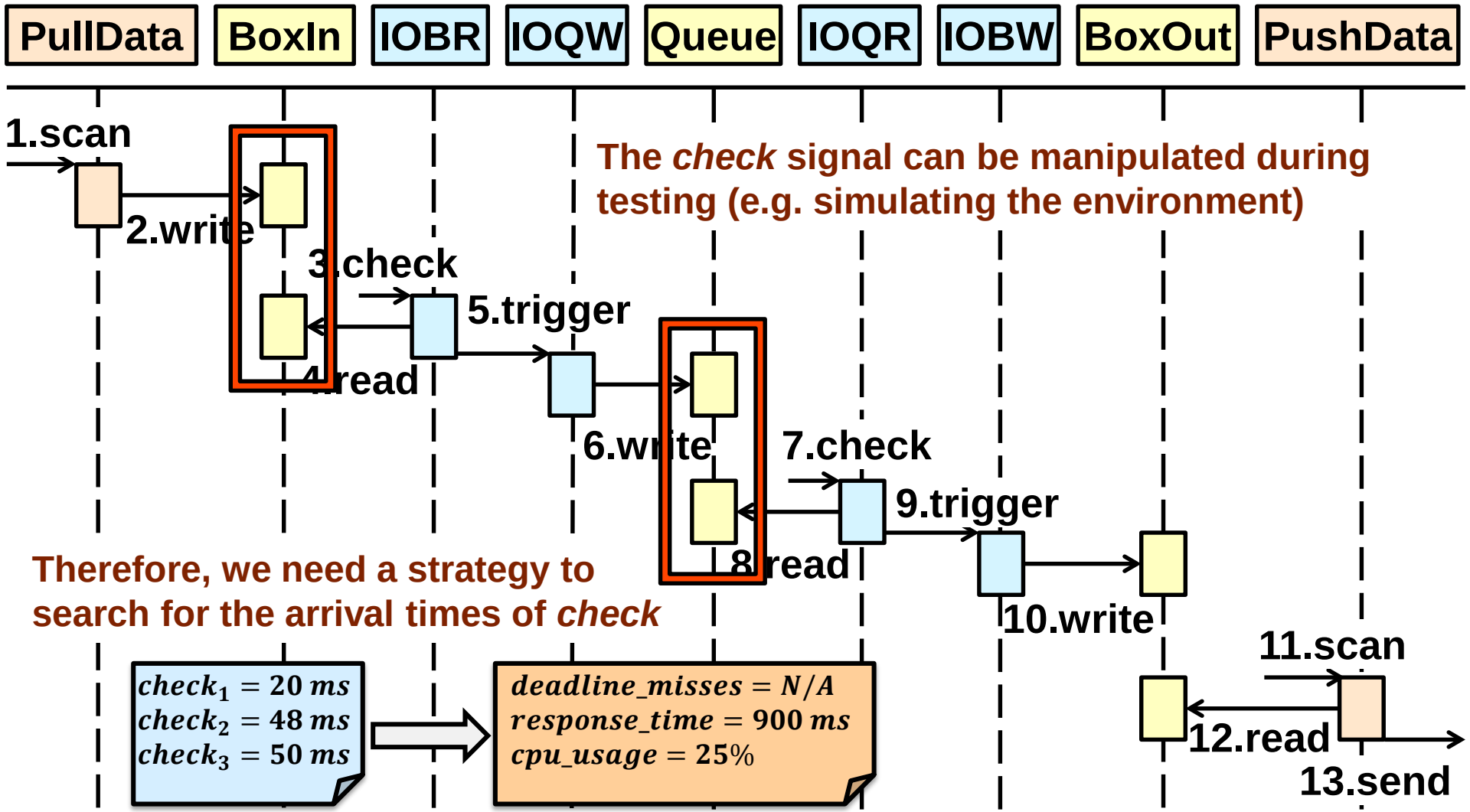
The FMS drivers have performance requirements on task deadlines, response time, and CPU usage



Our goal is to identify worst-case scenarios w.r.t. deadline misses, response time, and CPU usage



Each worst-case scenario can characterize a test case in terms of the arrival times of *check*



Several techniques have been used for solving similar problems, but each has its own weaknesses

SE	Verification		Testing	
	Schedulability Theory	**Model Checking**	**Performance Engineering**	**Genetic Algorithms**
Basis	Mathematical Theory	System Modeling	Practice and Tools	System Modeling
Background	Queuing Theory	Fixed-point Computation	Profiling, Benchmarking	Meta-Heuristic Search
Key Features	Theorems	Graph-based, Symbolic	Dynamic Analysis	Non-Complete Search
Weaknesses	Assumptions, Multi-Core	Complex Modeling	Non Systematic	Low Effectiveness

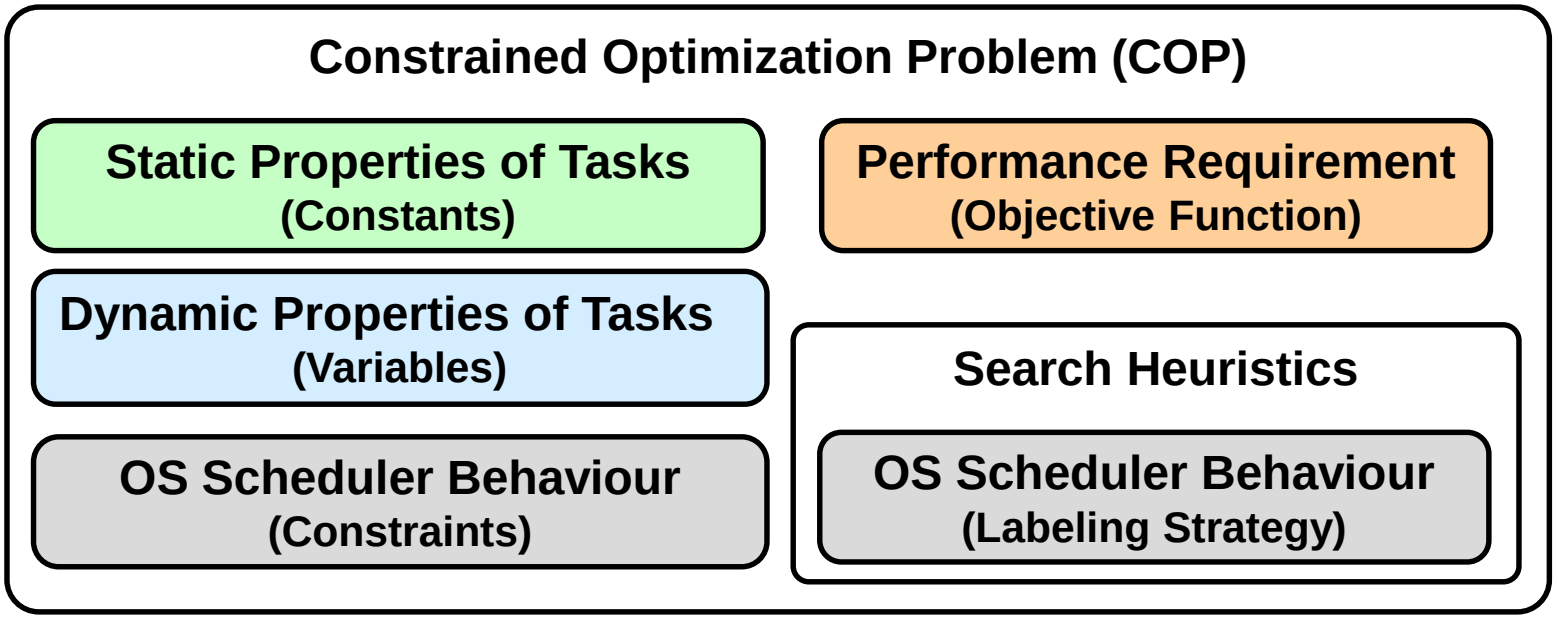
CP

Job-shop schedulability analysis

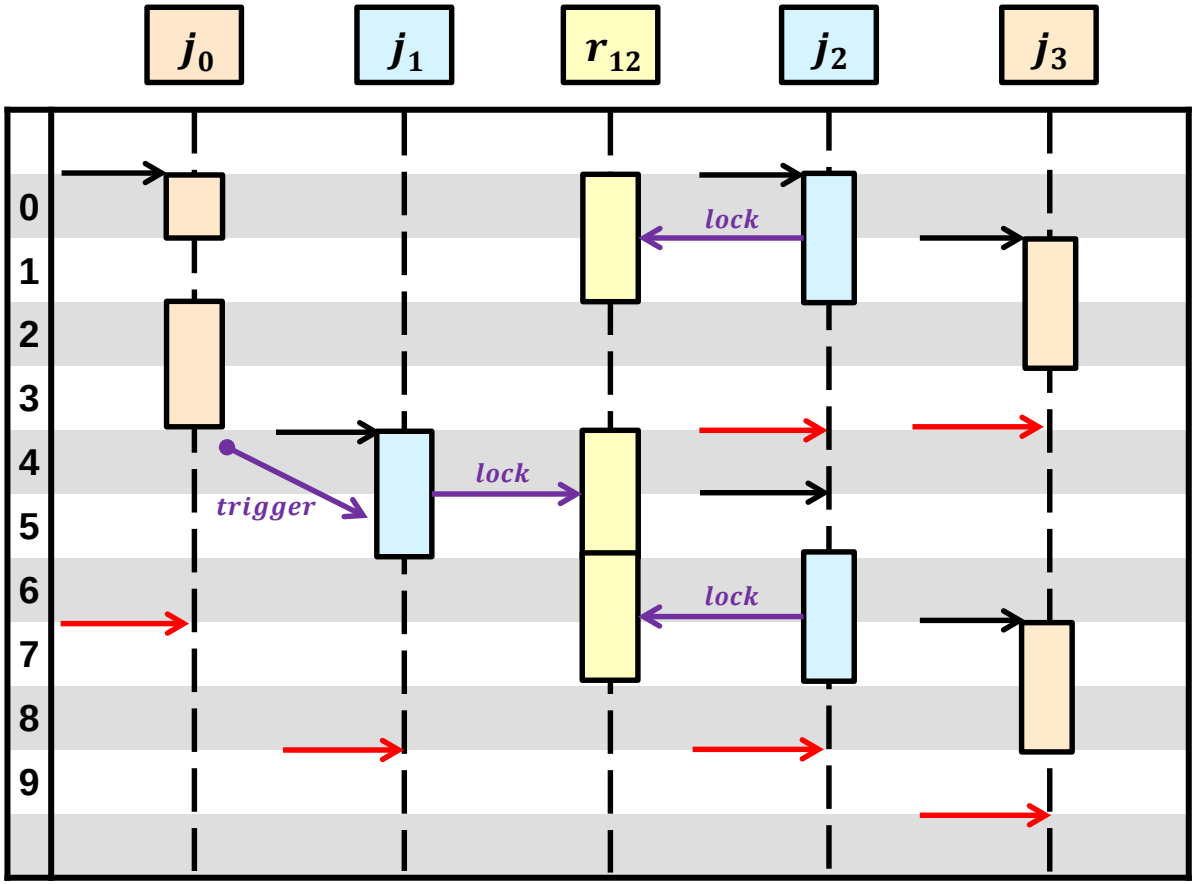
Multi-core
Priority
Preemption
Dependency
Triggering

We cast the search for the arrival times of the *check* signal leading to worst-case scenarios as a COP

The COP models a multi-core priority-driven preemptive scheduler with task triggering and dependencies



Static Properties depend on the FMS design, and are modeled as Constants

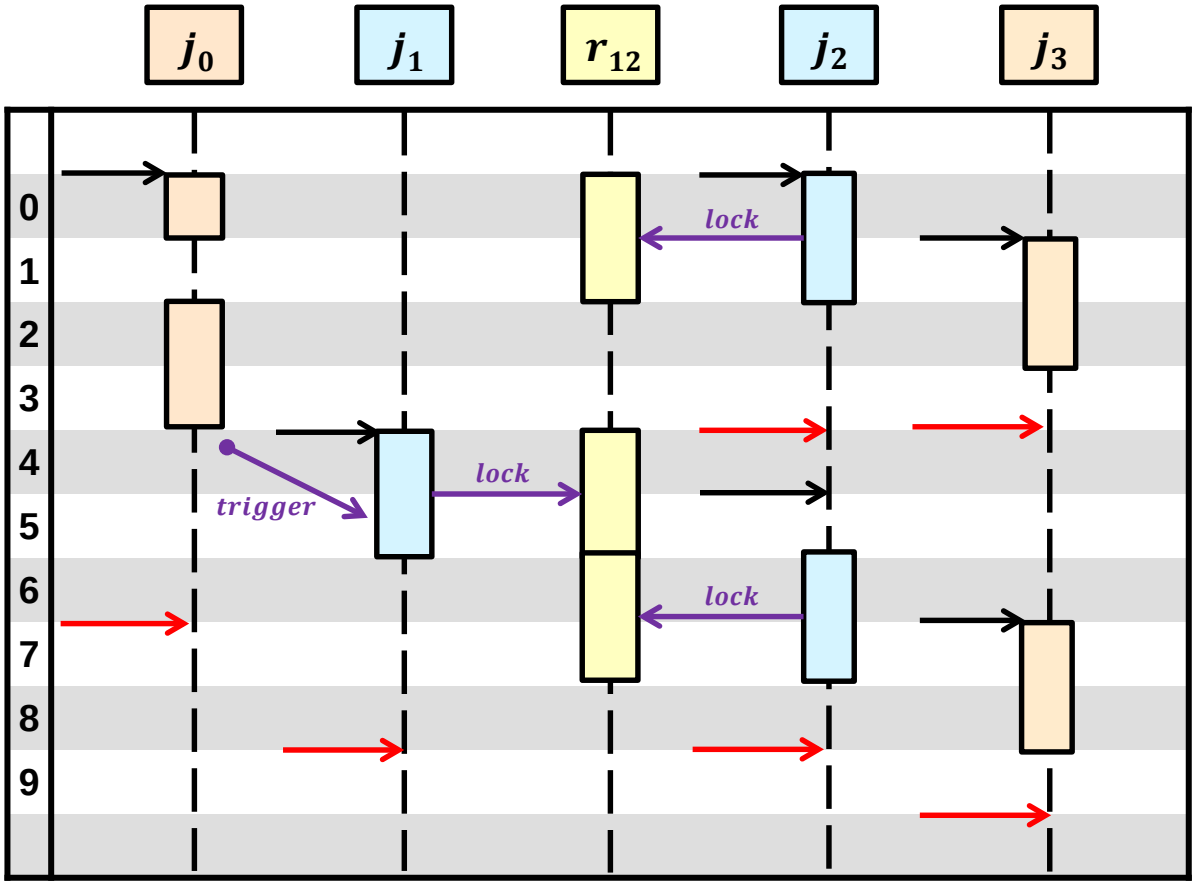


Constants

- Observation Interval: $T = [0, 9]$
- Number of cores: $c = 2$
- Set of Tasks: $J = \{j_0, j_1, j_2, j_3\}$
- Priority of Tasks: $pr(j_i) = i$
- Period of Tasks: $pe(j_2) = 5$
- Min/Max Inter-arrival time of Tasks: $mn(j_0) = 5, mx(j_0) = 10$
- Duration of Tasks: $dr(j_0) = 3$
- Deadline of Tasks: $dl(j_0) = 7$
- Triggering Relation: $tg(j_0, j_1)$
- Dependency Relation: $de(j_1, j_2)$
- Number of Periodic Task Executions: $te(j) = \left\lfloor \frac{tq}{pe(j)} \right\rfloor$, $te(j_2) = \left\lfloor \frac{10}{5} \right\rfloor = 2$

Time is discretized in our analysis: we solve an IP over finite domains

Dynamic Properties depend on the FMS runtime behavior, and are modeled as Variables (1/2)



Variables

Independent

- **Number of Aperiodic Task Exec.:**
  $te(j) \in \left[\frac{tq}{mx(j)}, \frac{tq}{mn(j)}\right]$,
  $te(j_0) \in [1,2], te(j_0) = 1$
- **Arrival time of Aperiodic Task Exec.:**
  $at(j,k) \in T,\ at(j_0,0) = 0,\ ac(j_3,1) = 7$
- **Active time of Task Executions:**
  $ac(j,k,p) \in T,\ p \in [0, dr(j) - 1]$,
  $ac(j_0,0,0) = 0,\ ac(j_0,0,1) = 2$

Dependent (1/2)

- **Set of Aperiodic Task Executions:**
  $K_j = [0,\ te(j) - 1], K_{j_0} = [0]$

Quantification over non-ground sets:

$$K = \left[0, \max_j \left(\frac{tq}{mn(j)}\right)\right]$$

$$\forall k \in K_j \cdot\ C(k) \leftrightarrow \forall k \in K \cdot \left(k < te(j)\right) \rightarrow C(k)$$

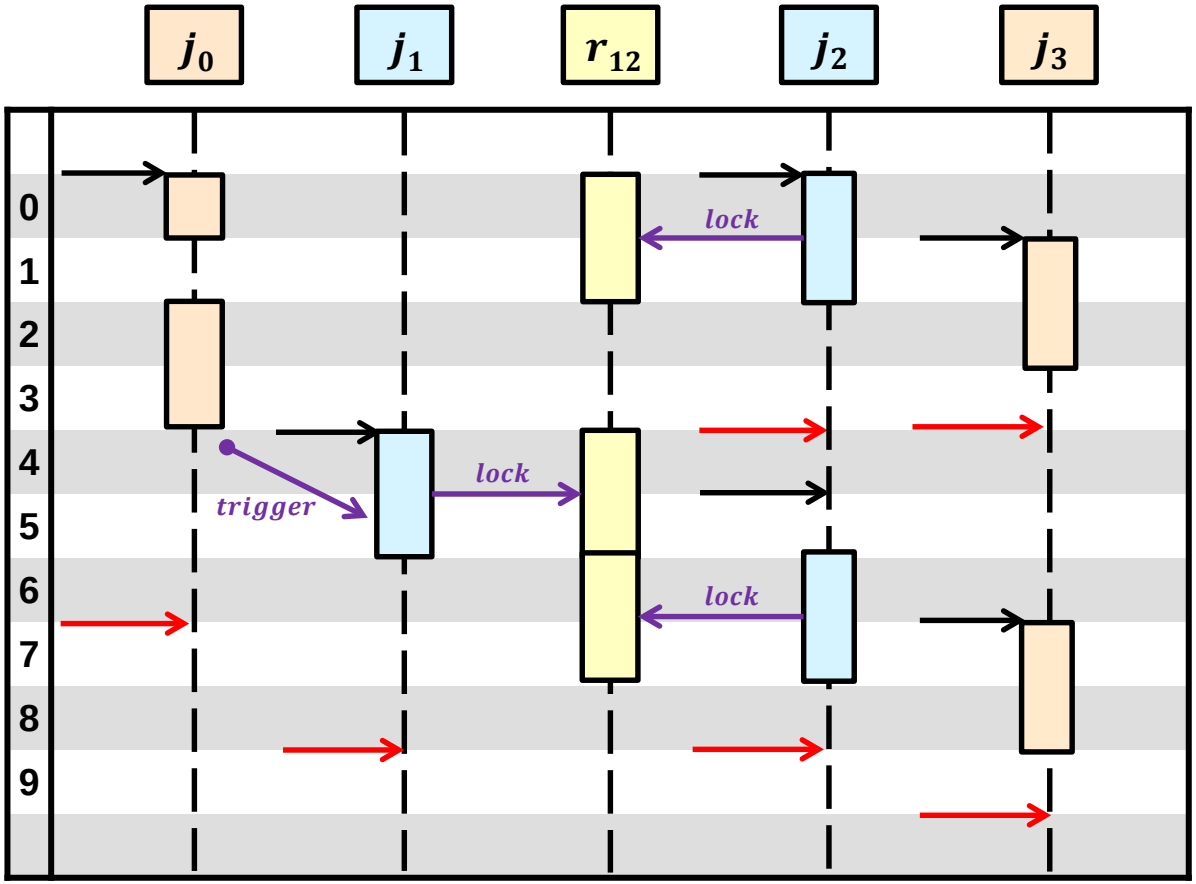
$$\sum_{k \in K_j} E(x) = \sum_{k \in K} \left(k < te(j)\right) \cdot E(x)$$

te **and** ***at*** **of Periodic Tasks Executions are constants:**

$$te(j) = \left\lfloor \frac{tq}{pe(j)} \right\rfloor, \qquad te(j_2) = \left\lfloor \frac{10}{5} \right\rfloor = 2$$

$$at(j,k) = k * pe(j), \qquad at(j_2,1) = 1 * 5 = 5$$

Dynamic Properties depend on the FMS runtime behavior, and are modeled as Variables (2/2)

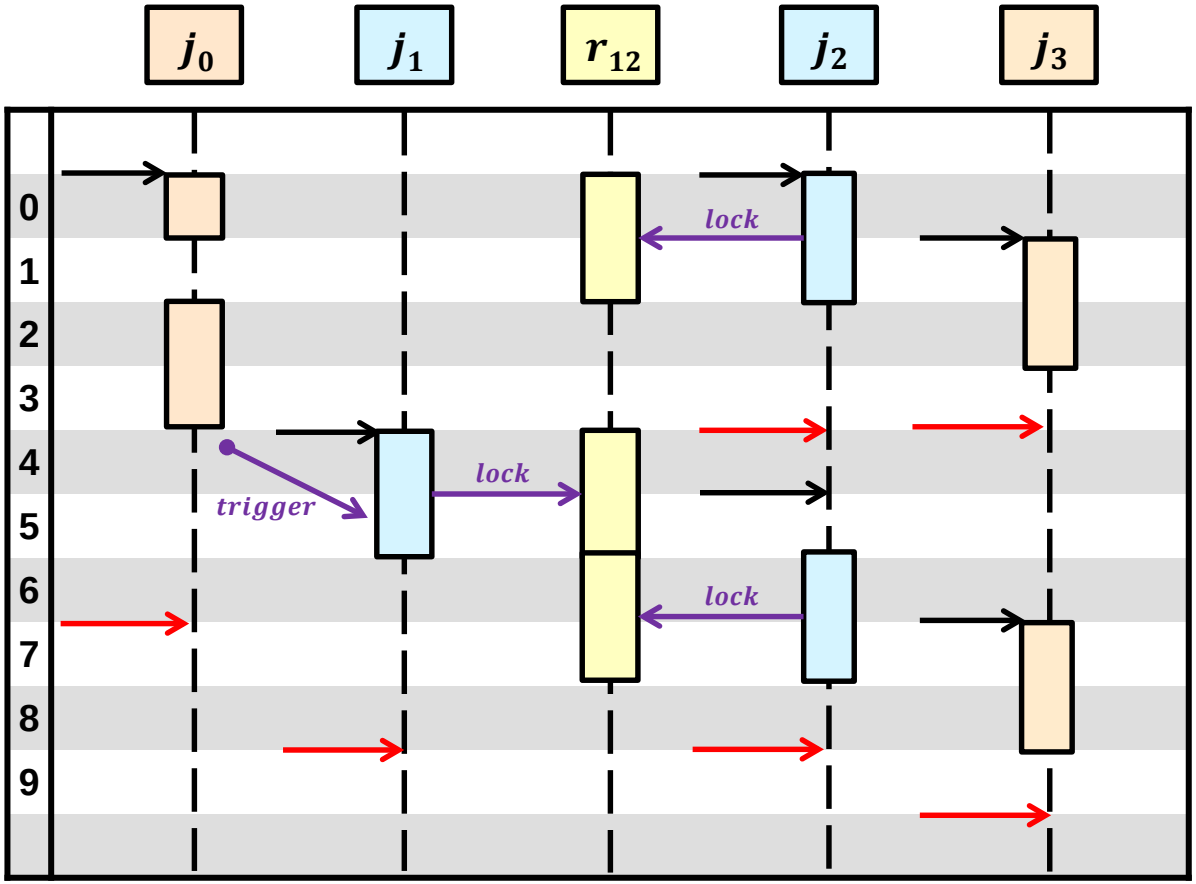


Variables

Dependent (2/2)

- **Start/End time of Task Executions:** $st(j,k) = ac(j,k,0)$, $st(j_0,0) = 0$, $en(j,k) = ac(j,k,dr(j)-1)$, $en(j_0,0) = 3$
- **Preempted time of Task Executions:** $pm(j,k,p) = ac(j,k,p) - ac(j,k,p-1)$, $pm(j_0,0,1) = 1$, $pm(j_0,0,2) = 0$
- **Waiting time of Task Executions:** $wt(j,k) = st(j,k) - at(j,k)$, $wt(j_2,0) = 0$, $wt(j_2,1) = 1$
- **Deadline of Task Executions:** $ed(j,k) = at(j,k) + dl(j)$, $ed(j_0,0) = 0 + 6 = 6$
- **Deadline Miss of Task Executions:** $dm(j,k) = en(j,k) - ed(j,k)$, $dm(j_0,0) = 3 - 6 = -3$
- **System Load:** $ld(t) = \sum_{j,k,p}(ac(j,k,p) = t)$, $ld(0) = 2$, $ld(3) = 1$

The Performance Requirements of the FMS are modeled as objective functions to maximize



Objective Function

- **Deadline Misses:**

$$F_{DM} = \sum_{j,k} 2^{dm(j,k)},$$

$$F_{DM} = 2^{-3} + 2^{-3} + 2^{-2} + 2^{-1} + 2^{-1} + 2^{-1}$$

- **Response Time:**

$$F_{RT} = \max_{j,k}\big(en(j,k)\big) - \min_{j,k}\big(at(j,k)\big),$$

$$F_{RT} = 8 - 0 = 8$$

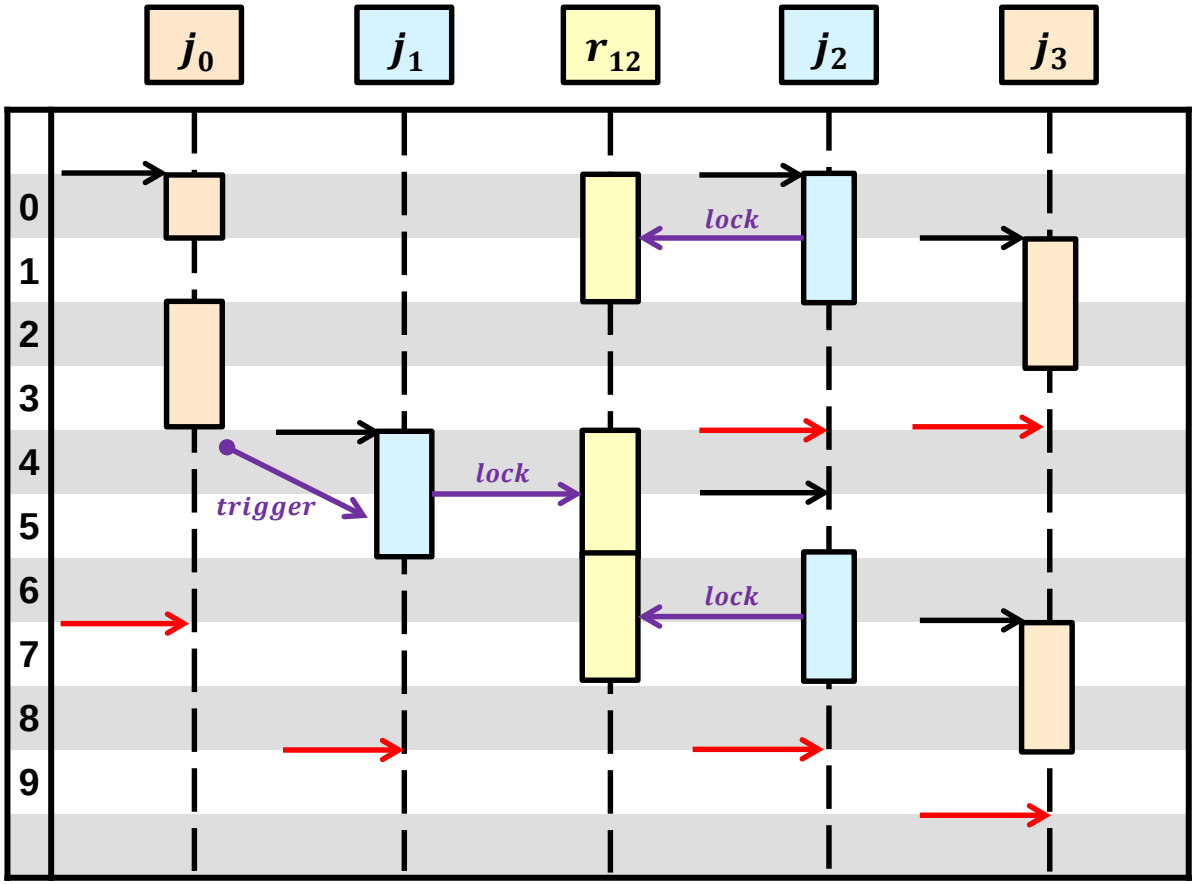
- **CPU Usage:**

$$F_{CU} = \frac{\sum_t (ld(t) > 0)}{tq}, F_{CU} = 0.9$$

F_{DM} should properly reward scenarios with deadline misses [1]

[1] L. Briand, Y. Labiche, and M. Shousha, "Using genetic algorithms for early schedulability analysis and stress testing in real-time systems", Genetic Programming and Evolvable Machines, vol. 7 no. 2, pp. 145-170, 2006

The FMS scheduler is modeled through constraints among Static and Dynamic properties (1/2)



Constraints

Well-formedness

- A task cannot start before it has arrived: $at(j,k) \leq st(j,k)$
- A task cannot finish before it has completed: $st(j,k) + dr(j) \leq en(j,k)$
- Arrival times of aperiodic tasks are separated by min/max interarr. times:
$$at(j,k-1) + mn(j) \leq at(j,k) \leq at(j,k-1) + mx(j)$$

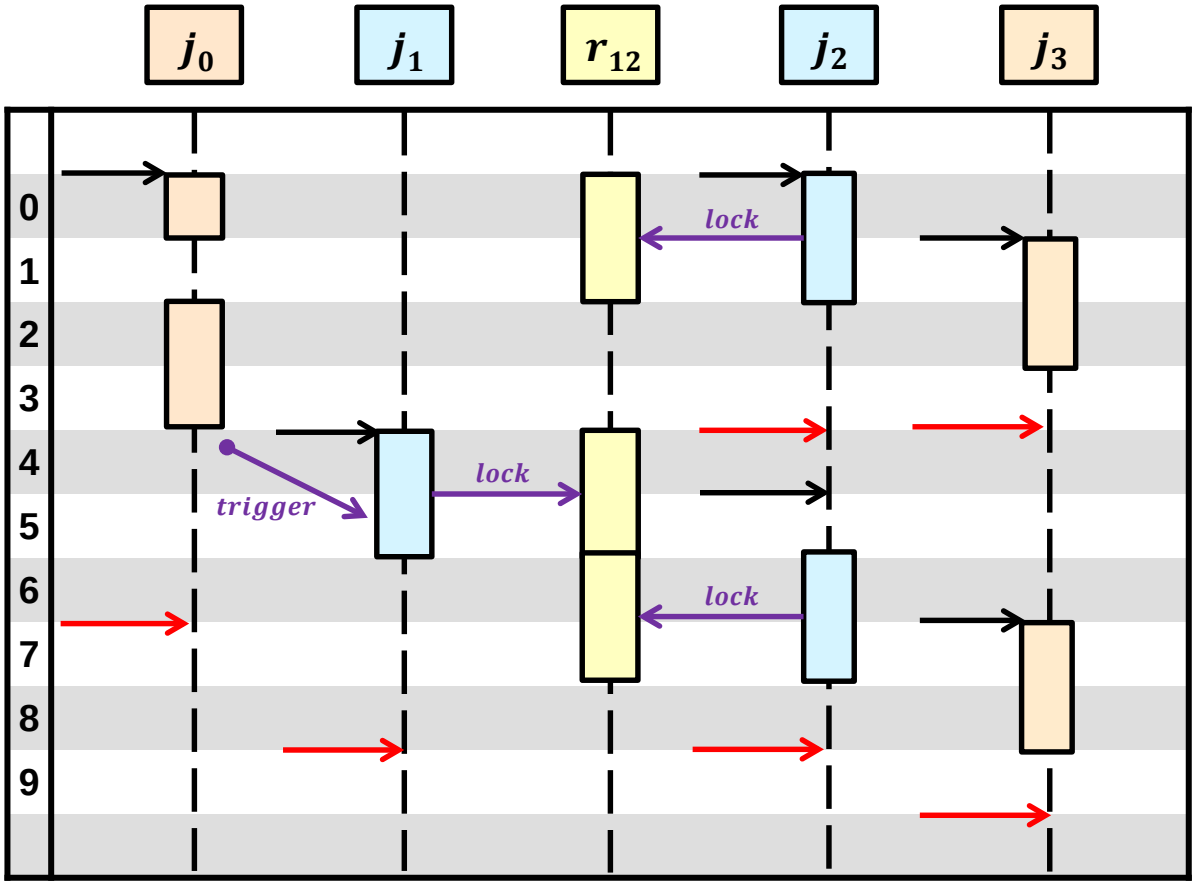
Temporal Ordering

- Triggered tasks arrive when their triggering task ends:
$$tg(j_1,j_2) \rightarrow en(j_1,k) = at(j_2,k)$$
- Dependent tasks cannot overlap:
$$de(j_1,j_2) \rightarrow en(j_1,k_1) < st(j_2,k_2) \vee en(j_2,k_2) < st(j_1,k_1)$$

Multicore

- The system load is always less than or equal to the number of cores:
$$ld(t) \leq c$$

The FMS scheduler is modeled through constraints among Static and Dynamic properties (2/2)



Constraints

Priority-Driven Preemption

- If a task is preempted, then there are c higher priority tasks running

Scheduling Efficiency

- If a task is waiting, then either
  - There are no free cores, or
  - A dependent task is active, or
  - A dependent task is preempted

ha, da and dp are defined as sums of boolean variables:

$$bl(j,k,j_1,k_1,p_1) = at(j,k) \leq ac(j_1,k_1,p_1) < st(j,k)$$

$$pm(j,k,p) \cdot c = \sum_{j_1 : pr(j_1) > pr(j),\, k_1, p_1} ac(j,k,p-1) < ac(j_1,k_1,p_1) < ac(j,k,p)$$

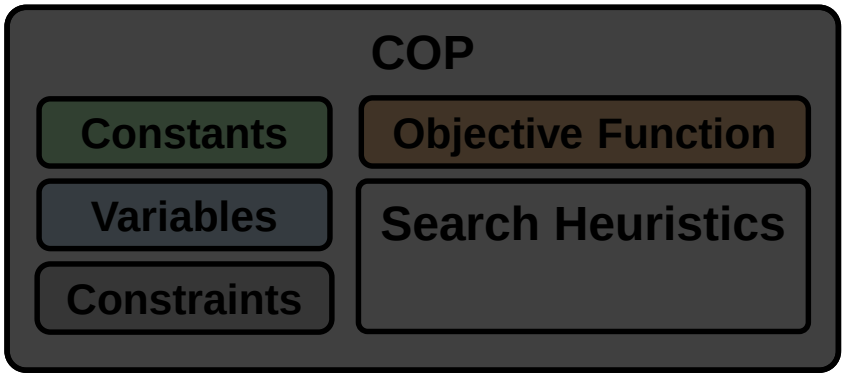
$$wt(j,k) = ha(j,k) + da(j,k) + dp(j,k)$$

- $ha(j,k)$ = Time quanta where c tasks with higher priority are active
- $da(j,k)$ = Time quanta where tasks depending on j are active
- $dp(j,k)$ = Time quanta where tasks depending on j are preempted

Our work originates from the interaction we had with Kongsberg Maritime over several months

1. Can the input data of our COP (constants) be provided with reasonable effort? [2]

~25 man-hours



2. Can one use the output data of our COP (variables) to derive test cases?

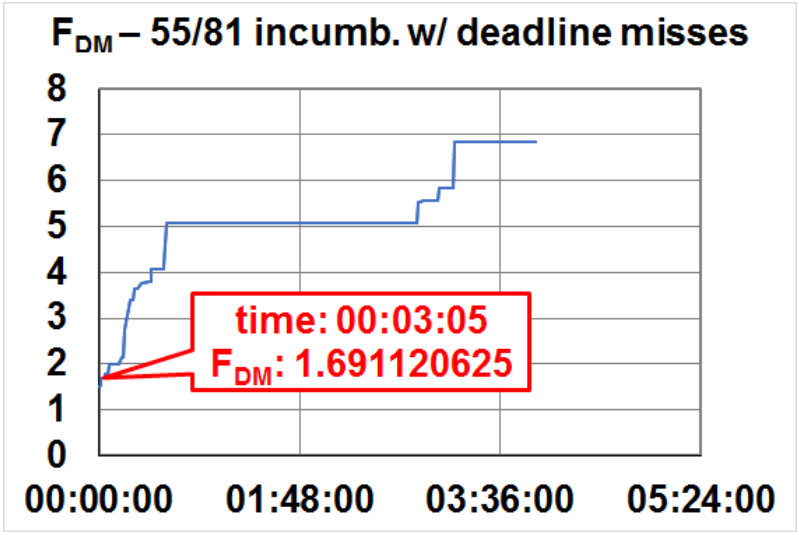
[2] Nejati, S., Di Alesio, S., Sabetzadeh, M., Briand, L.: Modeling and analysis of CPU usage in safety-critical embedded systems to support stress testing. In: Model Driven Engineering Languages and Systems, pp. 759–775. Springer (2012)

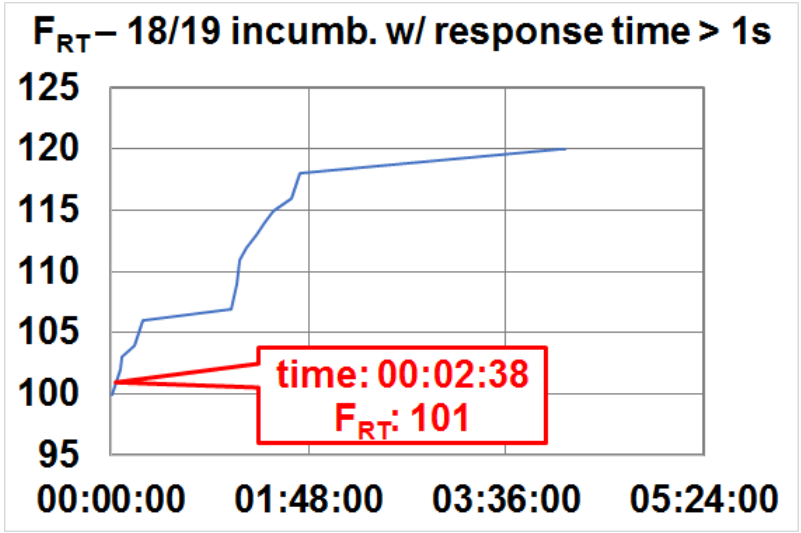
We run our COP model for 5 hours recording every incumbent, one run for each objective function

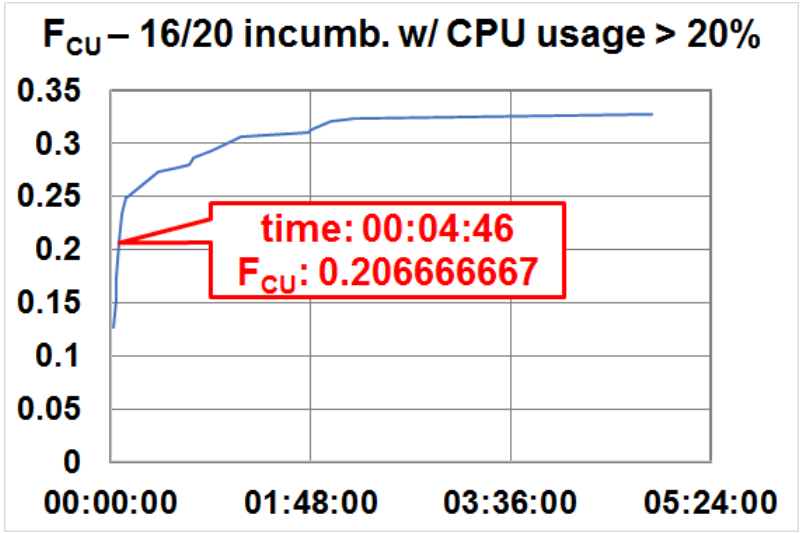
$T = 500, \quad 1\ tq = 10ms, \quad c = 3$

~600 variables, 1 million constraints in IBM ILOG CPLEX CP Solver

1. Worst deadline miss: *PushData*, by 10 ms in three executions
2. Highest response time: 1200 ms
3. Highest CPU Usage: 32%







In summary, we showed how Constrained Optimization can support Performance Testing

The COP models the System Scheduler, Tasks, and Perf. Reqs.

The COP finds arrival times leading to worst-case scenarios → test cases

We were able to generate test cases violating Perf. Reqs. in few minutes

Questions?