Optimal Performance Tuning in Real-Time Systems using Multi-Objective Constrained Optimization

Stefano Di Alesio

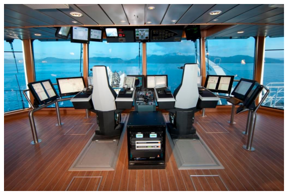
CP 2016

Toulouse, 06/09/2016

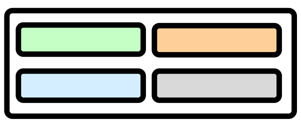
Certus
Centre for Software Verification & Validation
simula

Certus Centre for Software Verification and Validation
Simula Research Laboratory
Norway

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



Performance Requirements vs.
Real Time Embedded Systems (RTES)



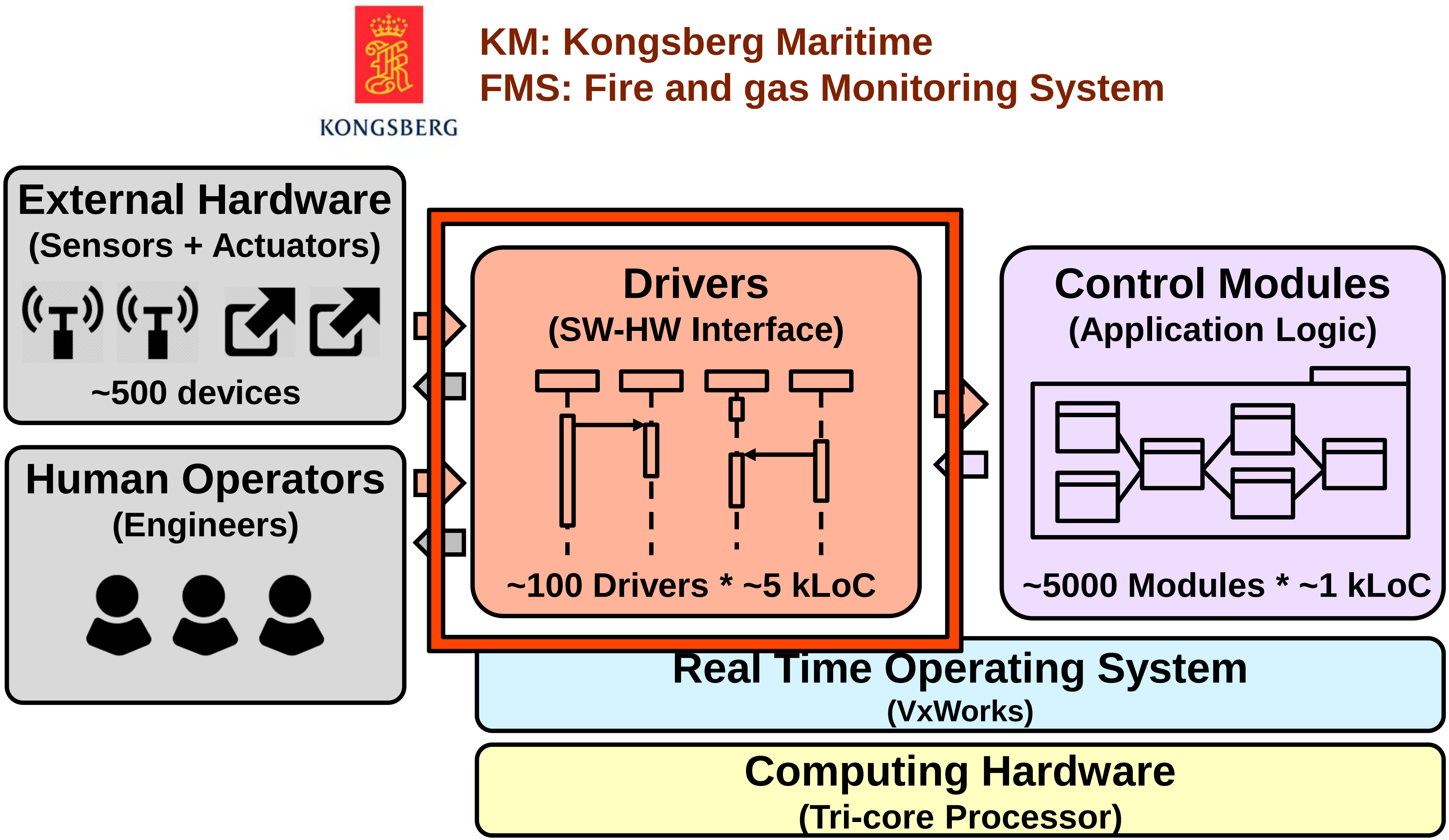
Supporting Performance Tuning:
A novel application for COPs

Industrial Experience:
Context, Process and Results

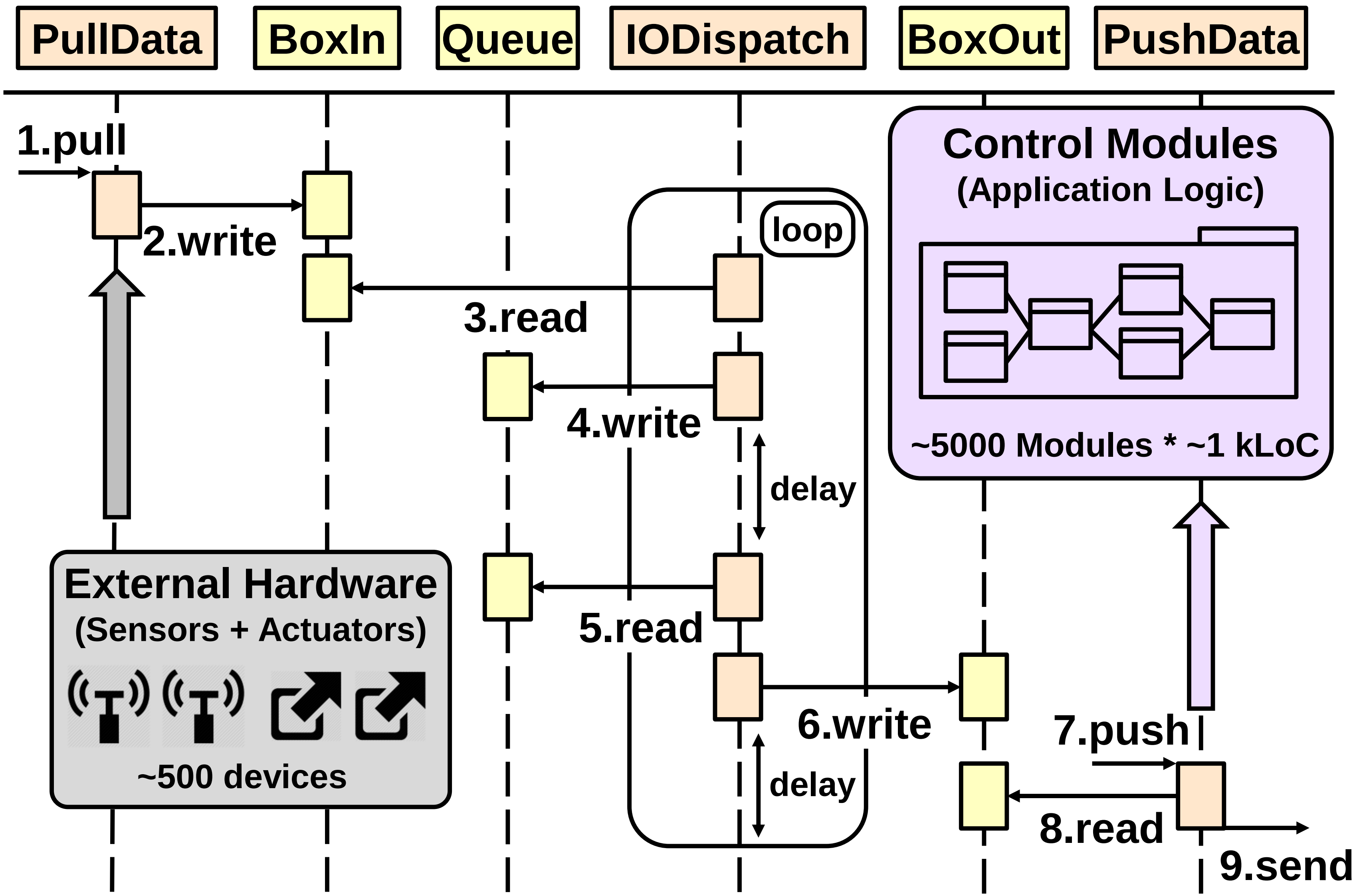
RTES are typically safety-critical, and thus bound to meet strict Performance Requirements

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

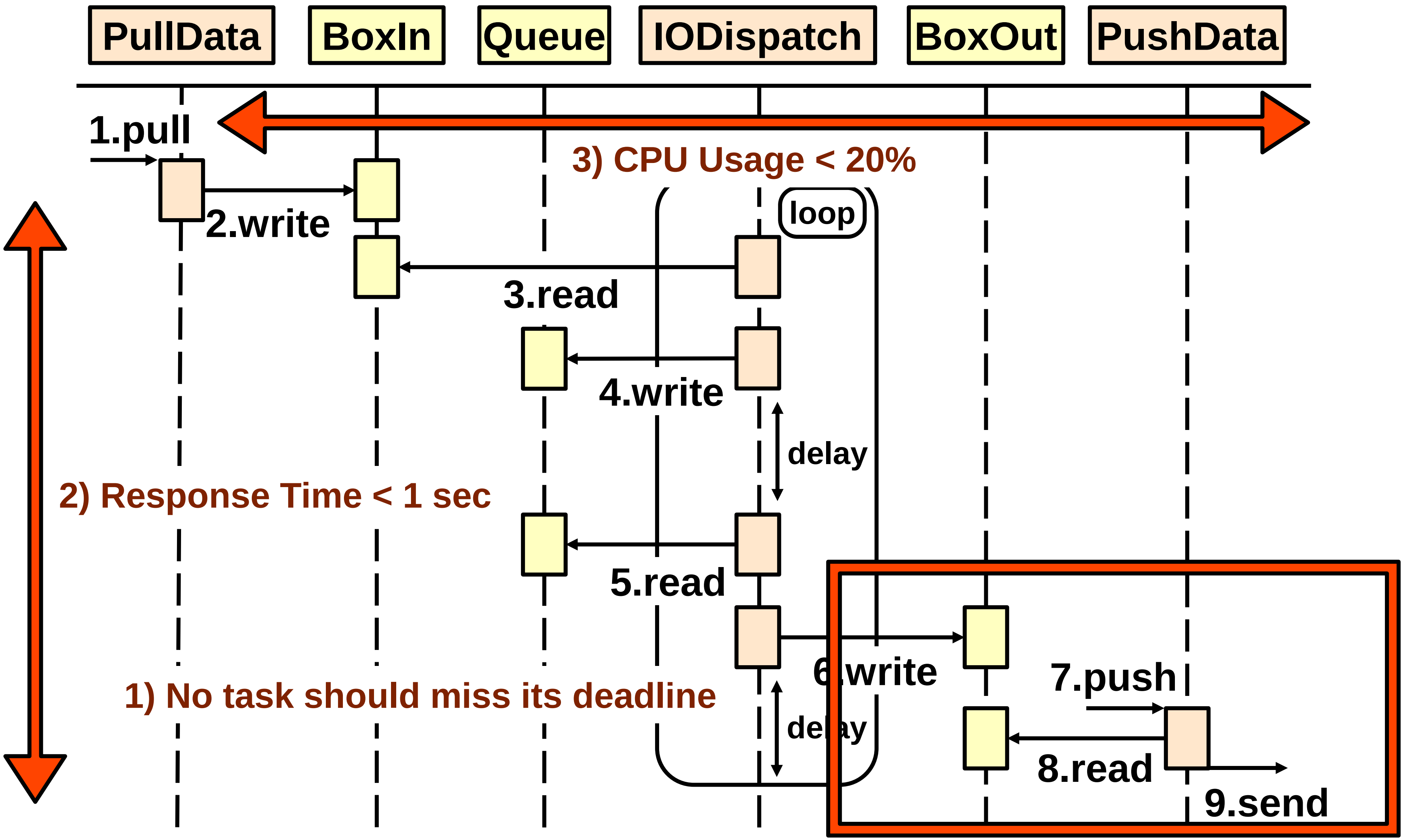
KM: Kongsberg Maritime
FMS: Fire and gas Monitoring System



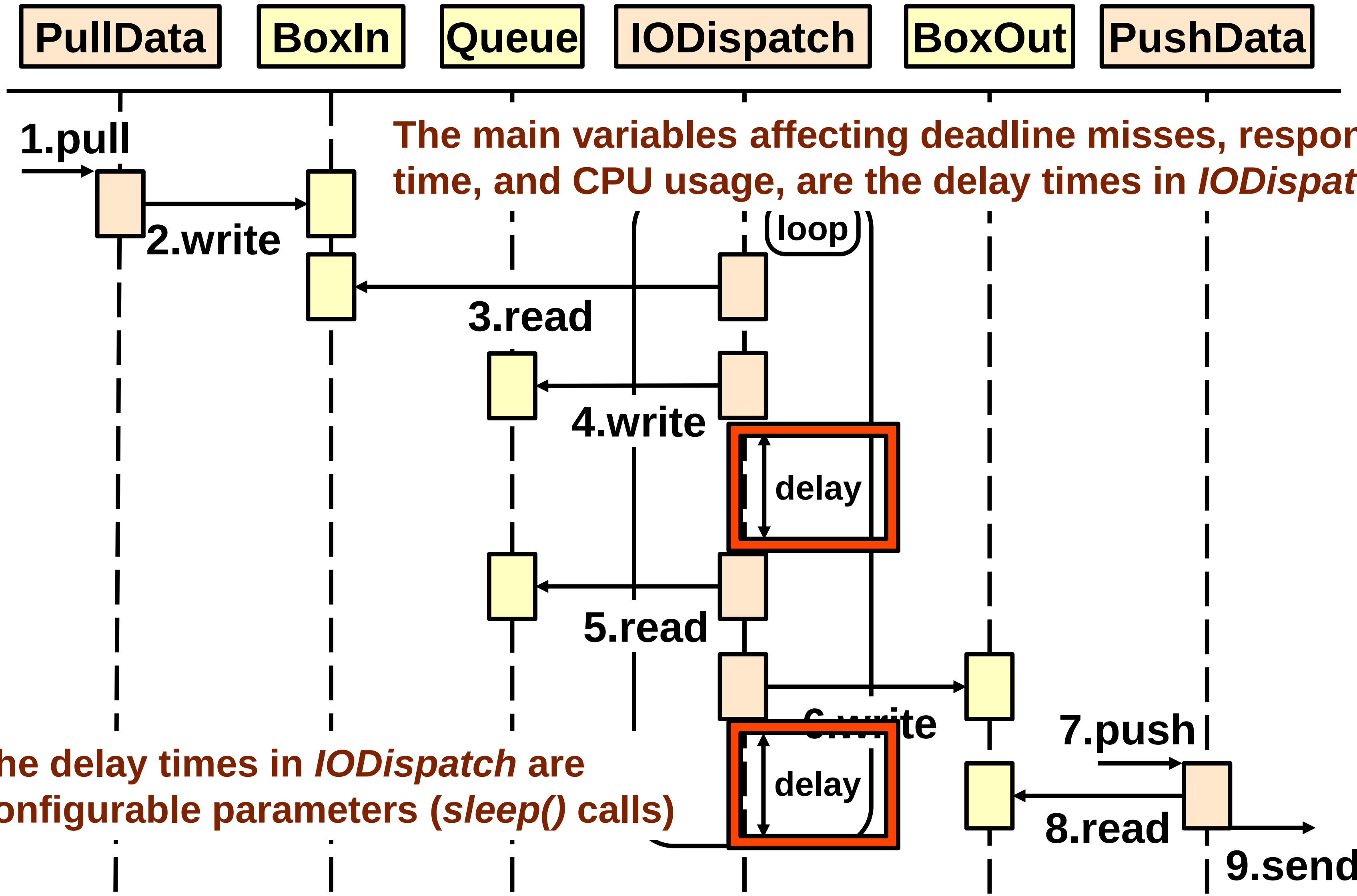
Drivers transfer data between external hardware (sensors and actuators) and control modules



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



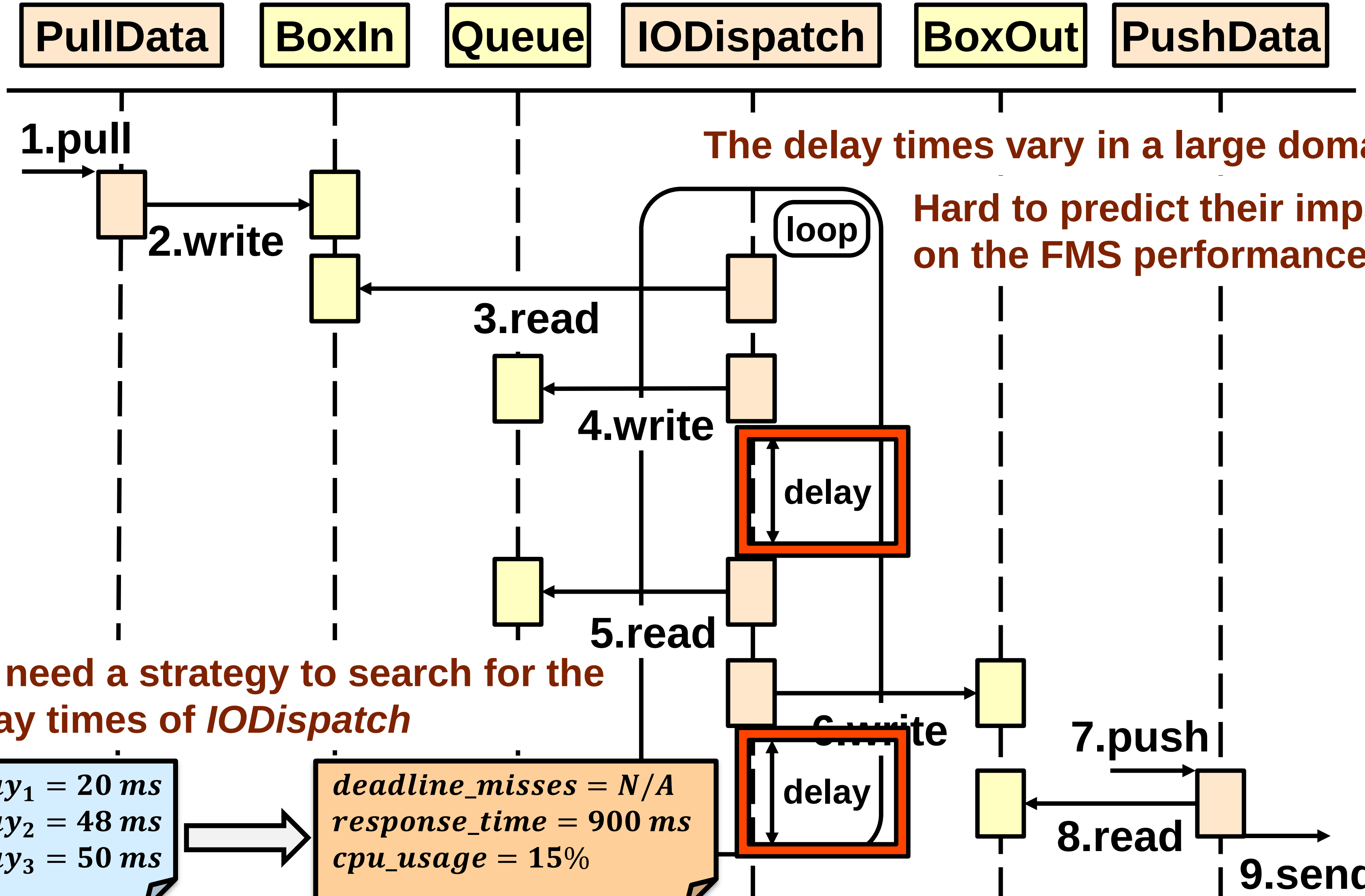
Our goal is to identify best-case scenarios w.r.t. deadline misses, response time, and CPU usage



The main variables affecting deadline misses, response time, and CPU usage, are the delay times in *IODispatch*

The delay times in *IODispatch* are configurable parameters (*sleep()* calls)

Each best-case scenario is characterized by the sequence of delay times of *IODispatch*



The delay times vary in a large domain

Hard to predict their impact on the FMS performance

We need a strategy to search for the delay times of *IODispatch*

$delay_1 = 20\ ms$
$delay_2 = 48\ ms$
$delay_3 = 50\ ms$

$deadline\_misses = N/A$
$response\_time = 900\ ms$
$cpu\_usage = 15\%$

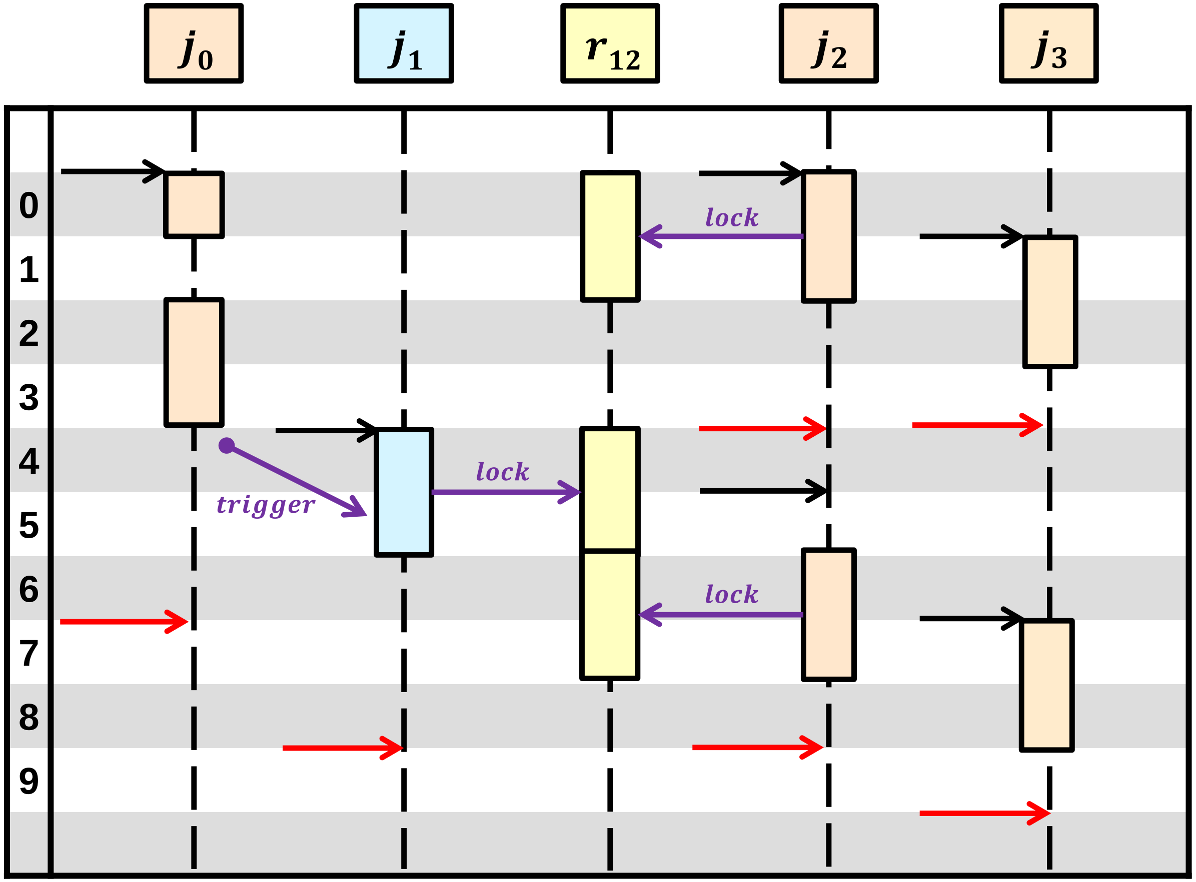
We cast the search for the delay times of *IODispatch* leading to best-case scenarios as a multi-obj. COP

The COP models a multi-core priority-driven preemptive scheduler with task (delayed) triggering and r/w dependencies

Multi-objective Constrained Optimization Problem (COP)

Static Properties of Tasks (Constants)	**Performance Requirements** (Objective Functions)
Dynamic Properties of Tasks (Variables)	**OS Scheduler Behaviour** (Constraints)

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

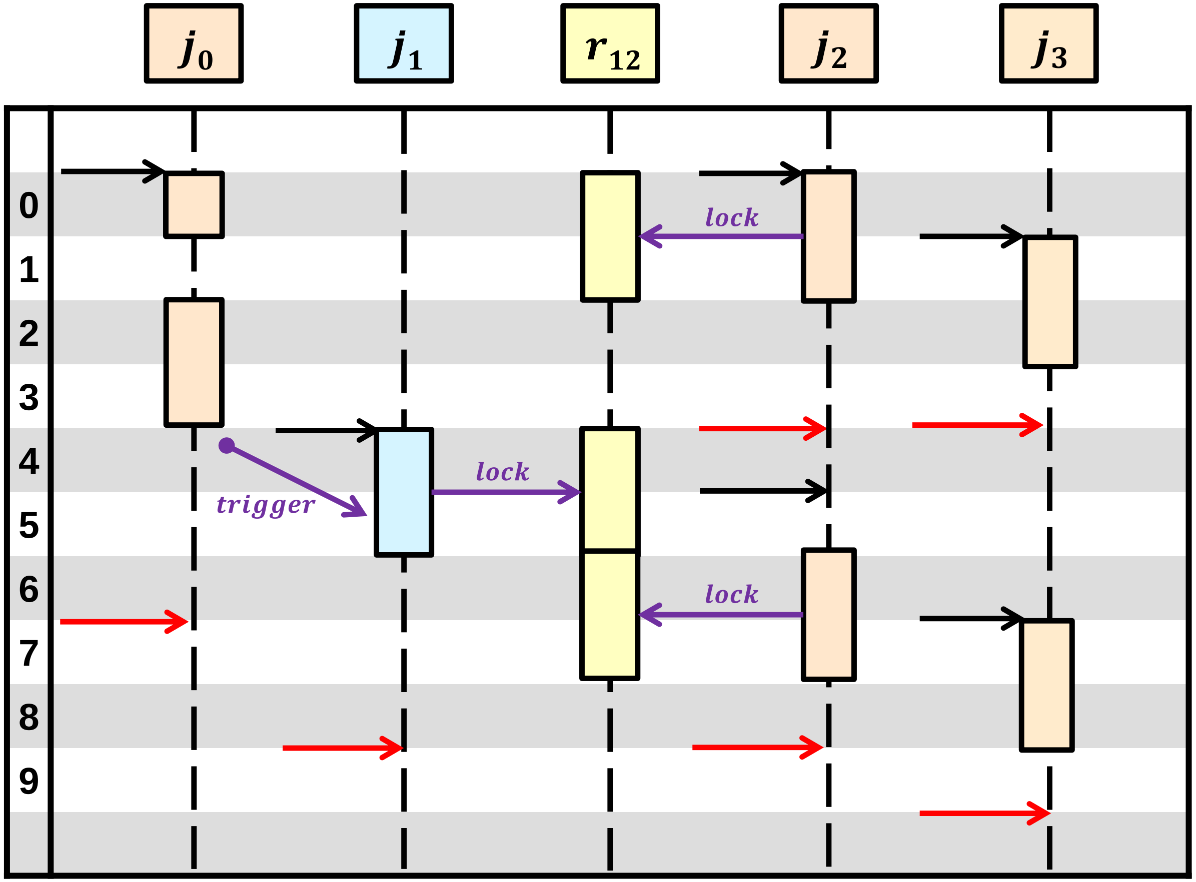


Constants

- **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:** $pe(j_3) = 6$
- **Offset of Periodic Tasks:** $of(j_3) = 1$
- **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:** $de(j_1, j_2)$
- **Read Relation:** $rd(j_1, r_{12})$
- **Write Relation:** $wr(j_2, r_{12})$

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

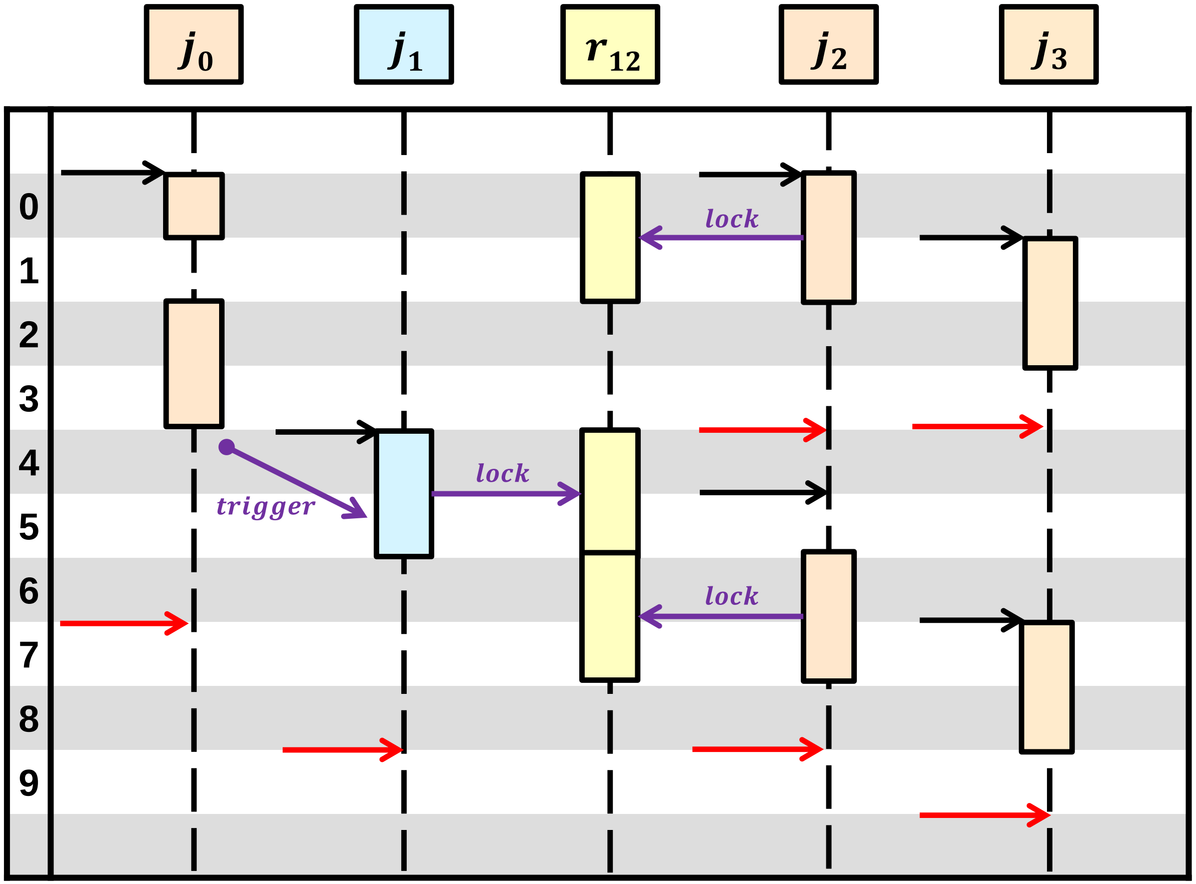
Independent (Decision)

- **Arrival time of Triggered Task Exec.:** $at(j,k)\ \epsilon\ T,\ at(j_0,0)=0,\ ac(j_3,1)=7$
- **Delay time of Triggered Task Exec.:** $dy(j,k)\ \epsilon\ T,\ dy(j_1,0)=0$
- **Active time of Task Executions:** $ac(j,k,p)\ \epsilon\ T,\ p\ \epsilon\ [0,dr(j)-1],$ $ac(j_0,0,0)=0,\ ac(j_0,0,1)=2$

The at **of a periodic tasks execution is a constant:**

$at(j,k)=of(j)+k*pe(j),\qquad at(j_3\ 1)=1+1*6=7$

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



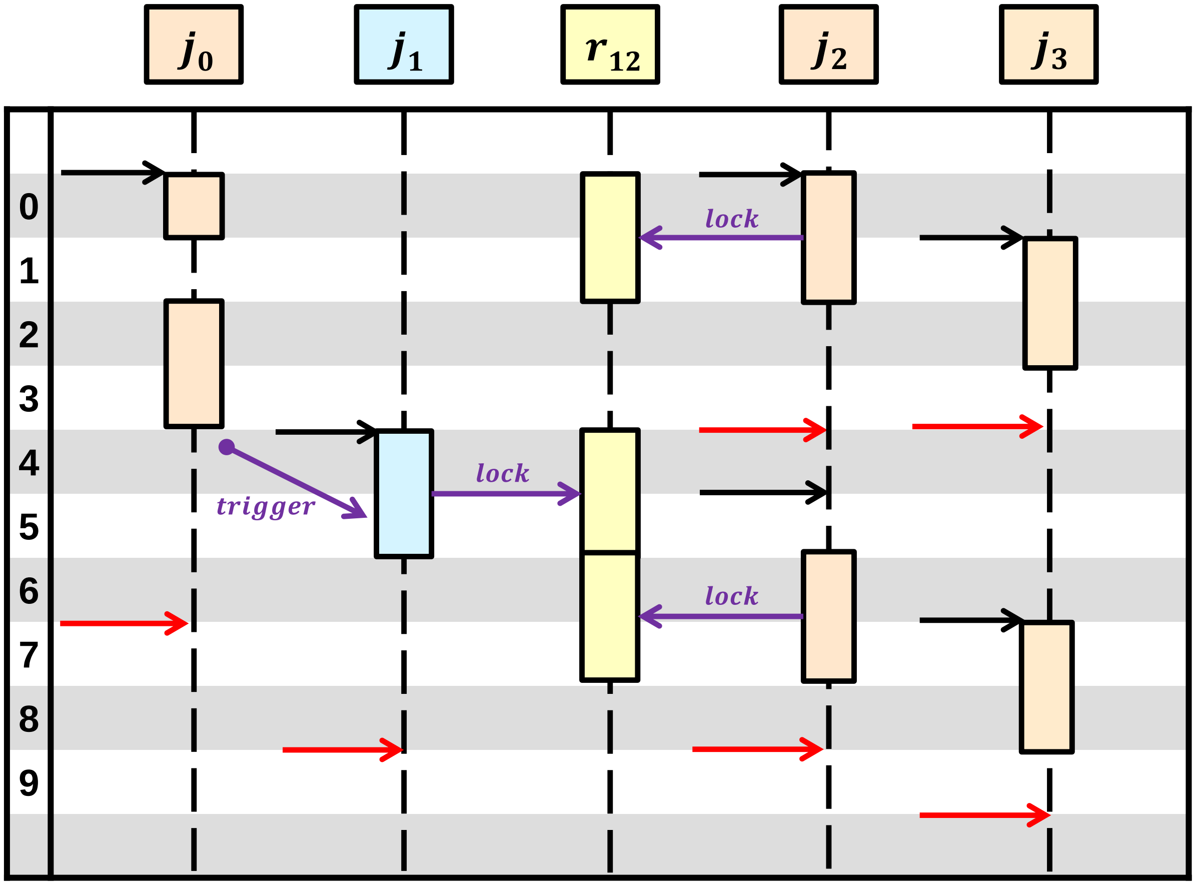
Variables

Dependent

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

- **System Load:** $ld(t) = \sum_{j,k,d}(ac(j,k,d) = t)$, $ld(0) = 2$, $ld(3) = 1$
- **Resource Status:** $rs(r,t) = \begin{cases} 1 & \text{if } t \in [en(j_2,k), en(j_1,k)] \\ 0 & \text{otherwise} \end{cases}$ $rs(r_{12},1) = 0$, $rs(r_{12},2) = 1$

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



Objective Functions

- **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:**

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

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

- **CPU Usage:**

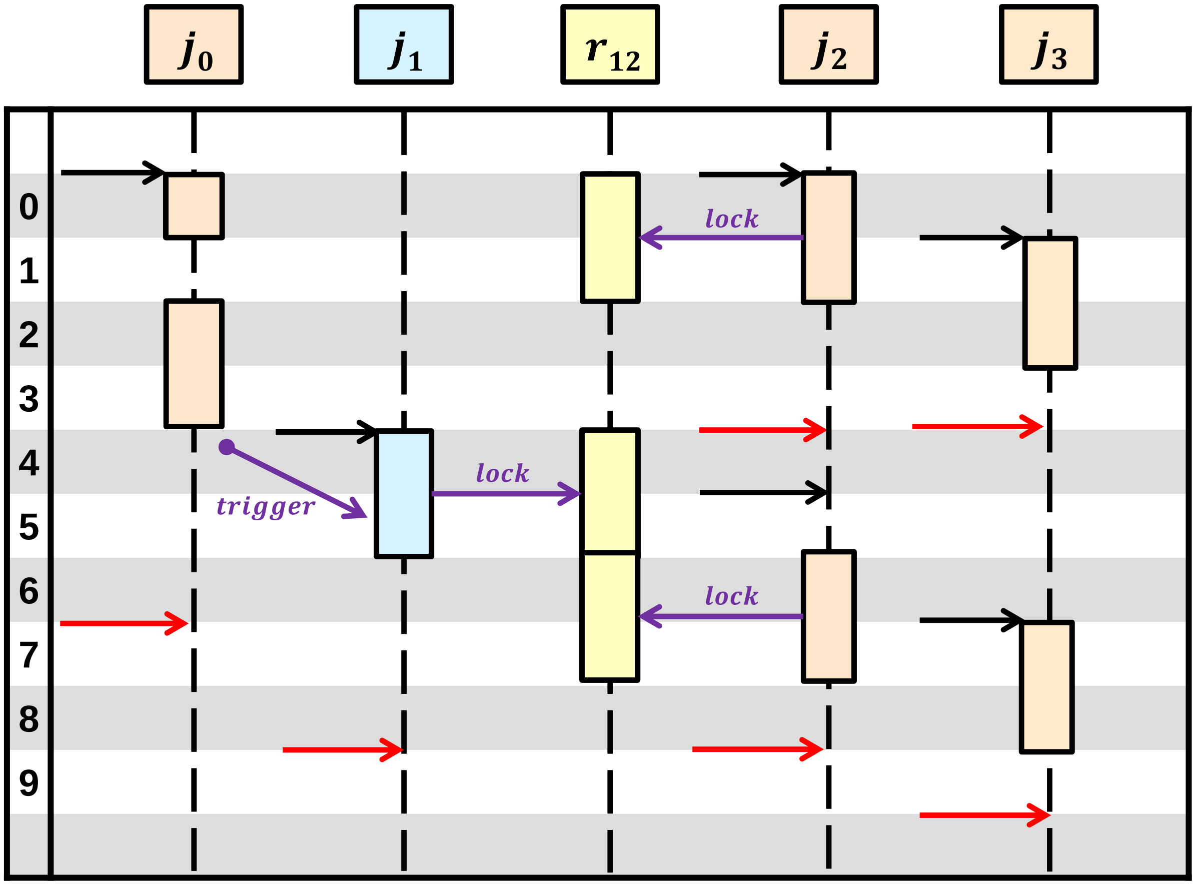
$$F_{CU} = \frac{\sum_t (ld(t) > 0)}{tq}, F_{CU} = 0.9$$

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

We used lexicographic multi-objective optimization:
3 criteria, 3! = 6 optimization runs

[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:

$$de(j_1,j_2) \rightarrow en(j_1,k_1) < st(j_2,k_2) \vee en(j_2,k_2) < st(j_1,k_1)$$

- Tasks must read from (write to) full (empty) buffers:

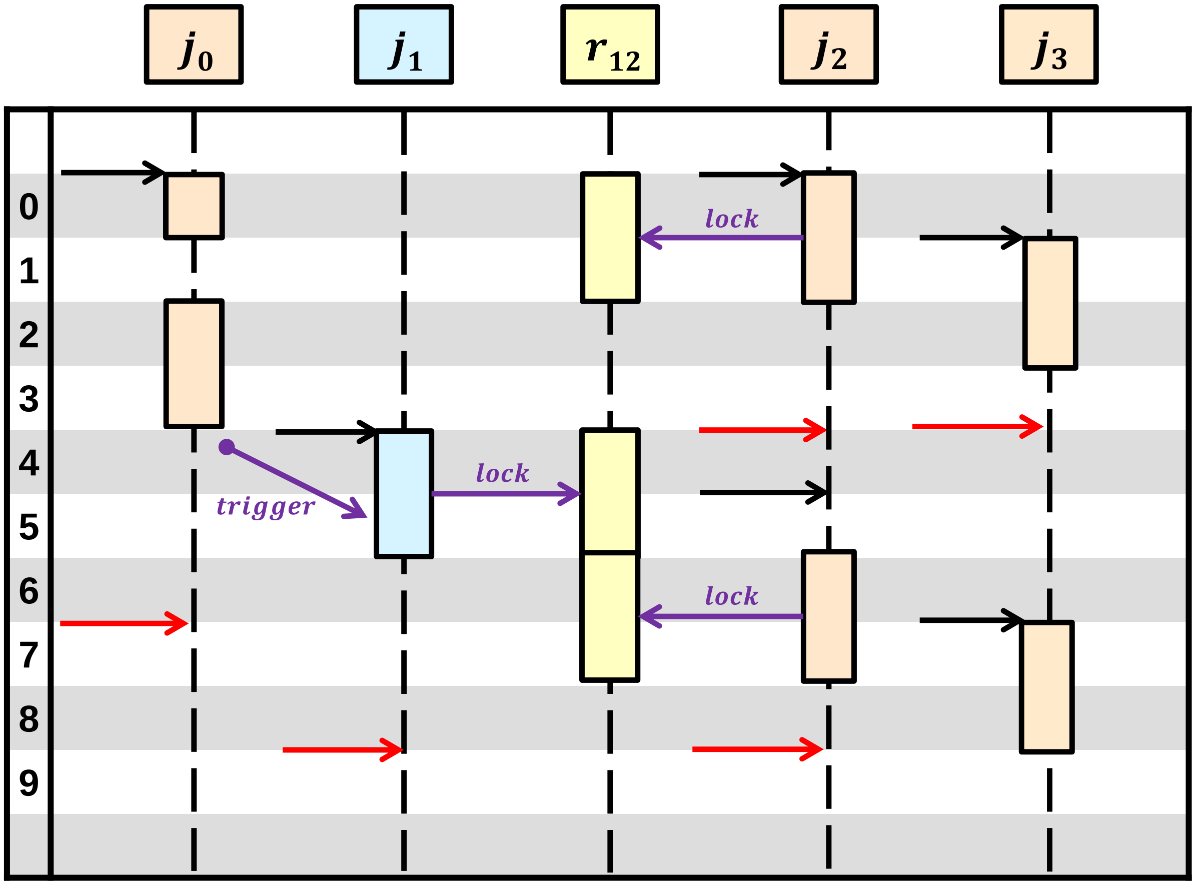
$$rd(j,r) \rightarrow rs\big(r, st(j,k)\big) = 1$$
$$wr(j,r) \rightarrow rs\big(r, st(j,k)\big) = 0$$

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 [2]

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

Scheduling Efficiency [2]

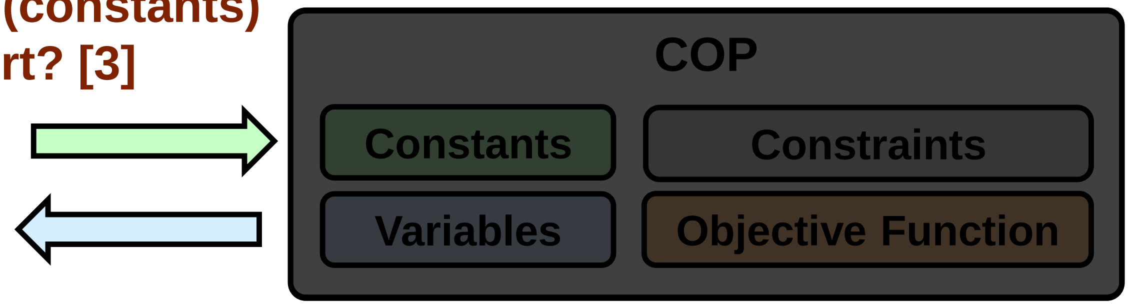
- If a task is waiting, then either
  - There are no free cores, or
  - A dependent task is active, or
  - A dependent task is preempted

[2] S. Di Alesio, S. Nejati, L. Briand, A. Gotlieb. "Worst-case Scheduling of Software Tasks – A Constraint Optimization Model to Support Performance Testing" In International Conference on Principles and Practice of Constraint Programming (CP 2014)

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? [3]

~25 man-hours



2. Can one *conveniently* use the output data of our COP (variables) to derive configurations?

Effectiveness: revealing power of best-case scenarios

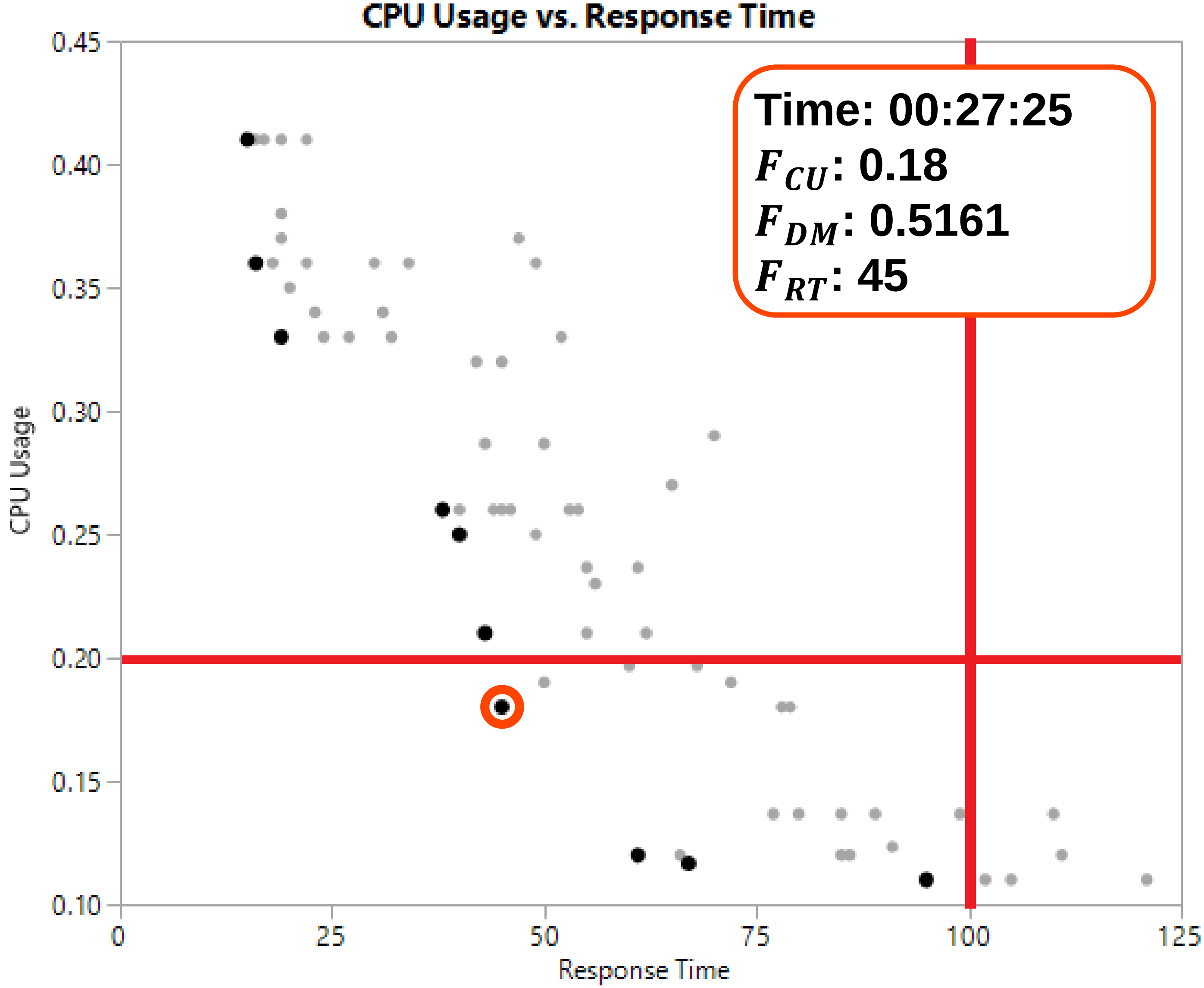
Efficiency: time needed to generate configurations

[3] S. Nejati, S. Di Alesio, M. Sabetzadeh, L. Briand: Modeling and Analysis of CPU Usage in Safety-critical Embedded Systems to Support Stress Testing. In: Model Driven Engineering Languages and Systems (MODELS 2012)

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

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

We found 4/10 solutions in the frontier (20/71 total) with F_{CU} < 0.2, no deadline misses, and F_{RT} < 100



In summary, we showed how CO can support Performance Tuning in complex industrial RTES

The COP models RT Scheduler, Tasks, and Performance Requirements

The COP finds delay times leading to best-case scenarios → configurations

We found Pareto-optimal delay times in < 30 min

Questions?