itk ENGINEERING

Central Vehicle Computer: Achieving freedom from interference by temporal software separation

Christopher Schwager

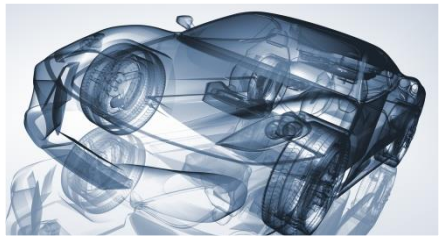
– public –

ITK Engineering GmbH

Realizing customers' vision

Test Methods

Cognitive Solutions



Development Methods

Internet of Things

Verification & Validation

Control System Design

Software Development

Functional Safety

Systems Engineering

System & Software Integration

Cyber Security

Image & Signal Processing

Agenda

- Central Vehicle Computers – Key Requirements
- Adaptive AUTOSAR – A Solution Provider for Temporal Software Separation ?
- Mastering Parallel Execution of Code
- Conclusion

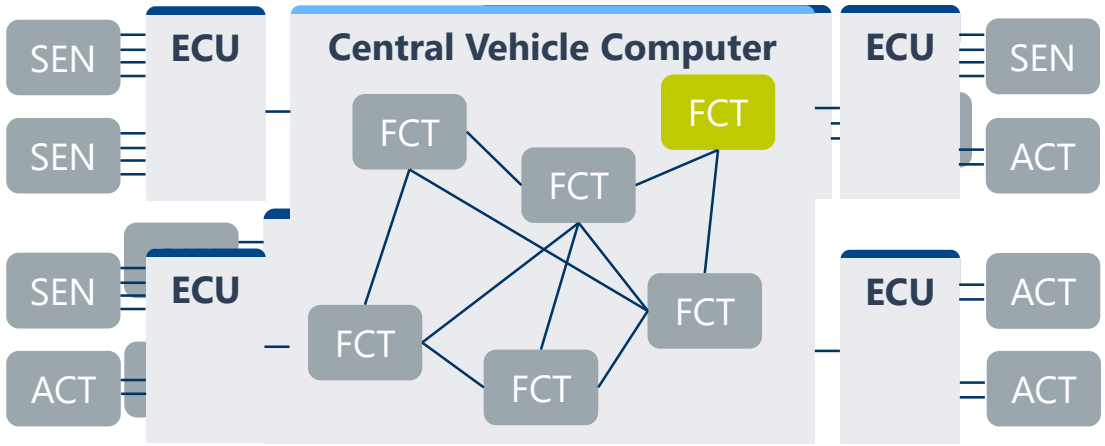
Central Vehicle Computers

Key Requirements

Central Vehicle Computers

Logical centralization instead of ECU patchwork

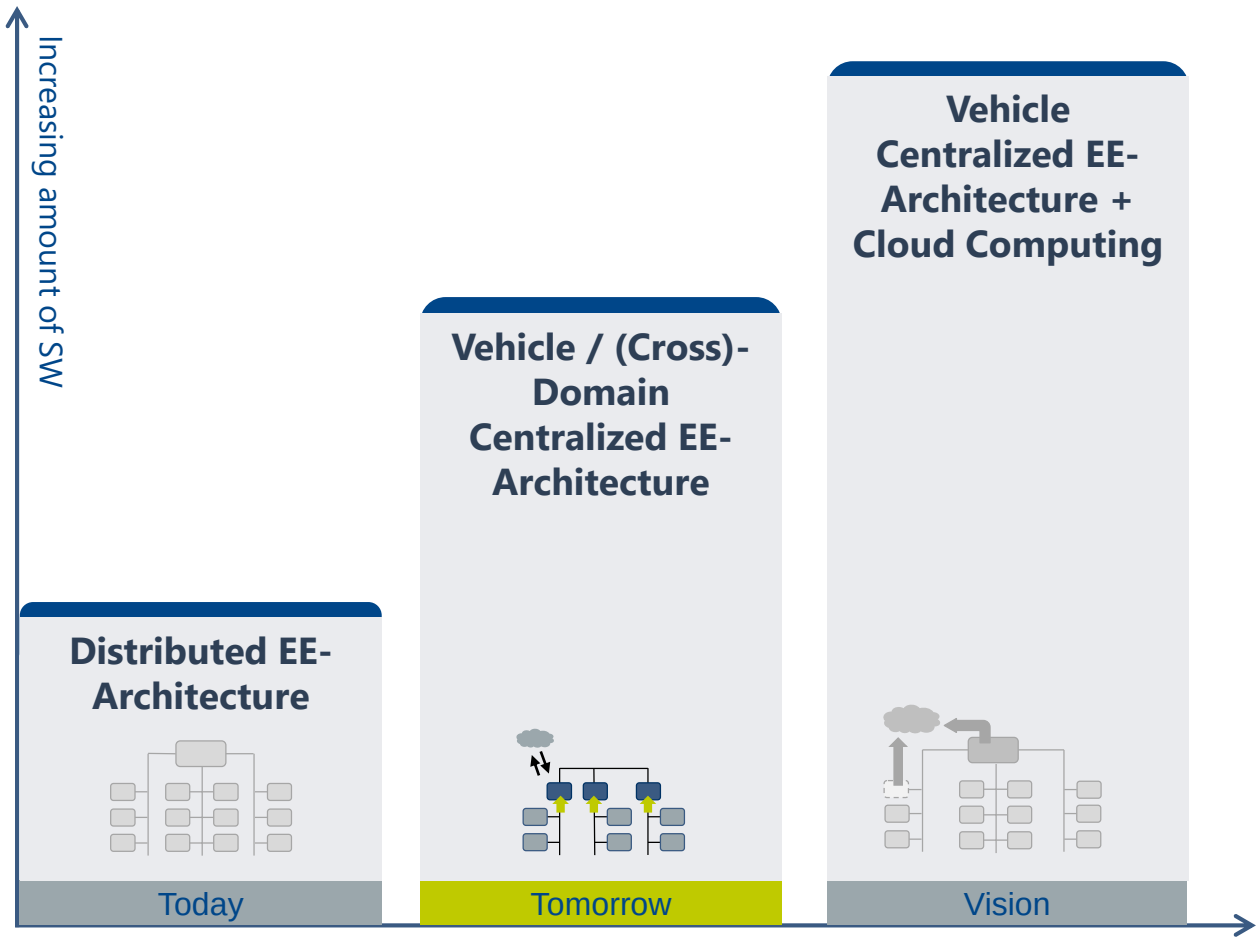
- Today's patchwork approach: adding new ECUs for new features is no longer sustainable
  - Distribution of highly interconnected functions on multiple ECUs is more complex than central integration



Logical centralization & EE architecture drives SW architecture.

Central Vehicle Computers

Centralized EE architecture



Central Vehicle Computers

Extensive software separation / isolation is key requirement

- Central vehicle computers as SW integration platform host many heterogenous applications in terms of
  - Real-time
  - Safety / security properties
  - Provided by cross-domain SW suppliers
- Fault detection via e.g. a watchdog is not sufficient as availability of functions is an increasing demand of future fault tolerant HAF systems

Key Requirement

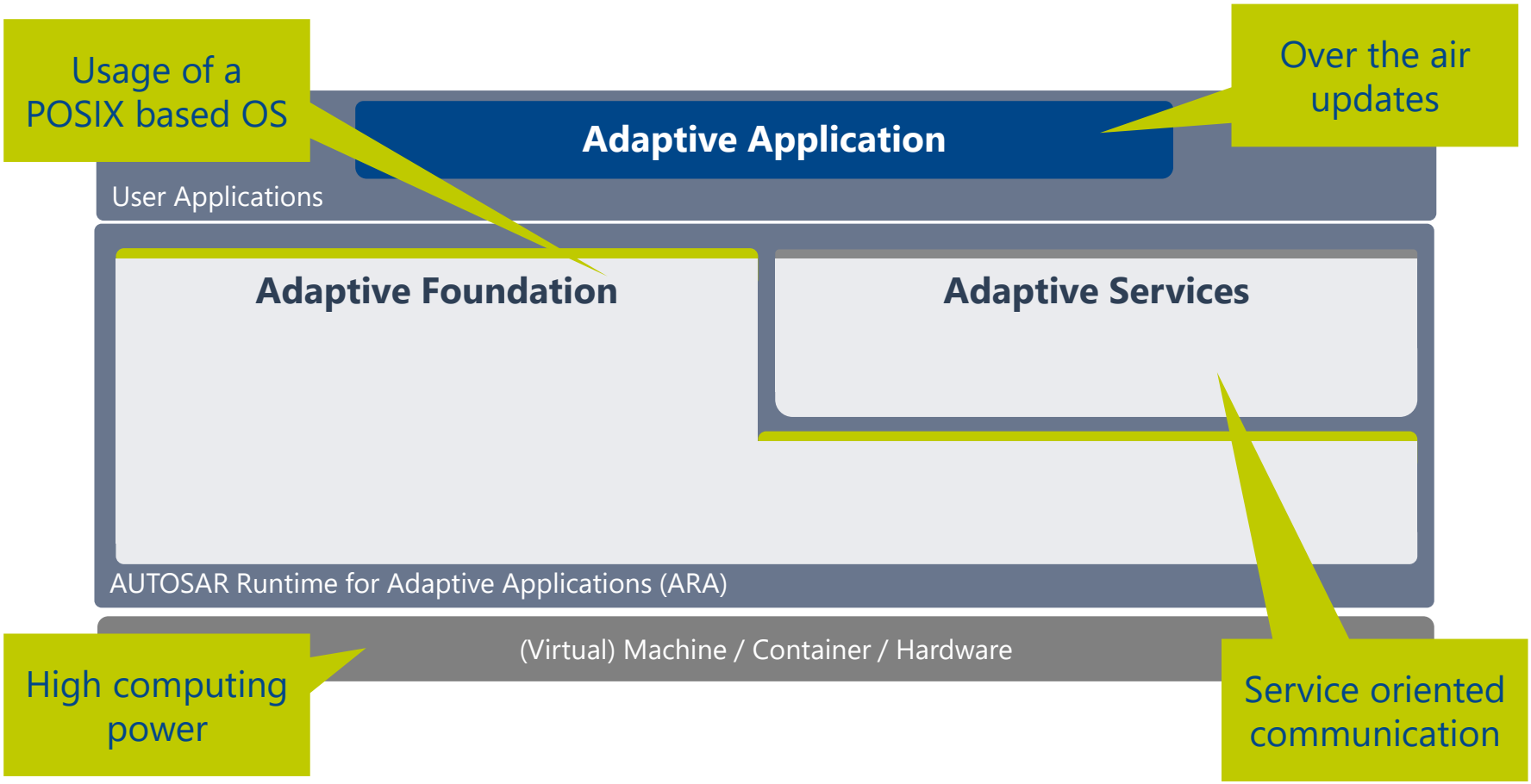
Extensive software separation and isolation capabilities to achieve freedom from interference

Adaptive AUTOSAR

A Solution Provider for Temporal Software Separation?

AUTOSAR Adaptive Platform

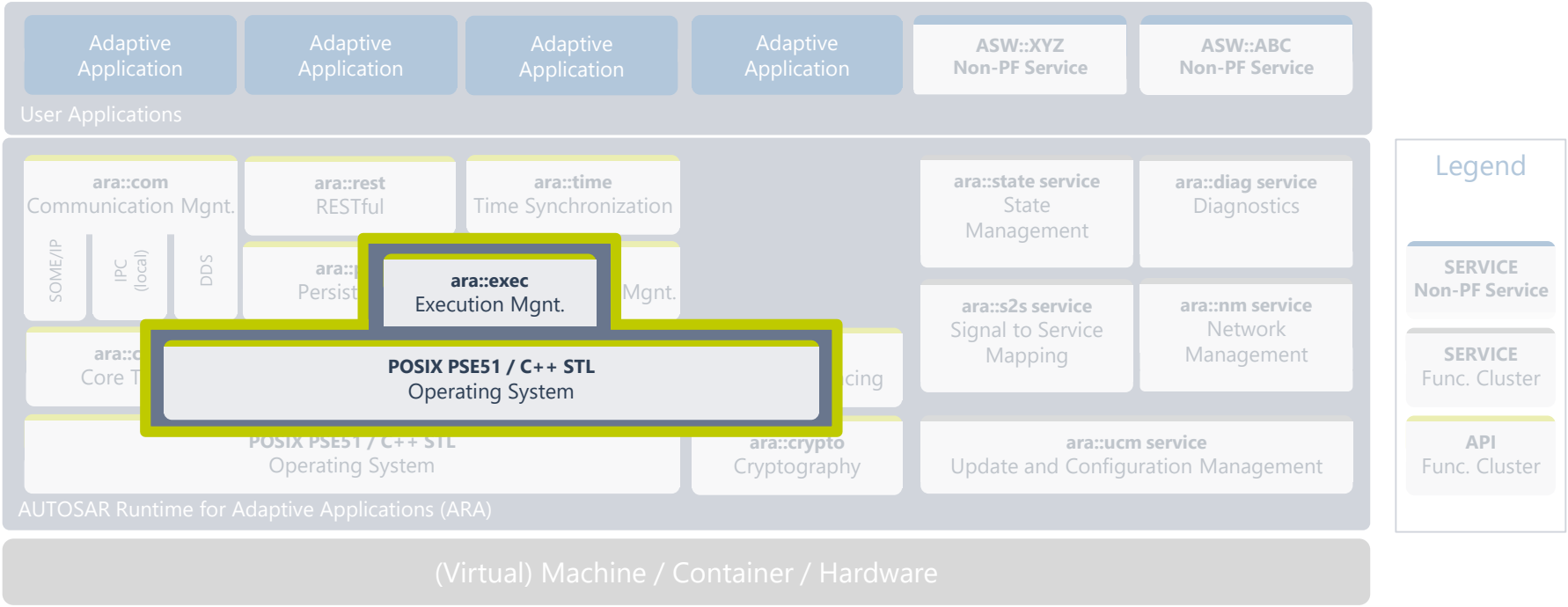
Adaptive AUTOSAR tackles new market requirements



AUTOSAR Adaptive Platform

A solution provider for temporal software separation?

- 2 functional clusters are related to scheduling & execution of applications
  - Operating system (conform to POSIX PSE51)
  - Execution management



AUTOSAR Adaptive Platform

A solution provider for temporal software separation?

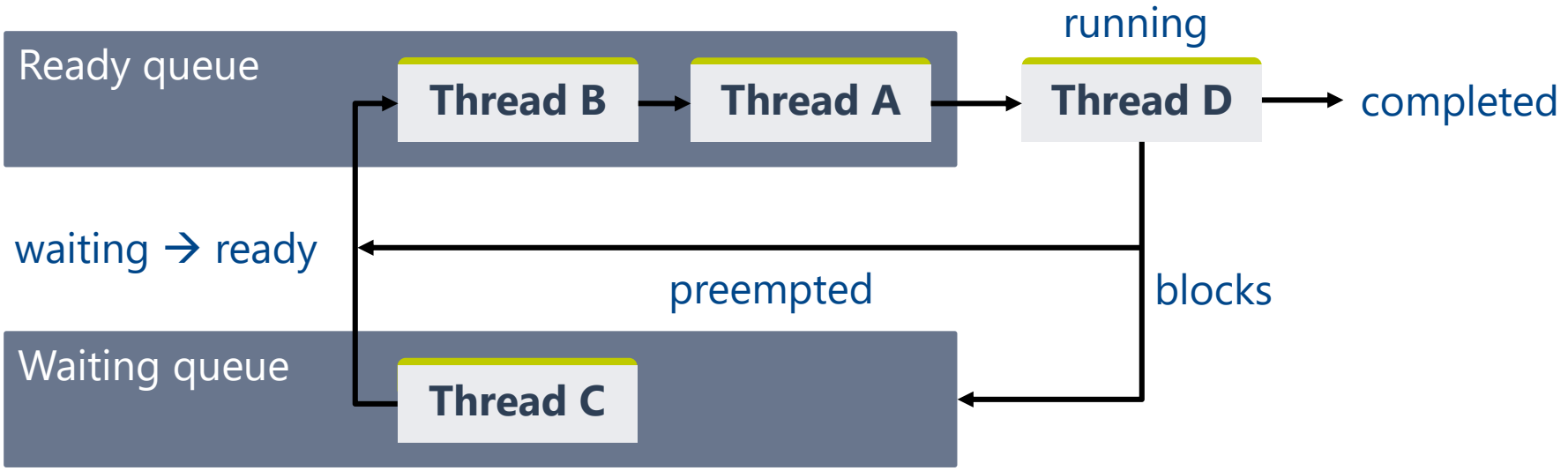
Operating System Interface Specification

«The Adaptive Platform Operating System shall support the following scheduling policies defined in the IEEE1003.1 POSIX standard»

- First in First Out (FiFo)
- Round Robing (RR)
- Other → This is no real-time scheduling policy

Remark: *«Since the above mentioned default scheduling policies may not guarantee proper execution for all real-time scenarios,…»*

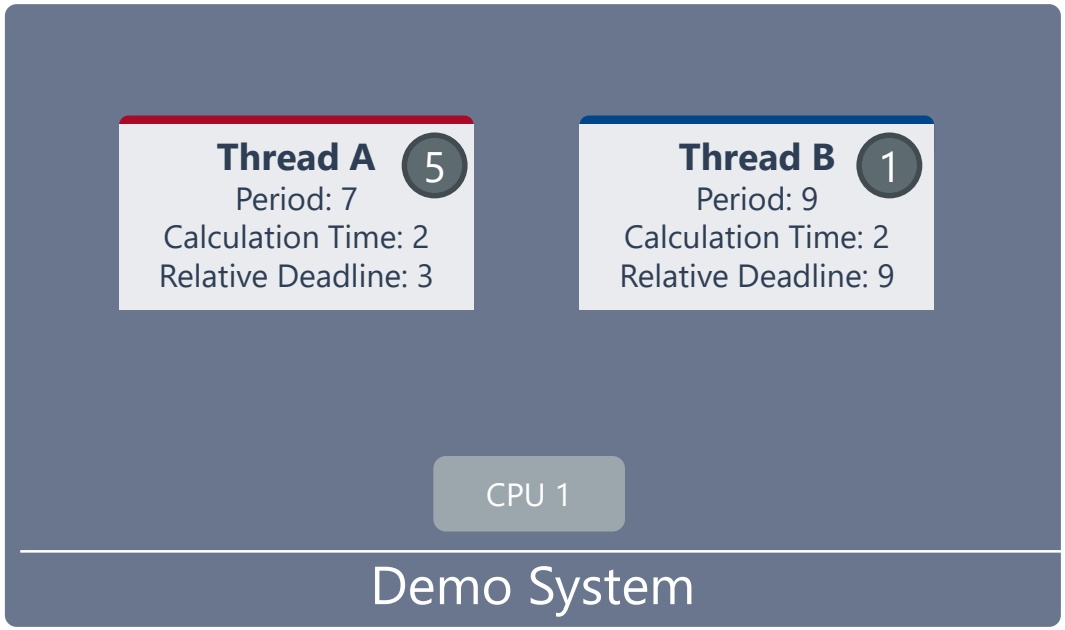
Wrap-Up: FiFo / Round Robin scheduling

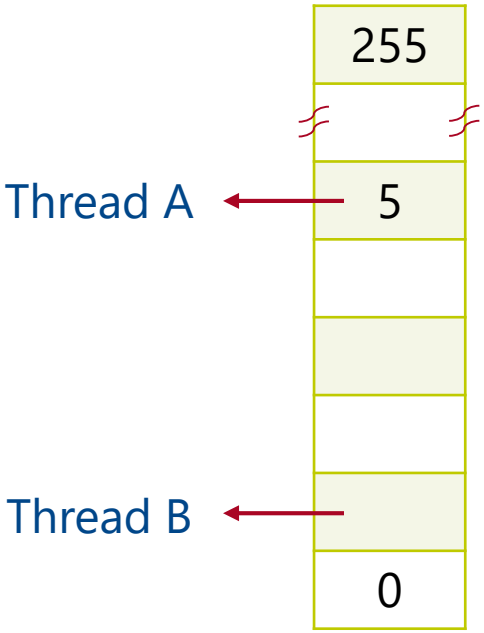


Demo Scenario

Initial Setup

- Demo System with 2 threads
  - Thread A has a high criticality (e.g. ASIL D)
  - Thread B has a low criticality (e.g. ASIL A)
- Rate-Monotonic priority assignment





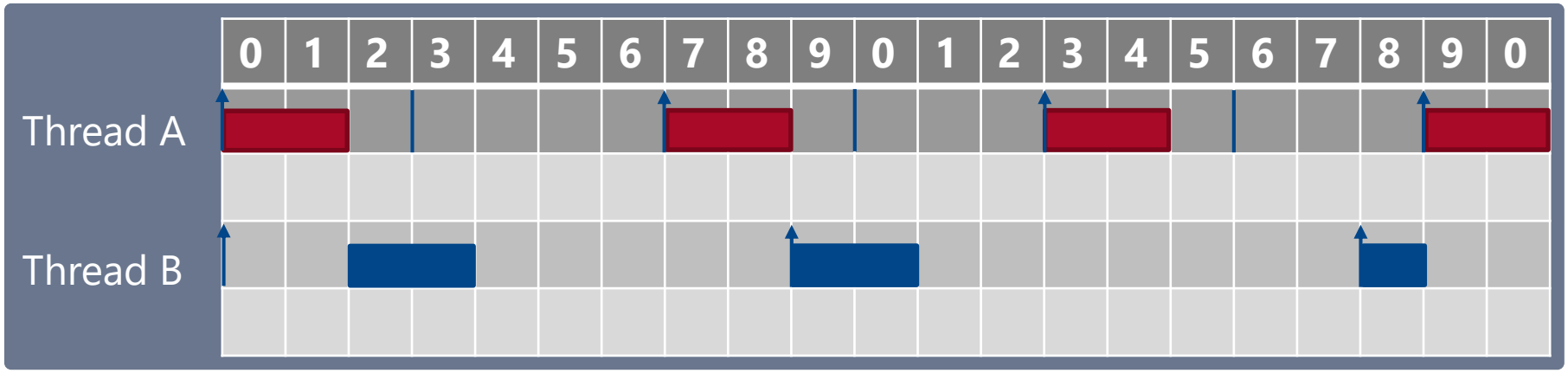
Demo Scenario

Initial scheduling using FiFo / Round Robin

- Thread A
  - Always preempts Thread B
  - Always meets its deadline
- Thread B
  - Could be preempted by Thread A

Thread A 5
Period: 7
Calculation Time: 2
Relative Deadline: 3

Thread B 1
Period: 9
Calculation Time: 2
Relative Deadline: 9



Demo Scenario

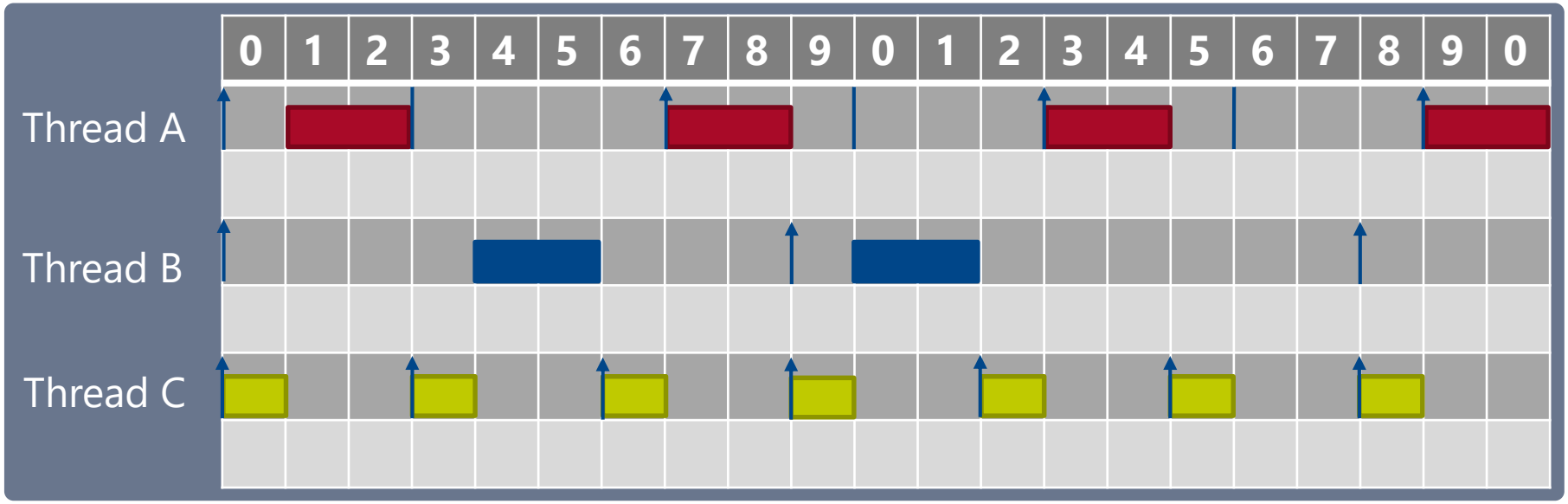
Adding a third thread

- Thread C
  - QM criticality – e.g. HMI Data Provider
  - Shortest period
  - Highest priority

Thread A 5
Period: 7
Calculation Time: 2
Relative Deadline: 3

Thread B 1
Period: 9
Calculation Time: 2
Relative Deadline: 9

Thread C 8
Period: 3
Calculation Time: 1
Relative Deadline: 3



Demo Scenario

Incorrect execution of thread C leads to criticality inversion

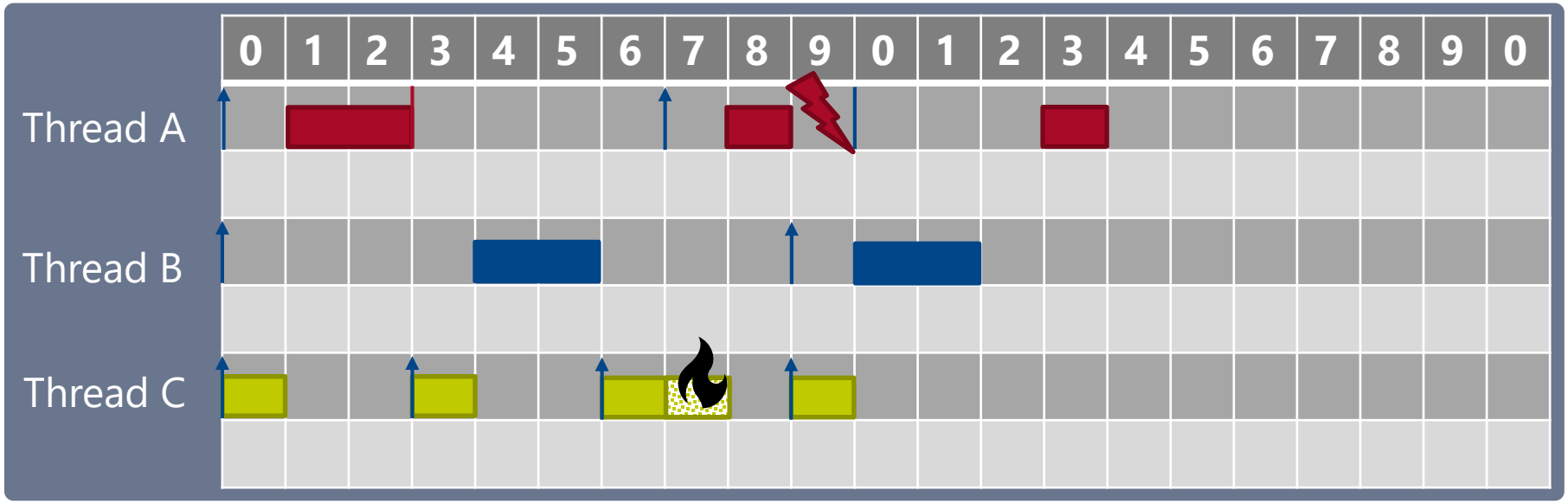
- Thread C
  - Thread C is at fault and still meets its deadline
  - Thread A is not at fault and misses its deadline

→ criticality inversion → no temporal isolation!

Thread A 5
Period: 7
Calculation Time: 2
Relative Deadline: 3

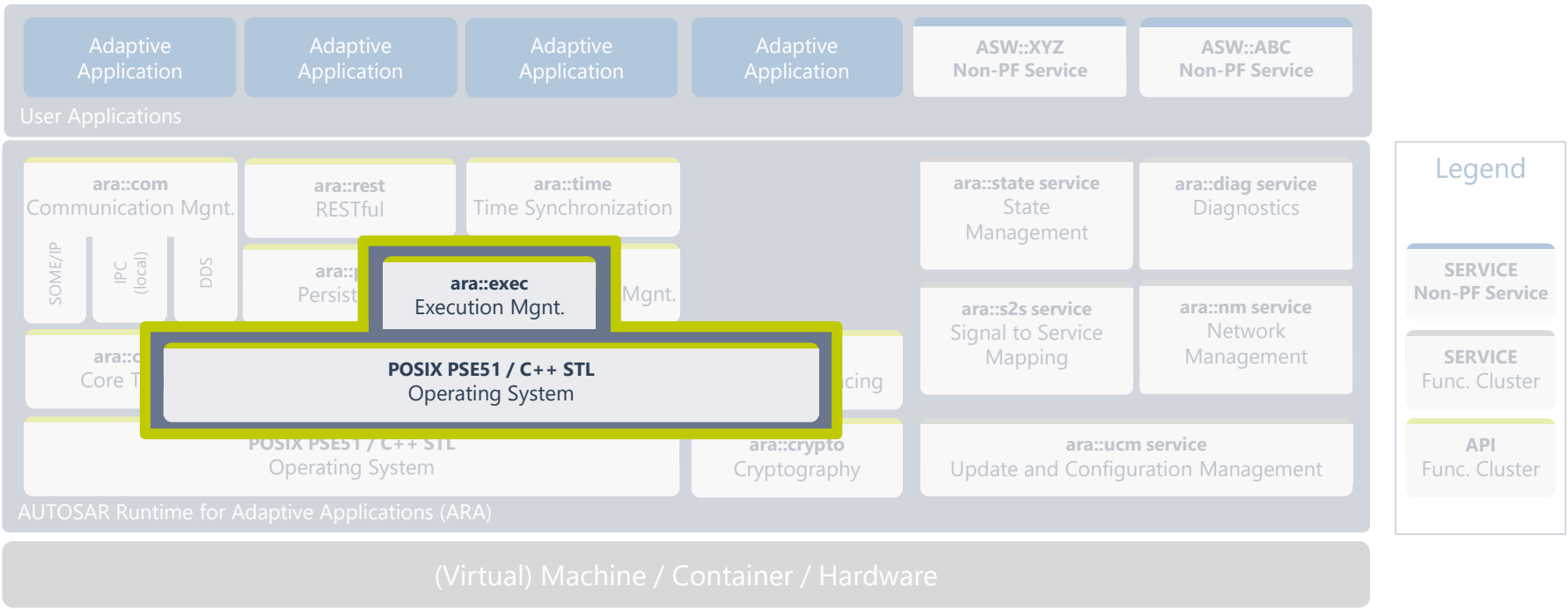
Thread B 1
Period: 9
Calculation Time: 2
Relative Deadline: 9

Thread C 8
Period: 3
Calculation Time: 1
Relative Deadline: 3



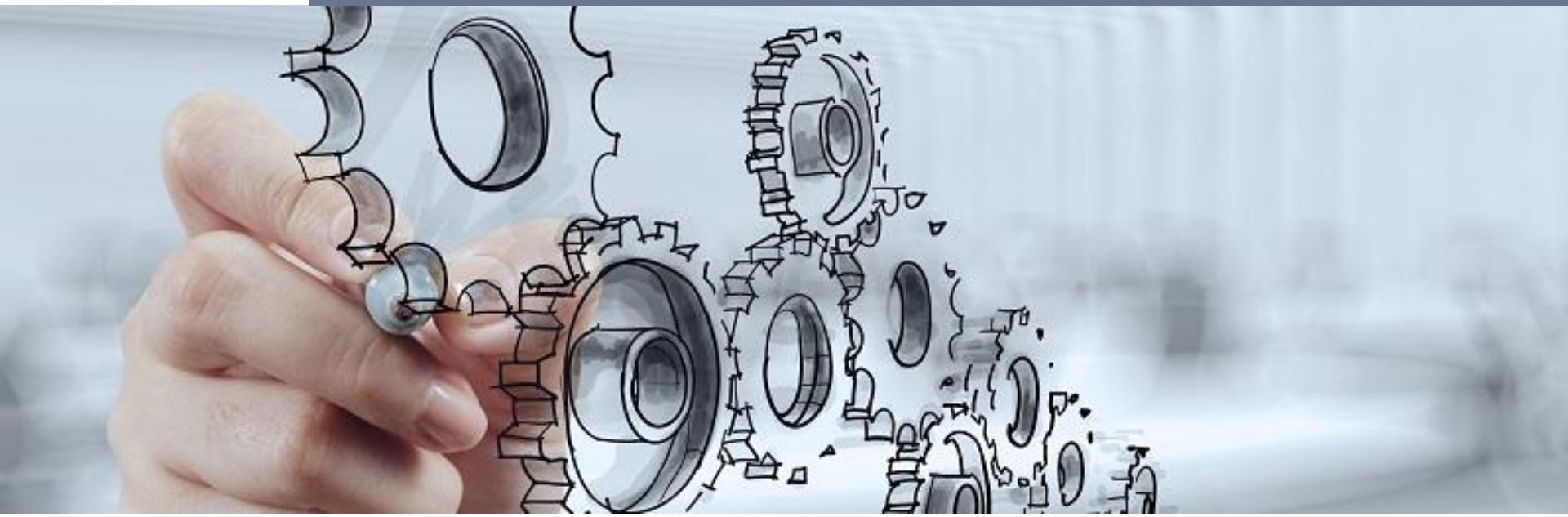
AUTOSAR Adaptive Platform

Adaptive AUTOSAR != Standard Solution for Freedom from Interference



POSIX scheduling policies may not guarantee proper execution for all real-time scenarios.

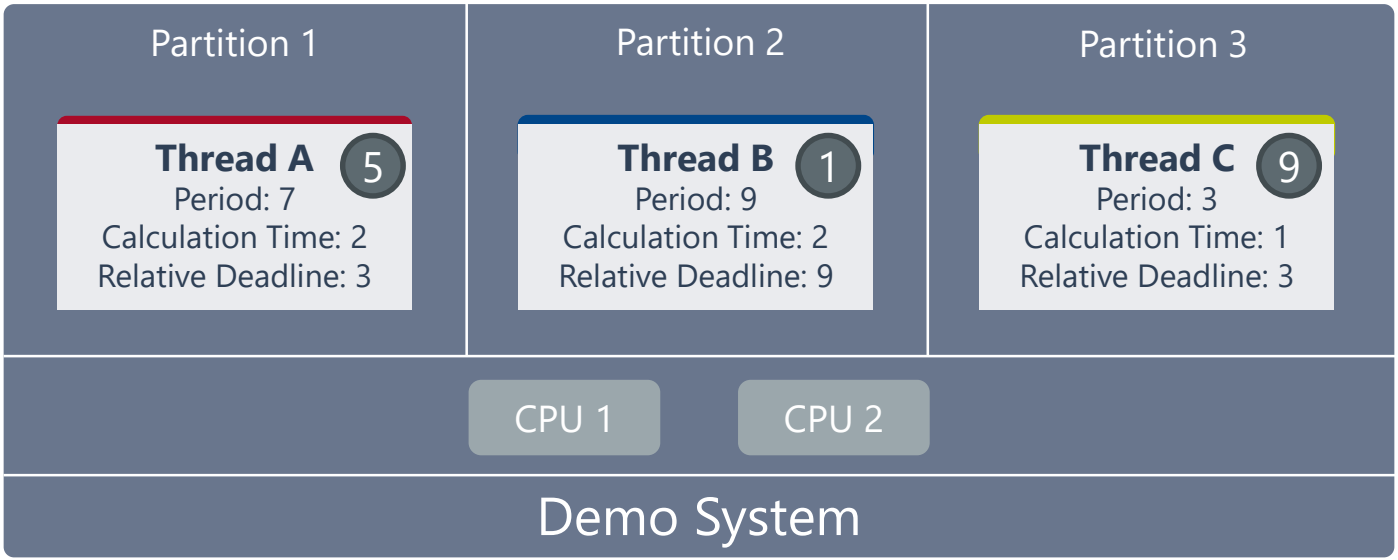
Mastering Parallel Execution of Code



Partition the system by criticality

Approach

- Partition the system based on the criticality and introduce a second stage scheduler

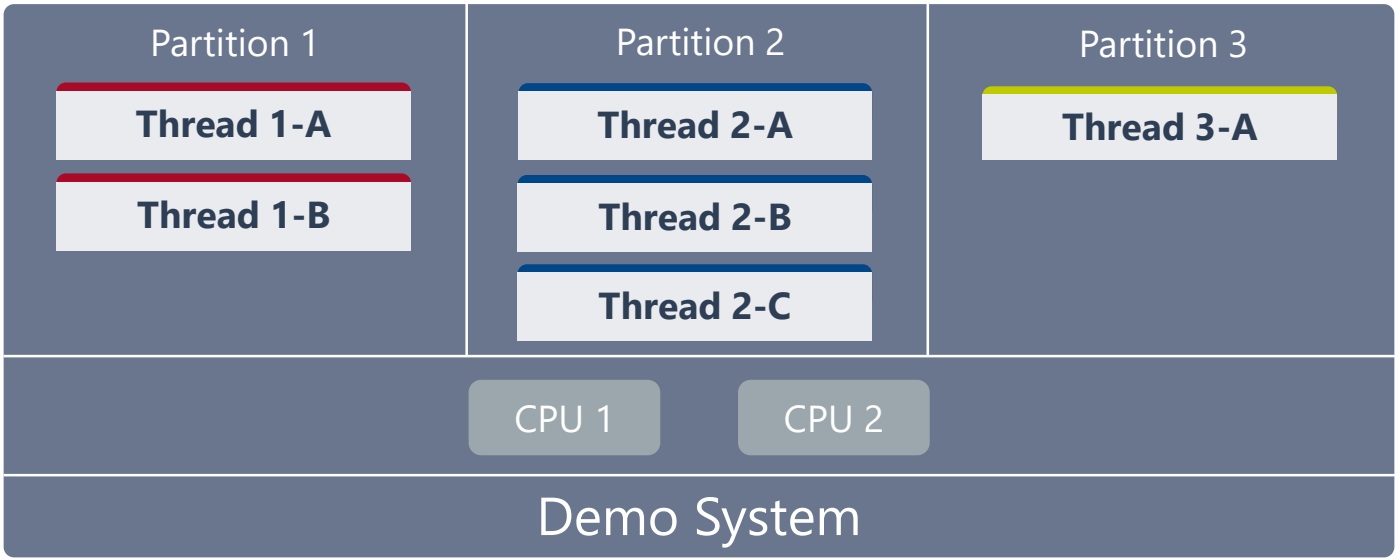


Mastering Parallel Execution of Code

Partition the system by criticality

Approach

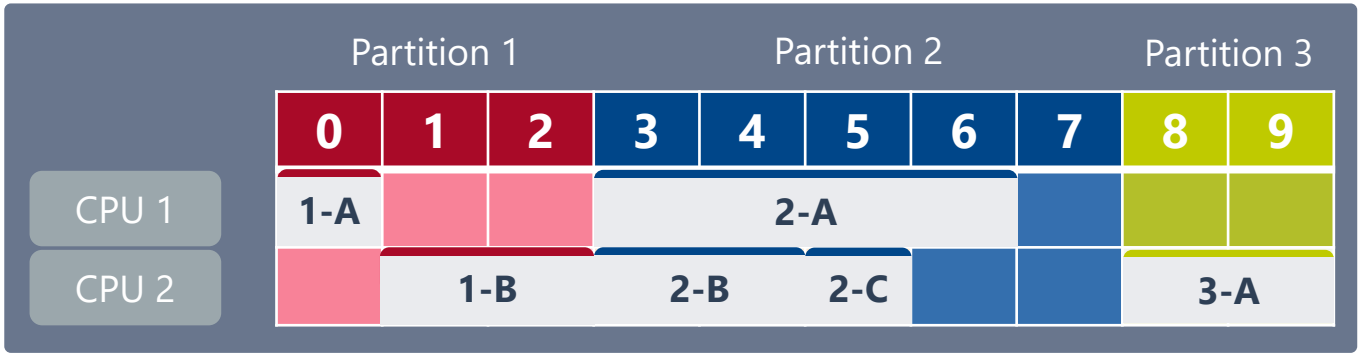
- Partition the system based on the criticality and introduce a second stage scheduler



Mastering Parallel Execution of Code

Intra-partition parallelism – symmetrical multi-processing

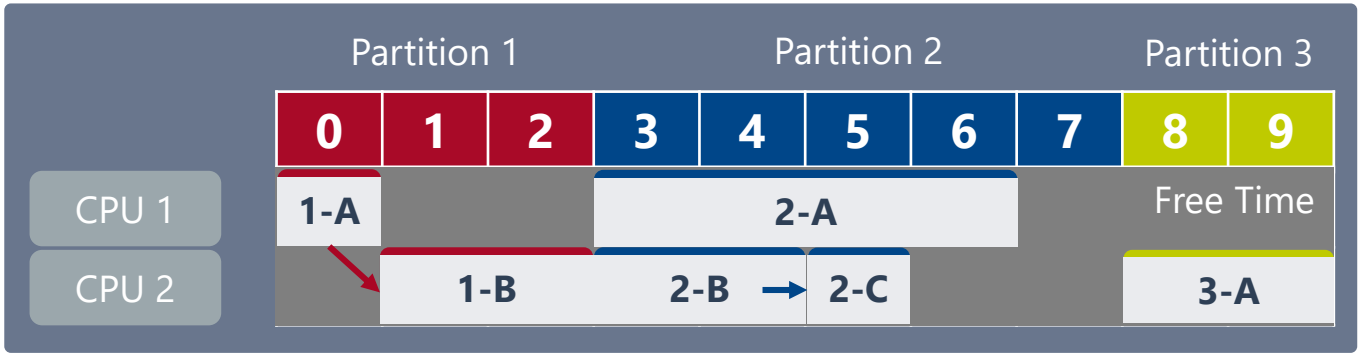
- Partitions are activated on each core
- Inside the partition threads may be executed in parallel on different cores



Mastering Parallel Execution of Code

Intra-partition parallelism – symmetrical multi-processing

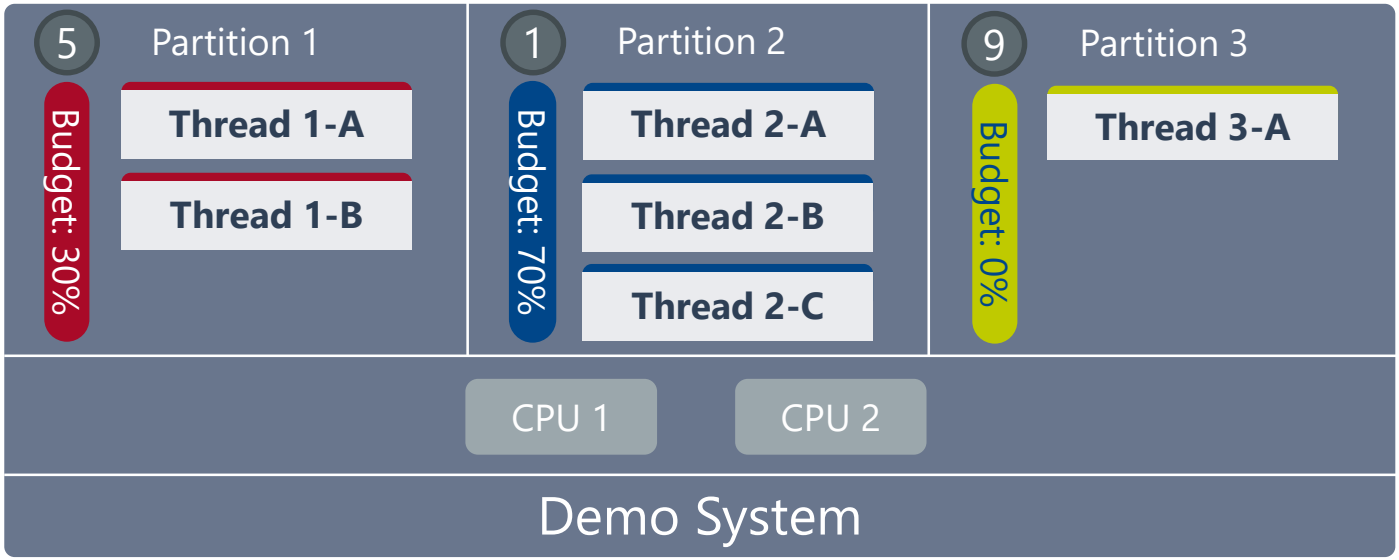
- Partitions are activated on each core
- Inside the partition threads may be executed in parallel on different cores
- Problematic:
  - Dependencies within a partition lead to available free-time which could not be used by other threads living in other partitions



Mastering Parallel Execution of Code

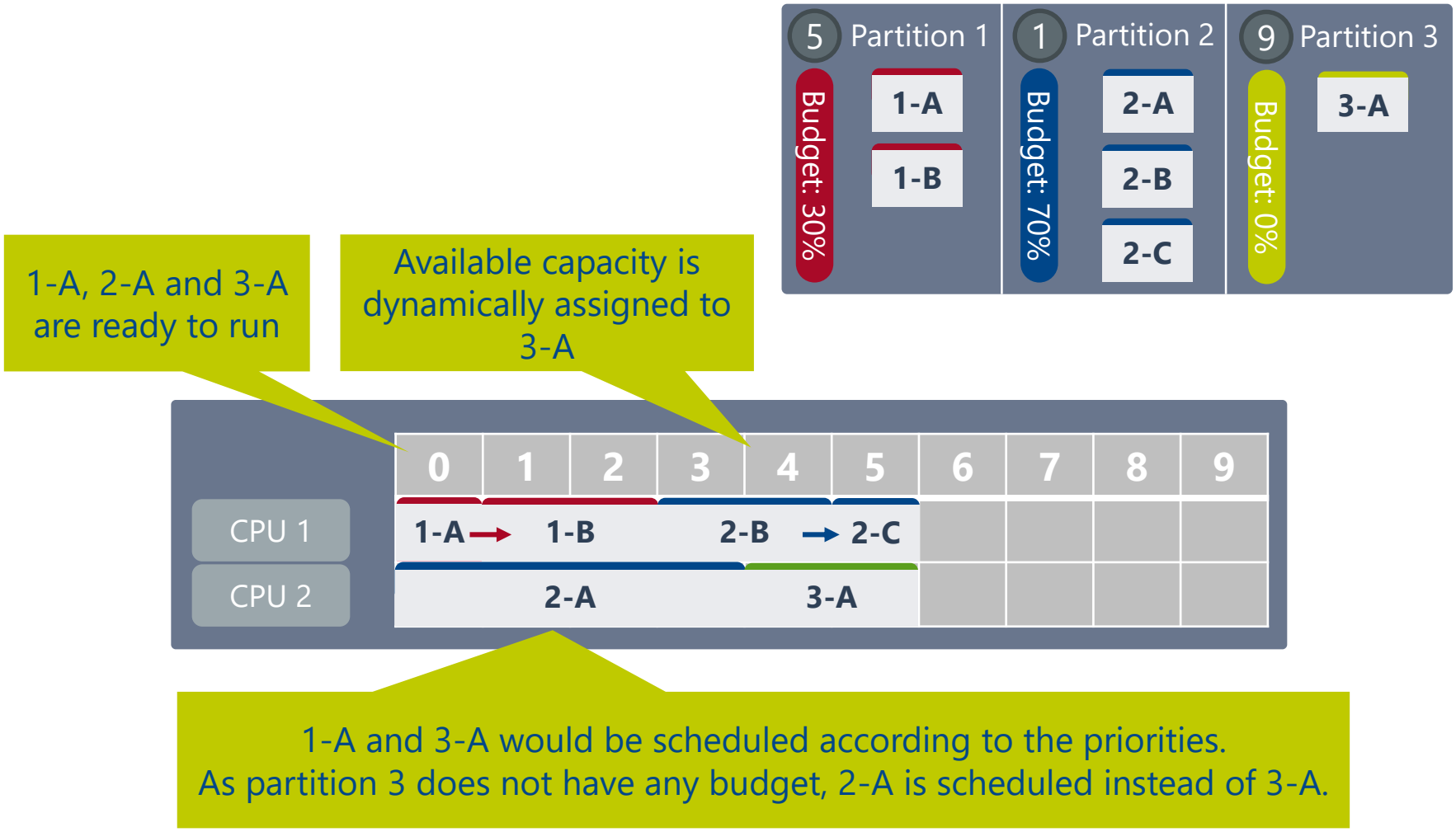
Adaptive Partitioning

- Instead of assigning static timeslots to partitions, individual budgets which are replenished periodically are defined
- Scheduler runs the thread with the highest priority while the threads partition still has budget available.



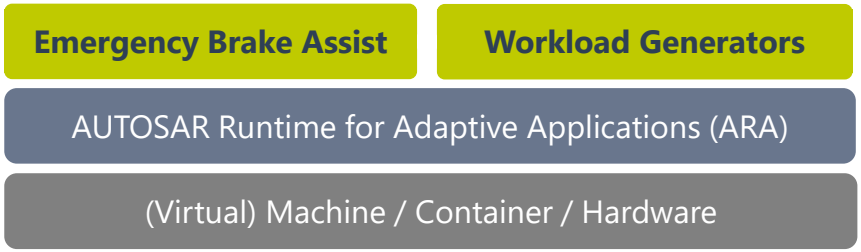
Mastering Parallel Execution of Code

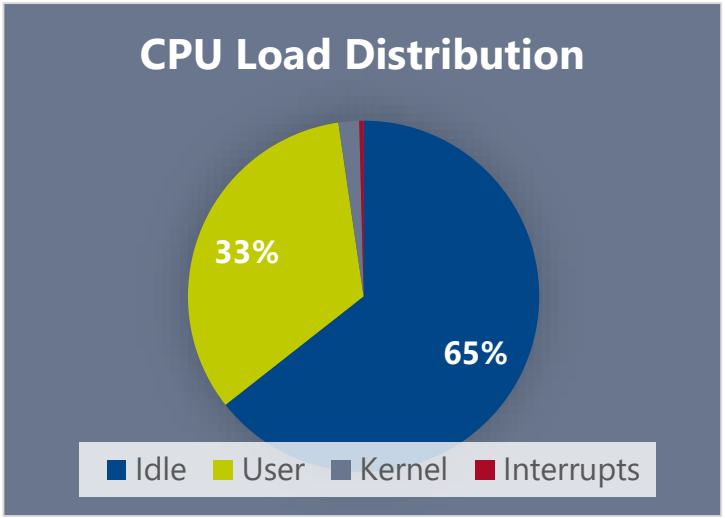
Available CPU capacity is used whenever possible

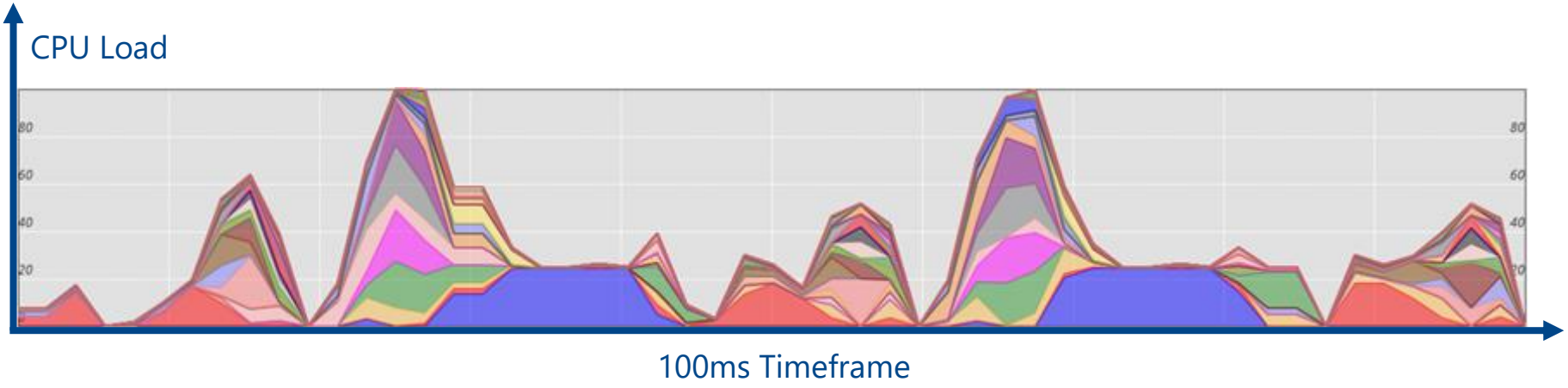


Mastering Parallel Execution of Code

Adaptive partitioning – real target evaluation



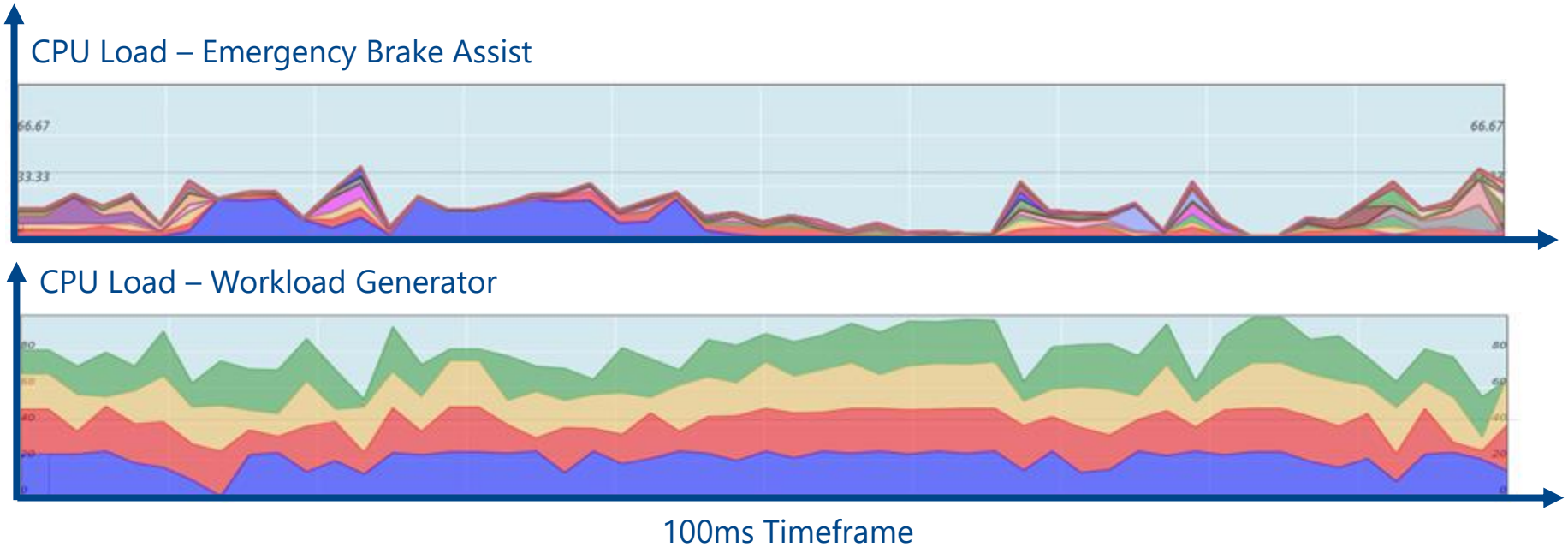




Mastering Parallel Execution of Code

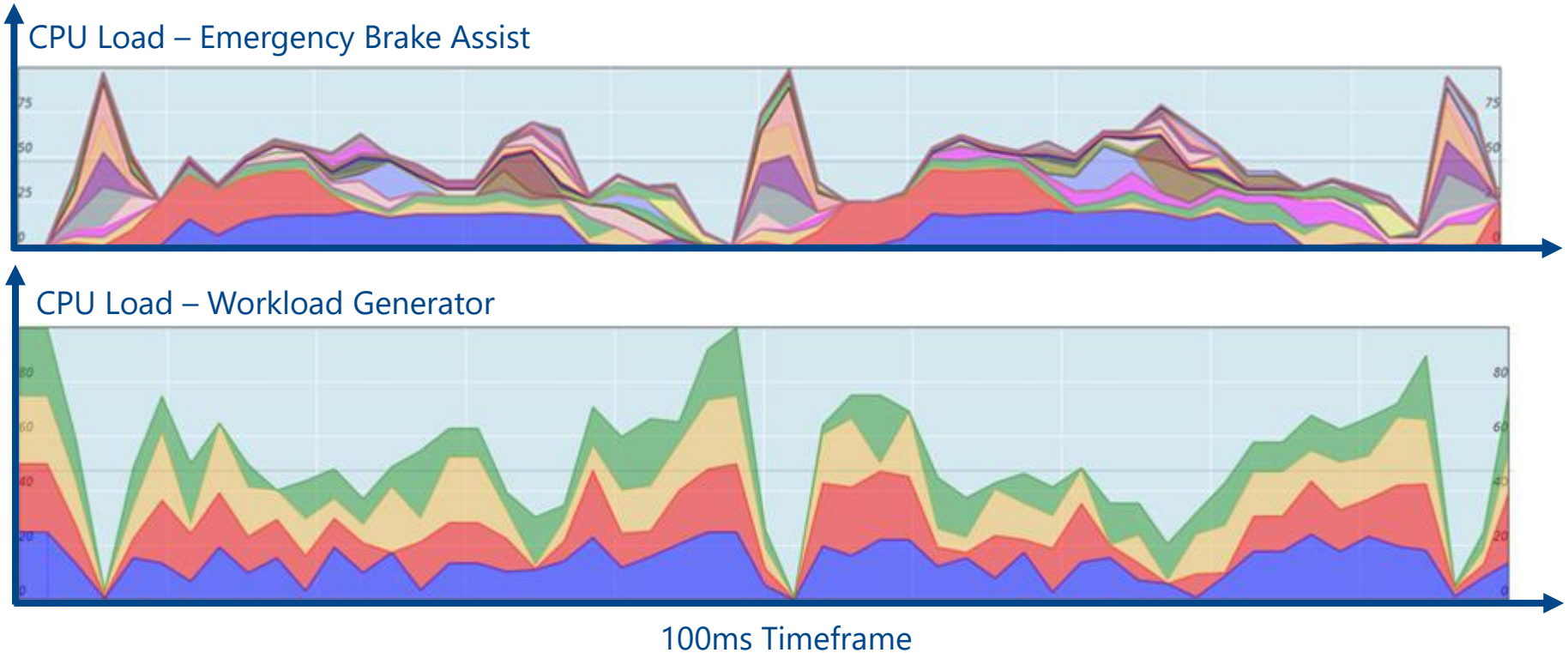
Four high frequency threads bring down the system

- **No** freedom from interference !
- System without adaptive partitioning is unable to provide its functionality



Mastering Parallel Execution of Code

Adaptive partitioning ensures freedom from interference



Mastering Parallel Execution of Code

Adaptive partitioning is not the silver bullet

Adaptive partitioning…

- … guarantees that every partition gets its assigned budget within a defined time window
- … does not directly influence the scheduling behavior within a partition but may introduce additional runtime delays

- 1:1 mapping of adaptive applications to partitions is not feasible

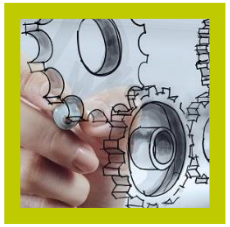
- Suitable scheduling algorithm for systems with aperiodic threads assigned to partitions can be based on the slack time

Conclusion

Conclusion

Extensive software separation and isolation is key requirement for central vehicle computers

POSIX scheduling policies may not guarantee proper execution for all real-time scenarios



Dynamic system partitioning allows borrowing available free time

Dynamic system partitioning is not silver bullet

info@itk-engineering.com
+49 (0)7272 7703-0

www.itk-engineering.com
www.itk-career.com