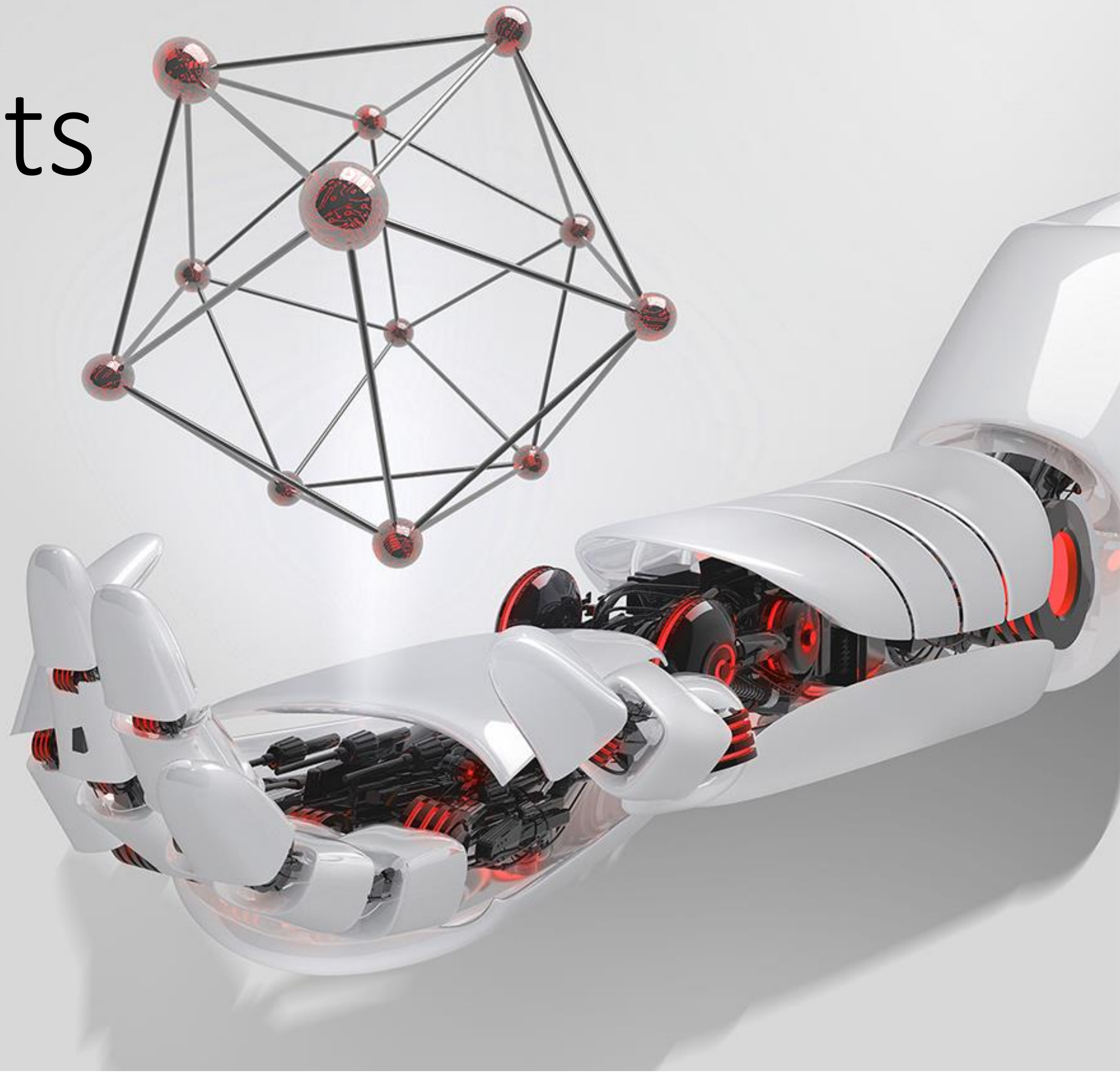
Functional Safety Aspects in Mixed Criticality Systems



HUAWEI | EVIDENCE | PISA RESEARCH CENTER

EVIDENCE® EMBEDDING TECHNOLOGY

HUAWEI

Who is this guy?

Current Role:

- Functional Safety Manager in Evidence – Huawei Pisa RC

Work Experience:

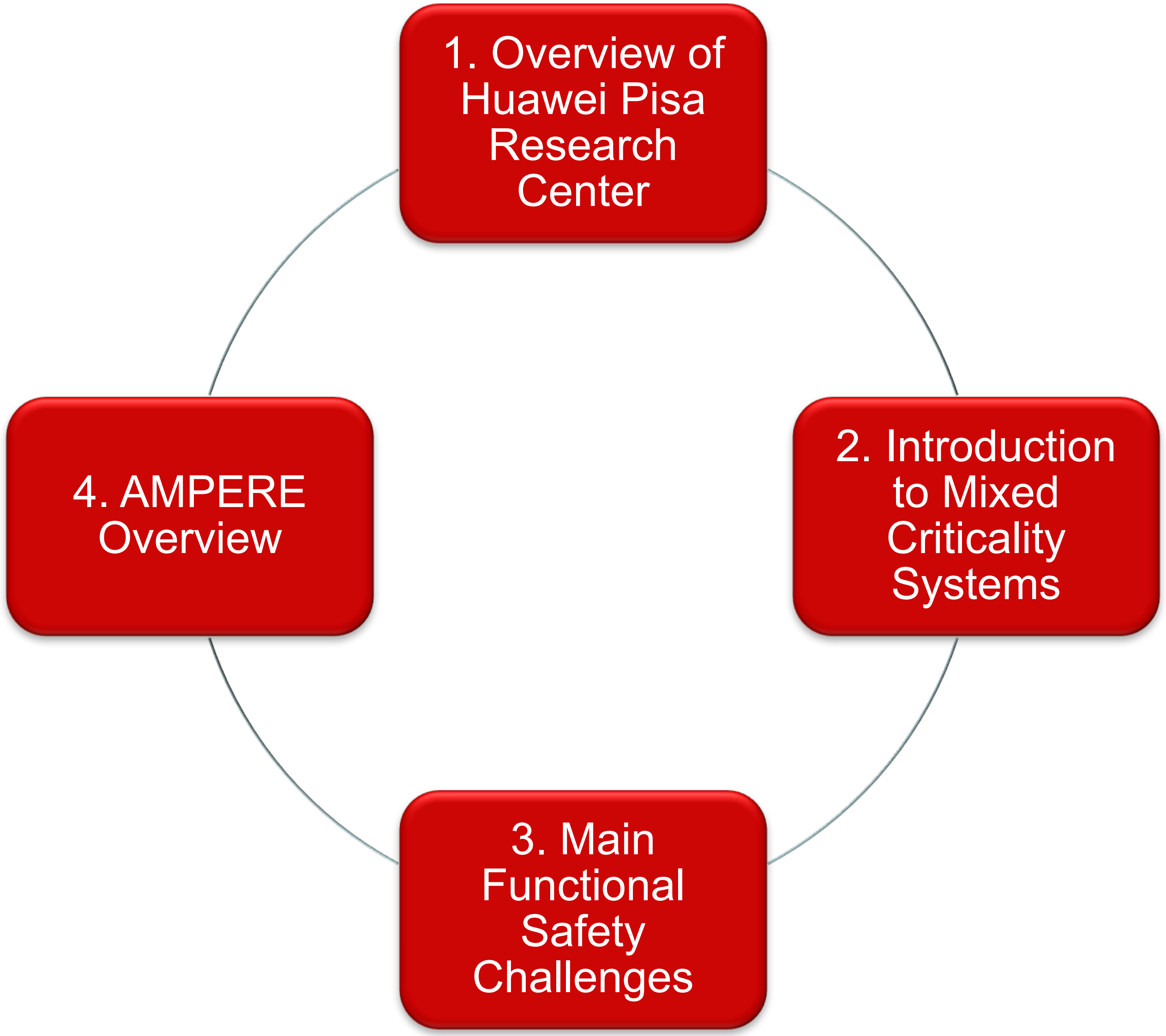
5+ years of experience in automotive and Functional Safety field

- Head of Safety Competence Center @Marelli (Venaria site)
- Functional Safety Lead Engineer @AVL Italia (Cavriago site)

Contact Info:

- Email: fabrizio.tronci@huawei.com
- Job location: Via Francesco Squartini, 42 - Pisa

Contents



1. Overview of Huawei Pisa Research Center

2. Introduction to Mixed Criticality Systems

3. Main Functional Safety Challenges

4. AMPERE Overview

Where we are: part of Godel Lab

Evidence S.R.L purchased by Huawei on September 6th 2019, and now forms the Embedded Software Lab under the administration of Pisa Research Center.

The current business is under the management of Godel Lab in Huawei 2012 Laboratories (under Huawei European Research Institute), mainly focusing on real-time embedded software with Safety attributes.

Major Interests:

- Automotive level operating system and basic software
- Certifiable model based development and code auto-generation
- Functional Safety analysis, testing and certification on embedded Software

ERIKA3 was Open-Source (Huawei is going to **keep its open source nature**)

Main competences

Automotive embedded OS

Os, Basic Software, Safety

ERIKA3 Open source AUTOSAR OS

Automotive grade Basic SW

ISO26262 Safety Engineering

Wide experience on MCUs, IoT

Model based design and Automotive Tool-chain

eclipse

AUTOSAR Classic

VFB and RTE Generator

EFORMS configuration plugins

E4CODER E4Coder code generation tool

Code generation from MathWorks functional models LabVIEW

High-performance OS

POSIX OS for MCUs

Linux embedded

Kernel drivers

U-Boot

Hypervisors / QEMU

Android

Ubuntu Core

VxWorks for industrial devices

Qt graphical libraries

Mixed Criticality Systems

Mixed-Criticality System (MCS) is a real-time system that comprises tasks having two or more distinct criticality levels.

There are two conflicting trends in the design of such systems:

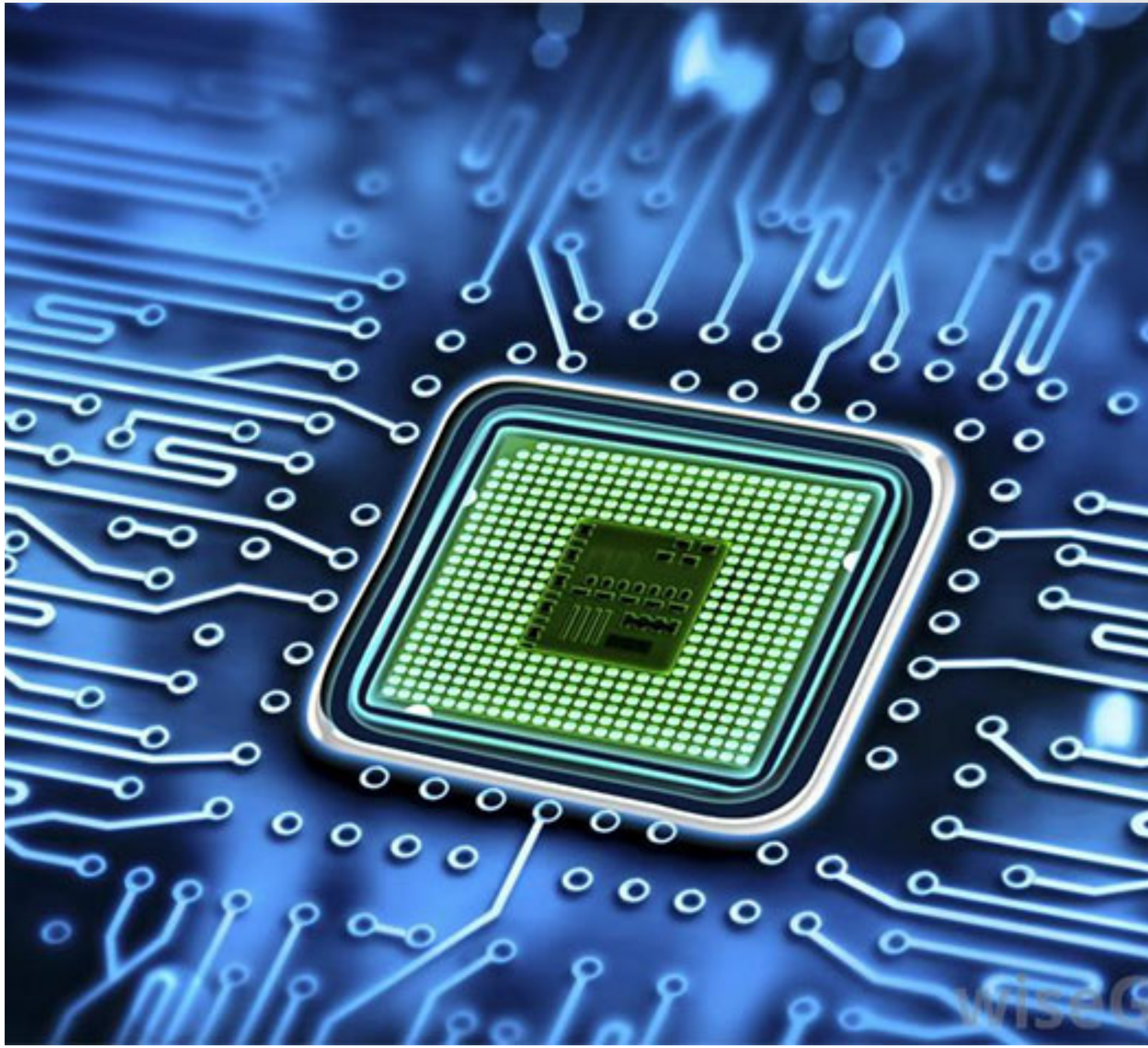
- Safety requirements are being increasingly emphasized
- More functionalities are being implemented on integrated platforms due to size, weight and power (SWaP) constraints.

Weight

Space

Power Consumption

Performance



Functional Safety Concerns

New detection, monitoring and reaction solutions

Exiguous methods in current FuSa standards

HW support for high criticality applications

(integrity, FIT…)

SW support for high criticality application

(OS, Hypervisor...)

Increasing complexity of Software Safety Analysis

About possible Improvements

Improvements in ISO26262 Part 11-5.4 for Multi-processor systems

Evidence is interested in joining ISO 26262 group and is active in European working groups

Independence in Mixed Criticalities

HIGHEST ASIL APPLICATION DEVELOPMENT

Develop all components of the system (including the non safety-related ones) according to the highest criticality level present on the system

✗ Generally cost and effort prohibitive

FREEDOM FROM INTERFERENCE APPROACH

Provide enough evidence of independence among the applications and develop each application according to its criticality level

? Guarantee that the coexisting applications do not corrupt the functional behavior and timing correctness

Main Isolation Topics

? Guarantee that the co-existing applications do not corrupt the functional behavior and timing correctness

SPATIAL PARTITIONING

The property of containment with respect to faults affecting resources shared by more than one application in the system.

For instance, a faulty application might change the configuration of a shared peripheral in an unpredictable way

TEMPORAL PARTITIONING

The property of containment with respect to faults affecting the timing of an application.

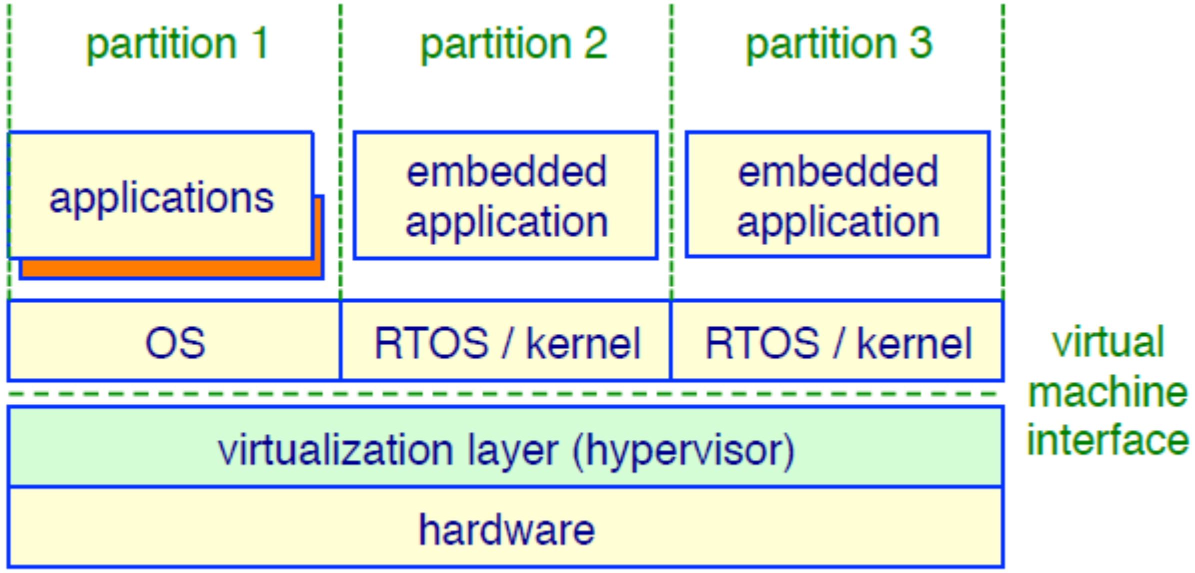
For instance, a faulty application might abuse a shared communication channel, causing a delay in other applications

Isolated Virtual Machines

- Virtualization provides a means to run a number of virtual machines (VM) on a single hardware platform. Each virtual machine has a set of virtual resources that are mapped to the available physical resources.

- An hypervisor takes care of implementing isolation by scheduling the execution of partitions in separate time frames and providing isolated memory spaces for them.

- Each VM is possibly equipped with a different operating system depending on the criticality requirements of the subsystems running on it



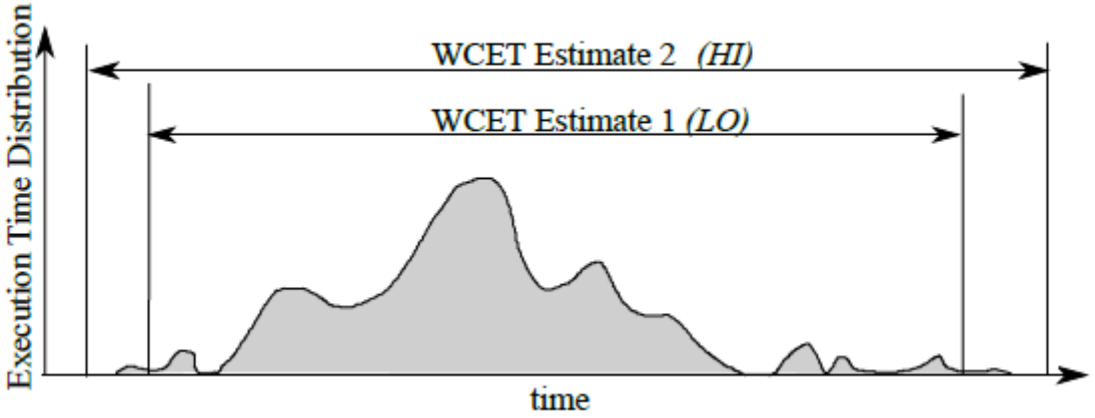
Possible technique: Hypervisor

- Hypervisors can be classified into two types:
  - **Type-I hypervisor** directly runs on the hardware without relying on a host operating system. Such hypervisor has to bring its own set of device drivers and low-level system mechanisms (e.g. virtual memory management).
  - **Type-II hypervisors** on the other hand rely on a host operating system to run on. They leverage the operating system facilities, which are already in place and run as a normal process. However, the host operating system has to cooperate with the hypervisor process and reflect specific types of exceptions back to this process.

- Hypervisor became a bottleneck and potentially source of dependent failure initiators
- Typically not able to provide sufficient prevention or detection of permanent or transient faults affecting the MPSoCs
- Used application impacts on safety development/qualification

Theoretical modelling

- The MCS model differs from the traditional real-time task model because of the Worst Case Execution Time (WCET) uncertainty
- The majority of MCS models have only two criticality levels: LO and HI
- Run-time techniques are used to monitor execution times of running tasks



NORMAL CONDITIONS

- Each task has WCET(LO) and WCET(HI), where WCET(LO) ≤ WCET(HI).
- The system operates initially in a normal mode
- Both high and low criticality tasks utilize the hardware resources considering WCET(LO).

EMERGENCY CONDITIONS

- If a critical task exceeds its WCET(LO), the system switches to a degraded mode
- All low criticality tasks are suspended
- Resources are allocated to HI task and WCET(HI) is considered

Challenges with current model

- No guarantees are given to lower-critical tasks. The dual-criticality model views lower-critical as non-critical (not always true)
  - Instead of directly switching the mode and suspending lower-critical tasks, the memory service guarantees those tasks are degraded to reduce the interference on the higher critical to accommodate for the increase in the execution time
  - Keep other noninterfering cores running the same set of tasks (i.e. no effective mode switching), while switching only the necessary core(s).
  - Migrating tasks from SRT (soft real time) cores to HRT (hard real time) cores (not applicable in Classic AUTOSAR environment)

CONS: Switching-mode overheads

MPSoCs heterogeneity impacts run-time decision!

Temporal Partitioning

- WCET is usually a pessimistic estimation and the WCET of a task with higher criticality is even more pessimistic.
- The high pessimism of WCET estimations leads to the inefficient usage of resources at run time.

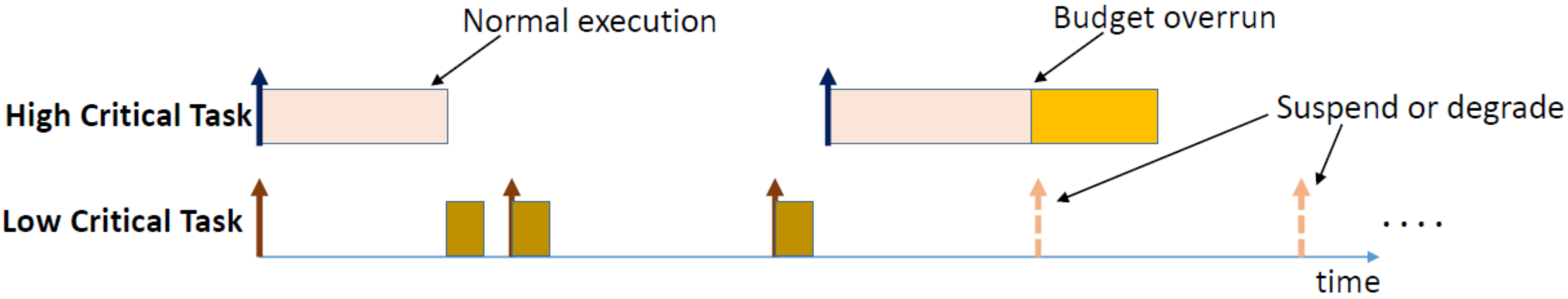
Multiple features impact Temporal Partitioning:

- Huge interference with the increase of processing elements (Pes) competing on the shared resources.
- Each type of PEs has its own memory access behavior, which complicates the analysis, leading to more WCET pessimism.
- Understanding the architectural details of shared resources (such as the interconnect and the memory hierarchy) is necessary to derive realistic WCET.

Reaction to Budget overrun

Budget Overrun

- **How to degrade?**
  - Reduced Budget
  - Increased Deadline
- **When to degrade?**
  - Immediately
  - As late as possible and in case stop the system
- **Which tasks to degrade?**



Challenges with Temporal Partitioning

- **Which tasks to degrade?**

  Allow the designer to choose the task to be degraded

SINGLE TASK OVERRUN

Specific degraded budget based on overrun **task**

MULTIPLE TASKS OVERRUN

Specific degraded budget based on the **criticality** of overrun tasks

⚠ WCET analysis and definition is a key point for safety recovery mechanisms implementation

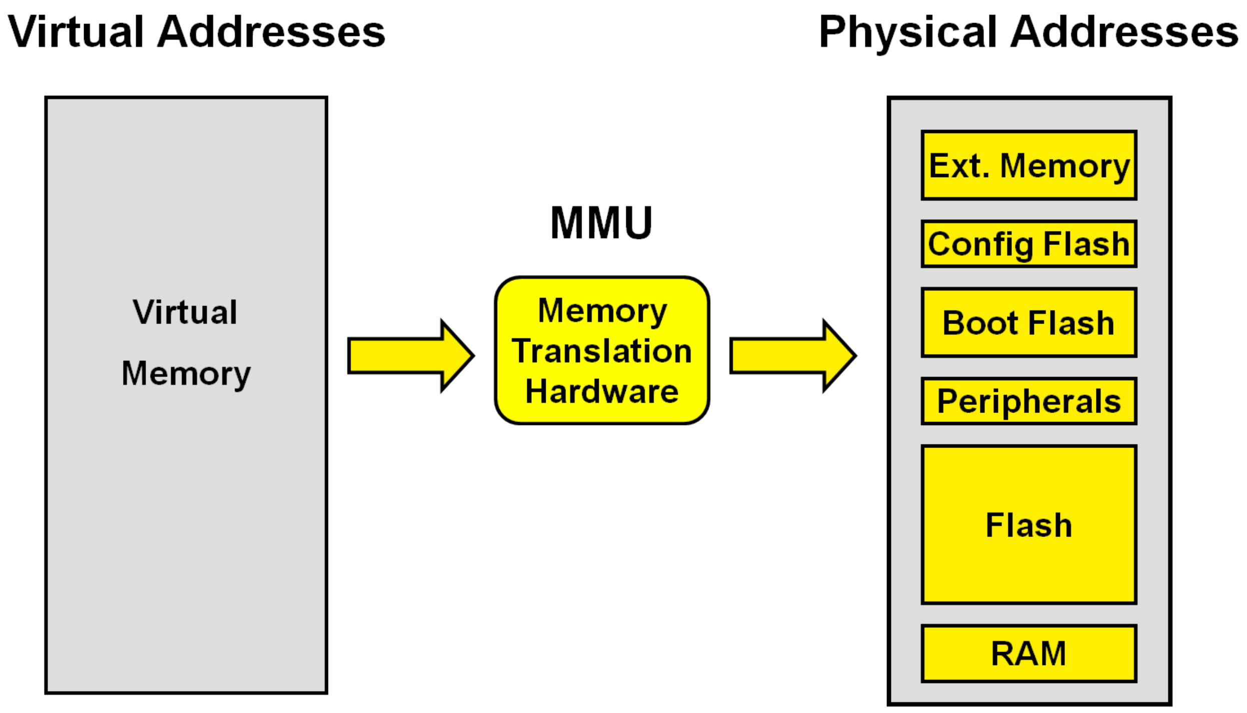
Spatial Isolation

- In MPSoCs based on Memory Management Units (MMU), to prevent software running in a partition from reading or writing into address space allocated to other partitions.

- MMUs (implemented in hardware) provide address translation mechanisms to map the logical memory space of an application to the regions of physical memory that are allocated to it.

- Usually MMUs provide sophisticated memory management schemes as they are designed to support complex paging and virtual memory in general-purpose operating systems.

- In MultiCore Microcontrollers is based on MPU (Memory Protection Unit)



Level of spatial isolation

The spatial partitioning can be used at different granularity levels, from finest to coarsest:

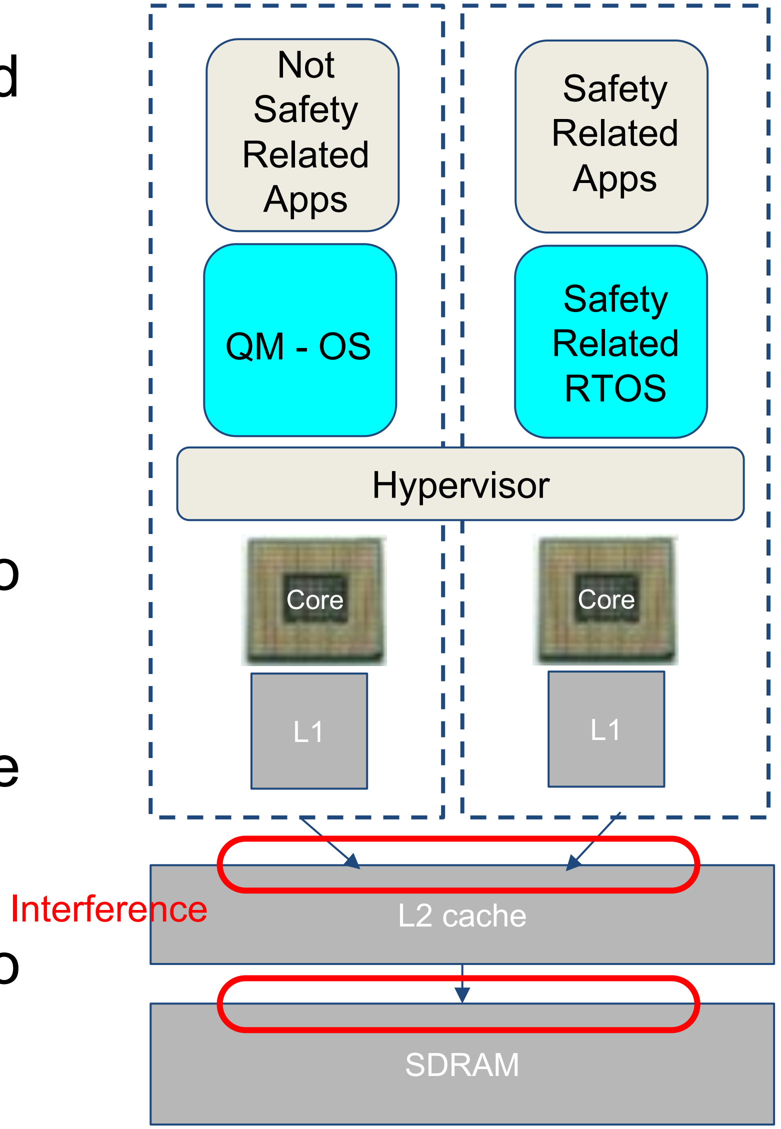
1. **Each task is isolated in its own partition:**
   - great number of partitions, which may even go beyond the support of the type-1 hypervisor.
   - noticeable number of IPC (inter partition communication) channels, introducing a performance overhead.

2. **Each application is isolated in its own partition (Classic AUTOSAR solution):**
   - the most suitable for integrating a limited number of applications
   - limited number of IPC channels

3. **Applications at the same criticality level share the same resource partition**:
   - reduces the need of IPC channels to a bare minimum

Partitioning is always the solution?

- Even with partitioning, there are still some interferences on shared hardware:
  - e.g. caches, memory bus, etc.
  - One core can affect the real-time responsiveness of other cores
- Software solutions:
  1. **Cache coloring** to avoid data eviction: handle virtual memory so that pages with different "colors" have different positions in cache
  2. **Memguard** forces memory bandwidth by monitoring performance counters
  3. **Co-scheduling algorithms** orchestrating memory accesses to determine the subset of applications that should share cache



Challenges with Spatial Isolation

The architect still has the possibility to almost eliminate the cache effects at system level:

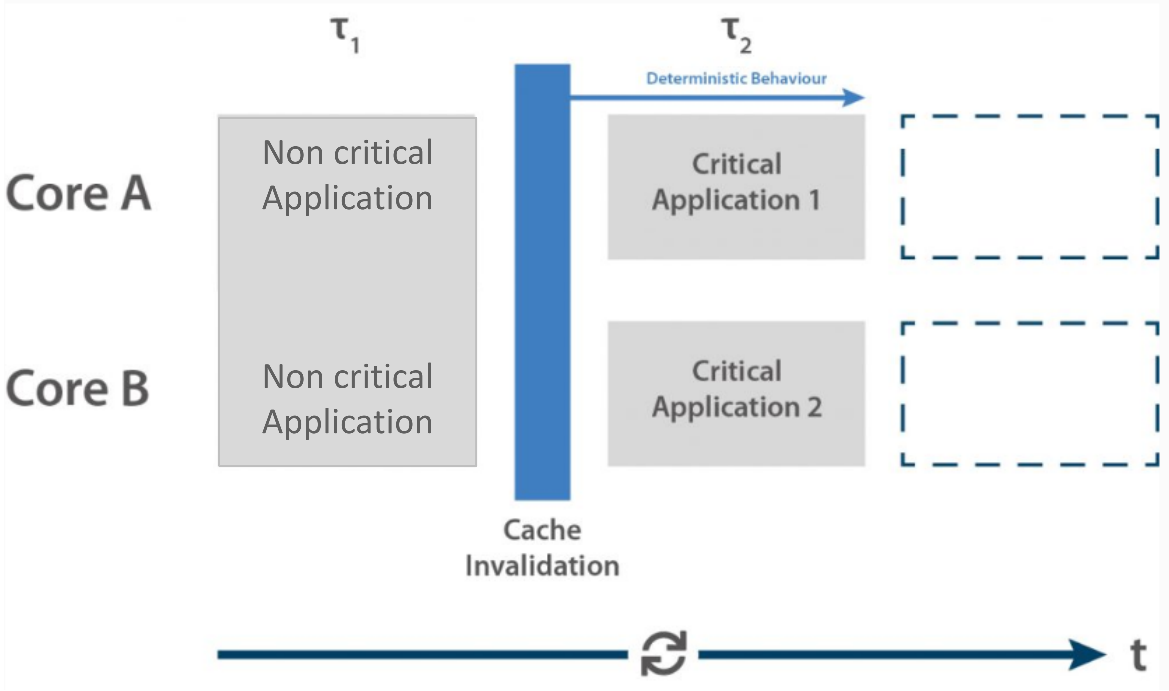
1. Increase the safety margins of the time partition windows, so that the WCET can be met

   👎 Performance impacted

2. Applications at the same criticality level share the same resource partition:

   - Cache can be invalidated at the start of the critical time frame

   👎 Intensive usage of the memory bus by one application is on the expense of applications running in parallel.



⚠ **Shared Processor Caches and Interference Channels impact Cache Coherency**

EU Funded Project – AMPERE overview

.AMPERE

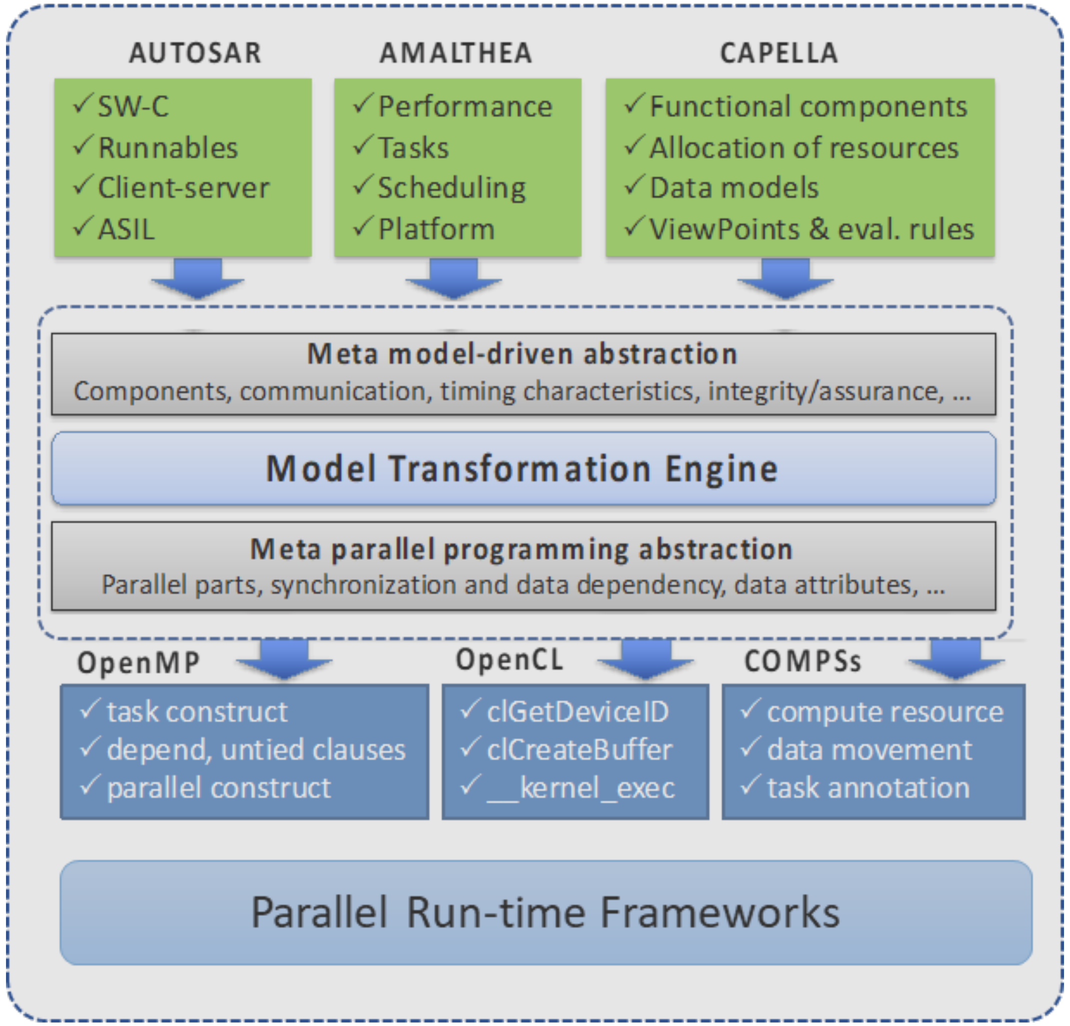
Model-driven development for highly Parallel and EneRgy-Efficient computation supporting multi-criteria optimisation

From 01/01/20 to 31/12/22

Develop a **novel software architecture** capable of:

1. Capturing the component definition and non-functional requirements described in the system model and transforming it to key parallel constructs
2. Fulfillment of non-functional properties described in the CPSoS (Cyber-Physical System-of-Systems) description
   - Energy-efficiency, safety and cyber-security, real-time response, resiliency and fault-tolerance, and testability
3. Efficient usage of advanced parallel and heterogeneous embedded architectures

AMPERE Framework



AUTOSAR

✓SW-C
✓Runnables
✓Client-server
✓ASIL

AMALTHEA

✓Performance
✓Tasks
✓Scheduling
✓Platform

CAPELLA

✓Functional components
✓Allocation of resources
✓Data models
✓ViewPoints & eval. rules

Meta model-driven abstraction
Components, communication, timing characteristics, integrity/assurance, ...

Model Transformation Engine

Meta parallel programming abstraction
Parallel parts, synchronization and data dependency, data attributes, ...

OpenMP

✓task construct
✓depend, untied clauses
✓parallel construct

OpenCL

✓clGetDeviceID
✓clCreateBuffer
✓__kernel_exec

COMPSs

✓compute resource
✓data movement
✓task annotation

Parallel Run-time Frameworks

AMPERE Partners and Use Cases

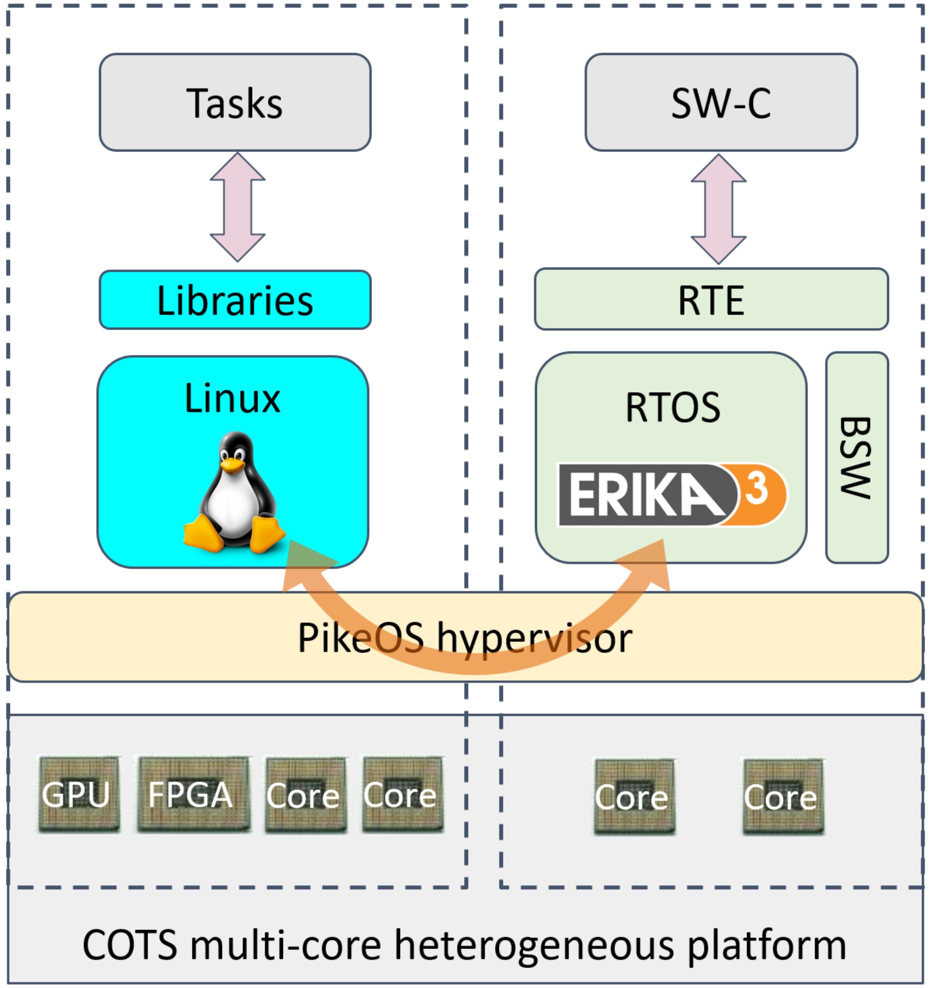
1. THALES: Obstacle Detection and Avoidance System (ODAS)

- ADAS functionalities (i.e. obstacle detection and collision avoidance) based on data fusion coming from tram vehicle sensors

2. BOSCH: Predictive Cruise Control (PCC)

- Extends Adaptive Cruise Control (ACC) functionality by calculating the vehicle's future velocity curve using the data from the *electronic horizon*
- Improve fuel efficiency (in cooperation with the powertrain control) by configuring the driving strategy based on predictive data analytics and AI methods

Evidence Contribution in AMPERE



Evidence will lead the Work Package

Operating systems and parallel platforms

- COTS parallel heterogeneous hardware platforms selection (up to 2)
- Porting on PikeOS hypervisor
- Software tracing on ERIKA
- POSIX PSE51 kernel development
- Validate the OSs and the hypervisor

Thank you

Send your CV to evidencehr@huawei.com

www.huawei.com

The information in this document may contain predictive statements including, without limitation, statements regarding the future financial and operating results, future product portfolio, new technology, etc. There are a number of factors that could cause actual results and developments to differ materially from those expressed or implied in the predictive statements. Therefore, such information is provided for reference purpose only and constitutes neither an offer nor an acceptance. Huawei may change the information at any time without notice.