Applications security in manycore platform: from operating system to hypervisor

Mehdi Aichouch[1], Clément Devigne[2], Guy Gogniat[3], Maria Mendez[3]

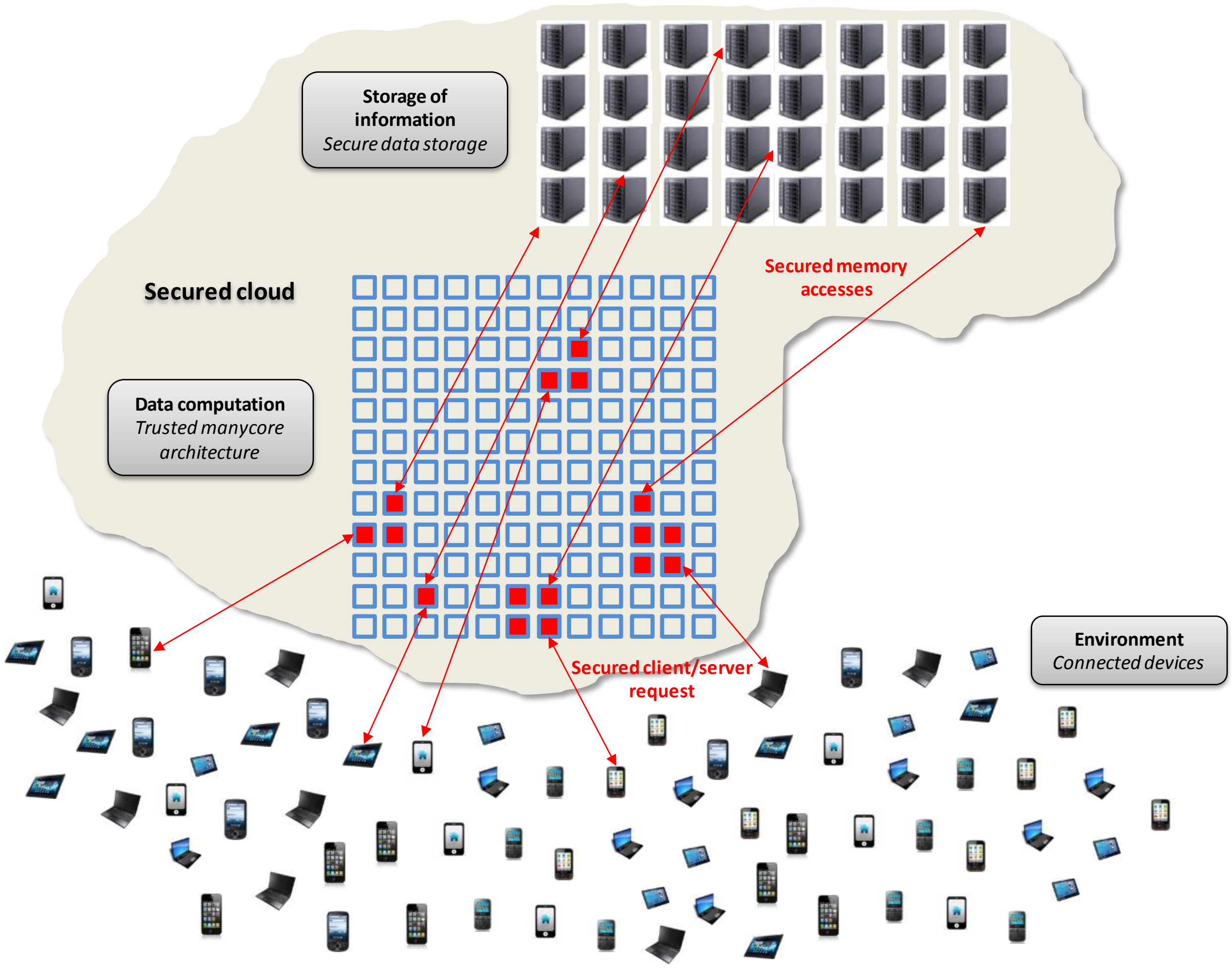
[1]CEA, Saclay, [2]LIP6, Jussieu, [3]Lab-STICC, Lorient

TSUNAMY Project (ANR)

CryptArchi'2015, Leuven, Belgium - June 29th, 2015

Manycore platform for trusted computing

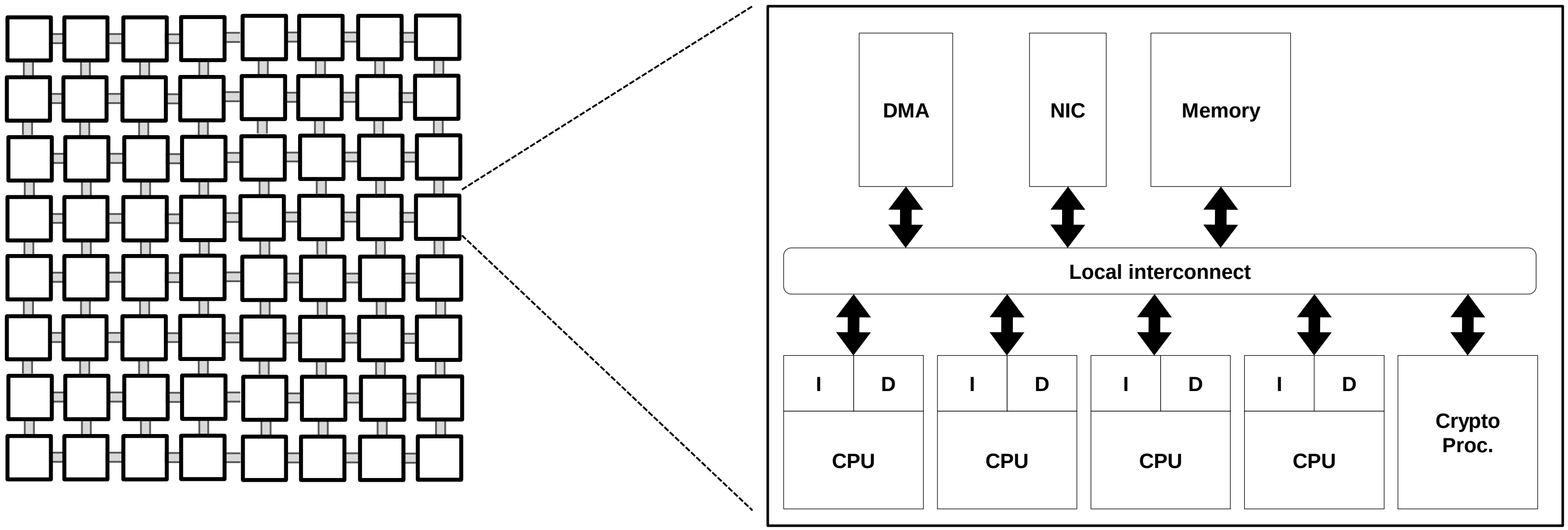
- **Context of cloud computing**
  - Many requests requiring security services
  - Secure data storage
- **Need of high performance and high security**
  - Enhance manycore platforms with cryptoprocessors
  - Virtual machines isolation
  - Applications isolation



Trusted platform for secure execution

- **Hundreds of clusters composed of**
  - Processing elements
  - Cryptoprocessor
  - Distributed memory

TSAR platform



Which threats?

- **Countermeasures depend on TCB (attack surface)**

Threat	Malicious element	Action	Countermeasure
Denial of service	Process (Application)	Over-consumption of resources	Software limitation
	Monitoring	Providing wrong information	Hardware monitoring
	Allocation	Stop reallocation	Hardware allocation
	Driver	Stop request to peripheral	Execution monitoring
	Interrupt controller	Behavioral modification	Hardware interrupt controller
	Control of memory access rights	Modification of access rights	Data ciphering and authentication
Information leakage	Process (Application)	Read peripheral output register	Register reset
	Scheduler	Access to scheduling information	Hardware scheduler
	Control of memory access rights	Read of access rights	Data ciphering and authentication
	Driver	Data copy on the physical memory	Reset memory zone
Unauthorized read and/or write of data in memory	Allocation	Modification of the virtual page assignation	Data ciphering and authentication
	Control of memory access rights	Modification of access rights	
	Driver	Data modification on physical memory	

Software countermeasure; Hardware countermeasure; Software or hardware countermeasure; Cryptographic countermeasure

Main issues

- **How to build a blind hypervisor?**
- **How to securely deploy virtual machines?**
- **How to securely map applications within one virtual machine?**

- **TSUNAMY project addresses these issues relying on TSAR manycore platform and ALMOS operating system**
  - https://www-soc.lip6.fr/trac/tsar
  - https://www-soc.lip6.fr/trac/almos

Agenda

- **Part1: Blind hypervisor in a nutshell**
  - Mehdi Aichouch

- **Part2: Executing Secured Virtual Machines within a Manycore Architecture**
  - Clément Devigne

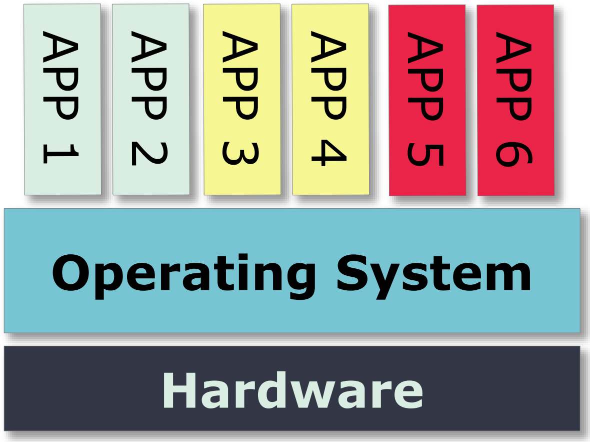
- **Part3: Secure application deployment**
  - Maria Mendez

Part1: Blind hypervisor in a nutshell

Mehdi Aichouch

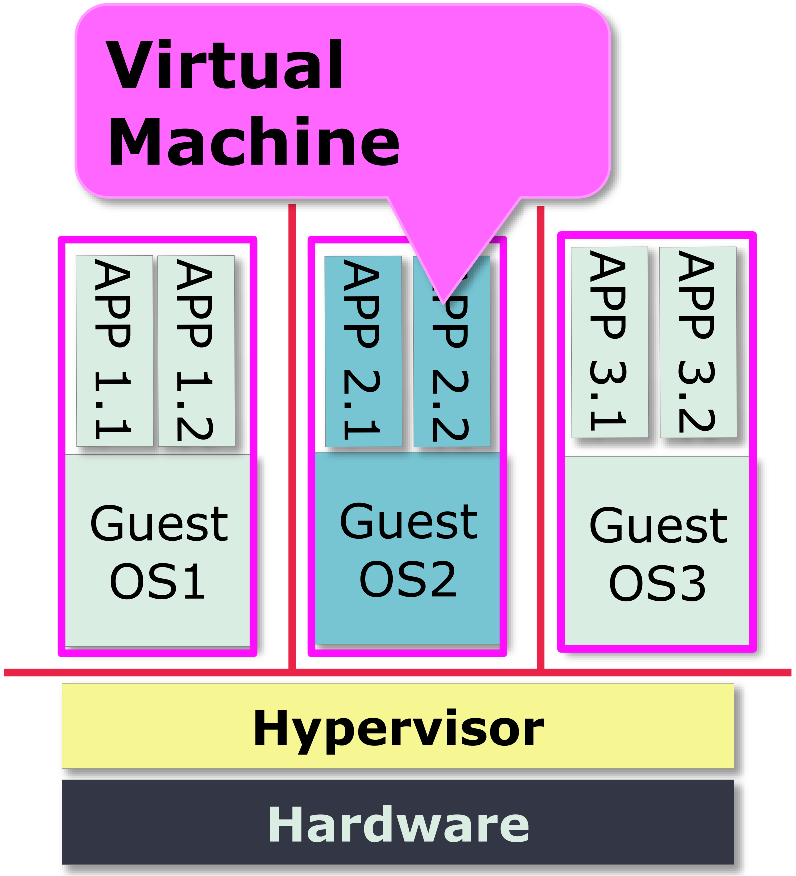
Background

- Traditional computing systems
  - ***Operating System*** *deployed on bare hardware*
  - ***User applications*** *installed on top of an operating system*



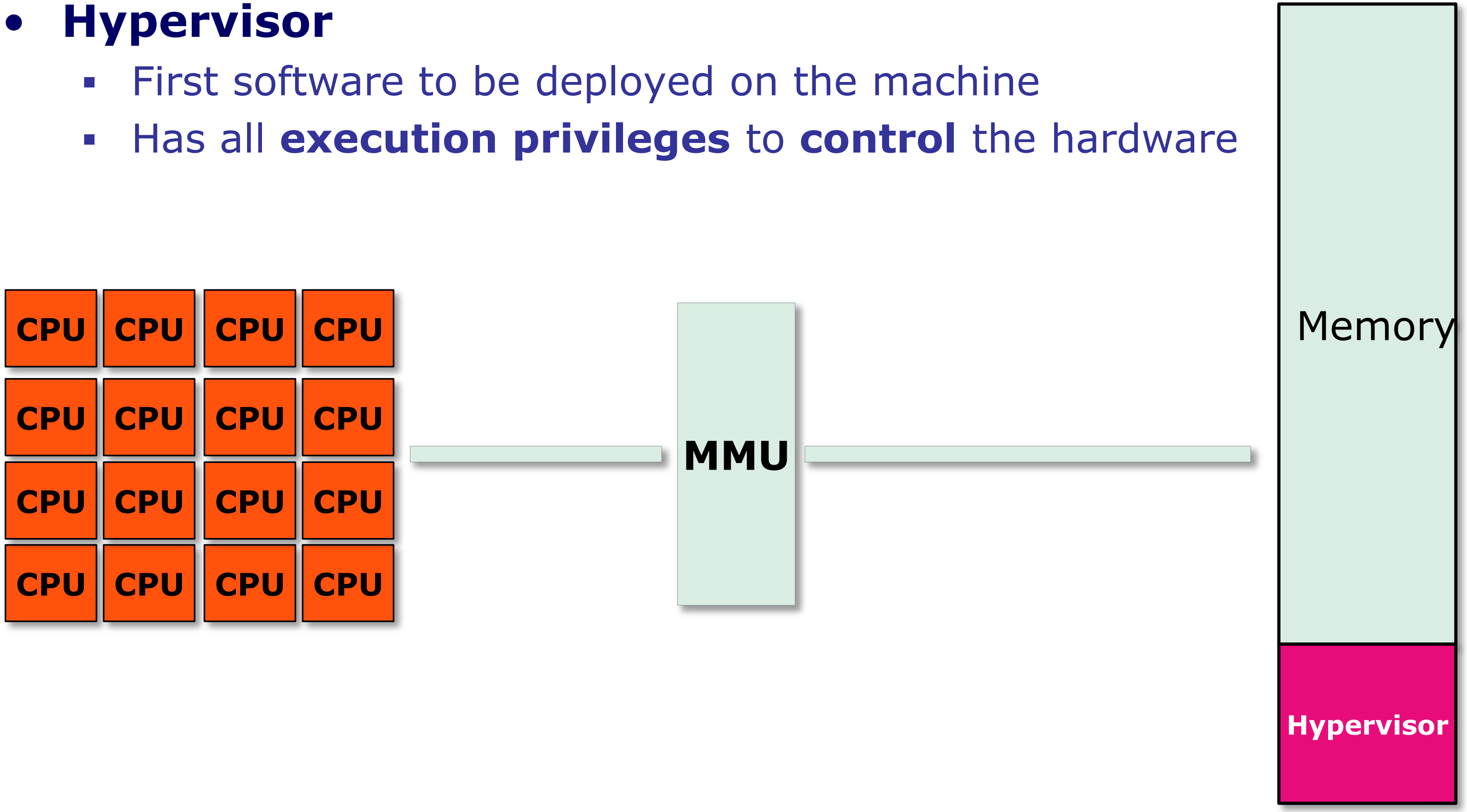
Virtualization

- **Hypervisor** deployed on the hardware
  - provides **virtual machine**
    - *Virtual* ***processor***
    - *Virtual* ***memory***
    - *Virtual* ***I/O devices***
  - manages a set of **protected** and **isolated** virtual machines
- **Uses cases**
  - **Multiple** operating systems deployed **simultaneously** on the same hardware
  - Run legacy OS/application on new hardware
  - Cost reduction through resources sharing



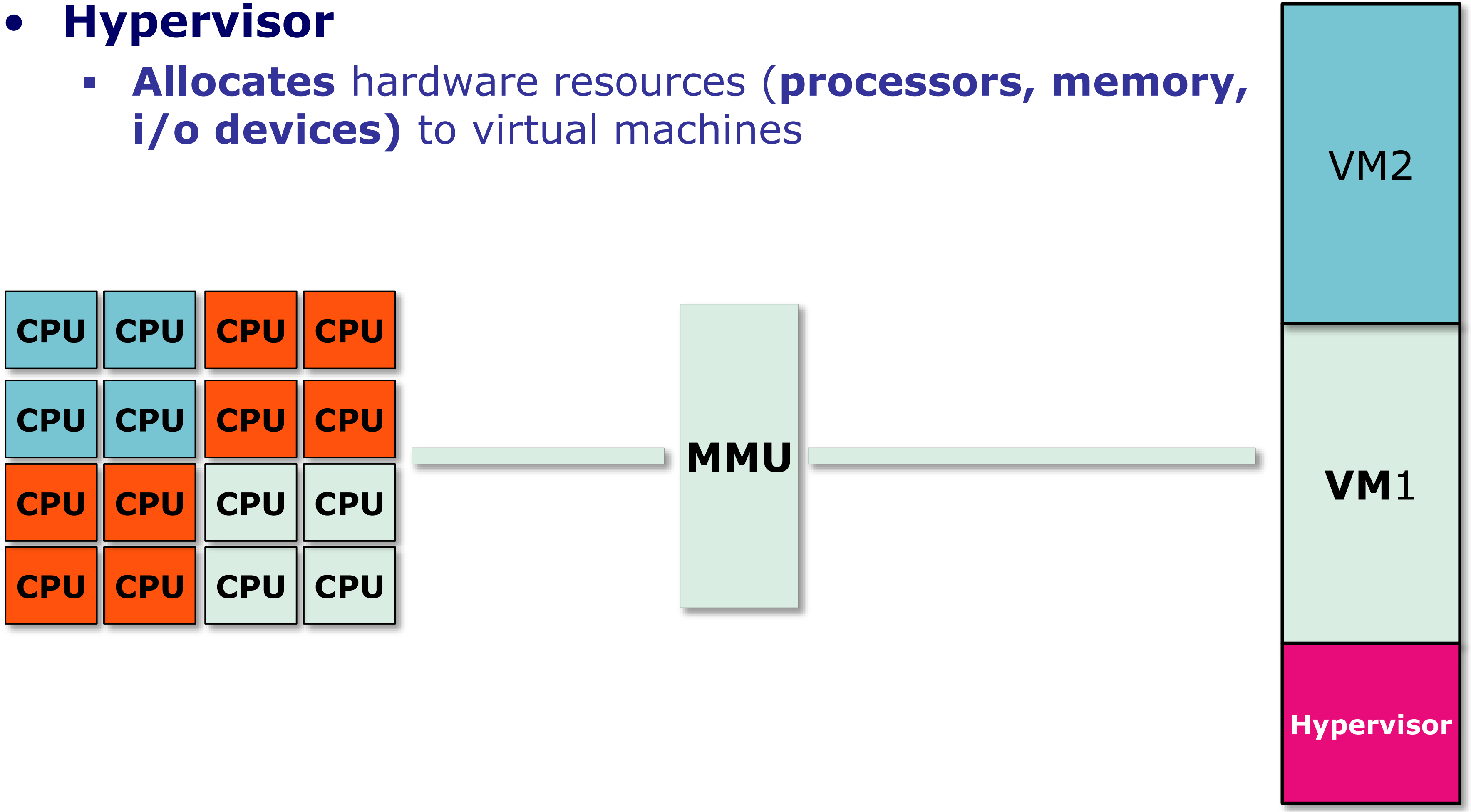
Hypervisor functionality

- **Hypervisor**
  - First software to be deployed on the machine
  - Has all **execution privileges** to **control** the hardware



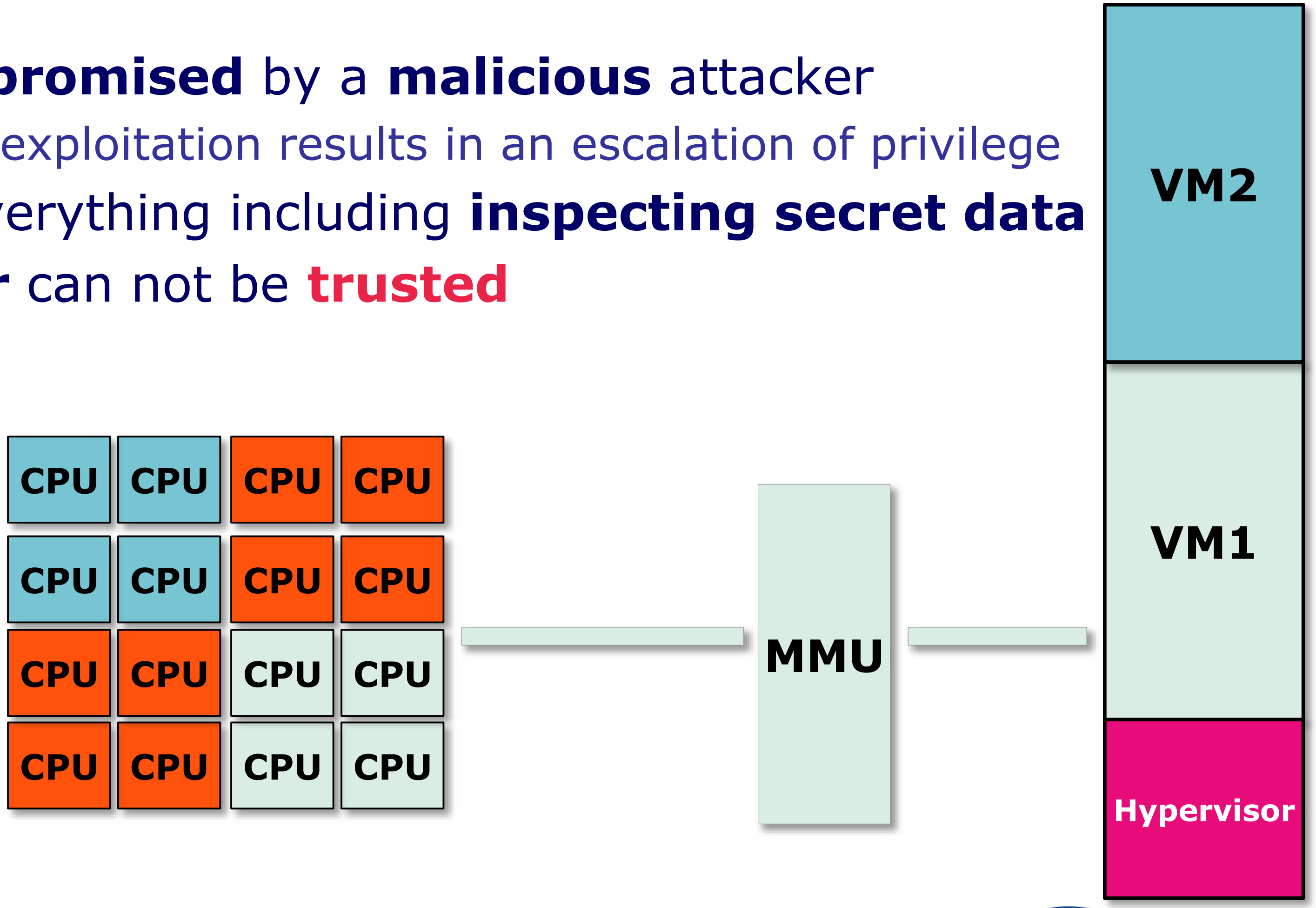
Hypervisor functionality

- **Hypervisor**
  - **Allocates** hardware resources (**processors, memory, i/o devices)** to virtual machines



Problem

- **Hypervisor** has access to virtual machines' **memory partitions**
- If it is **compromised** by a **malicious** attacker
  - e.g. BUG exploitation results in an escalation of privilege
- It can do everything including **inspecting secret data**
- **Hypervisor** can not be **trusted**



Blind Hypervisor

- **Goal**
  - Guarantee the **confidentiality** and **integrity** of VM's content even if **Hypervisor not trusted**
    - ***Confidentiality*** *means no read access permission*
    - ***Integrity*** *means no write access permission*

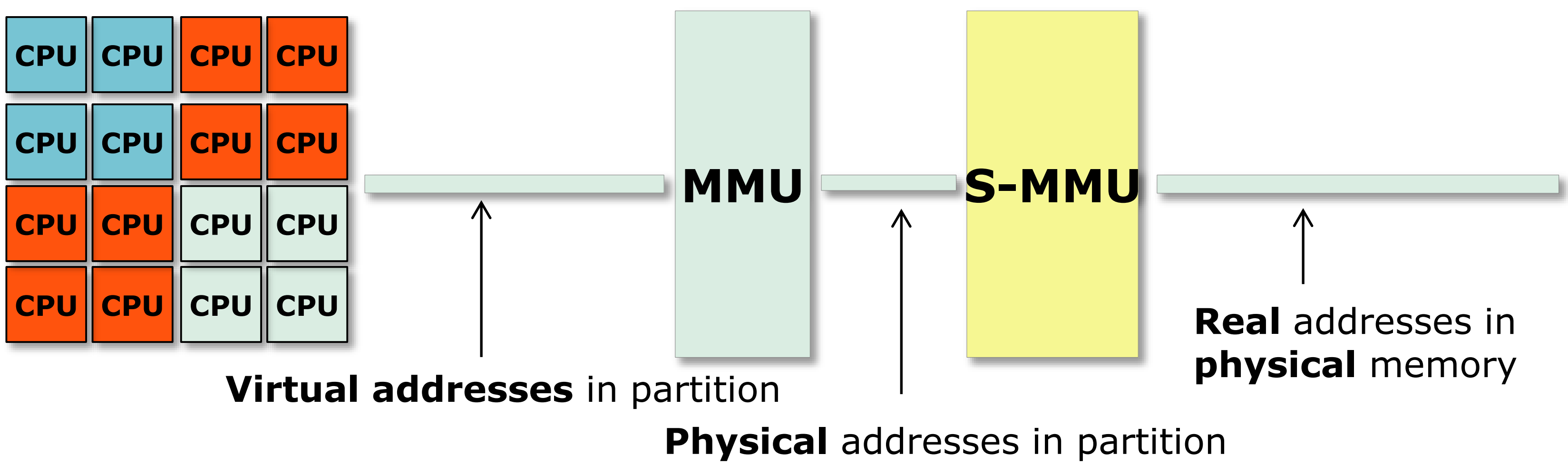
- **Protection scope**
  - Software attacks from virtual machines and hypervisor are avoided
  - Do not address physical attacks

Blind Hypervisor

- **Design**
  1. Prevent a hypervisor from accessing virtual machines **memory partitions**
  2. Virtual Machine **content** should be encrypted when stored on a hard disk or retrieved from a network
  3. Without **affecting** the original runtime performance
  4. Rely on a set of hardware assisted techniques
     - *hardware intrusiveness trade-offs*

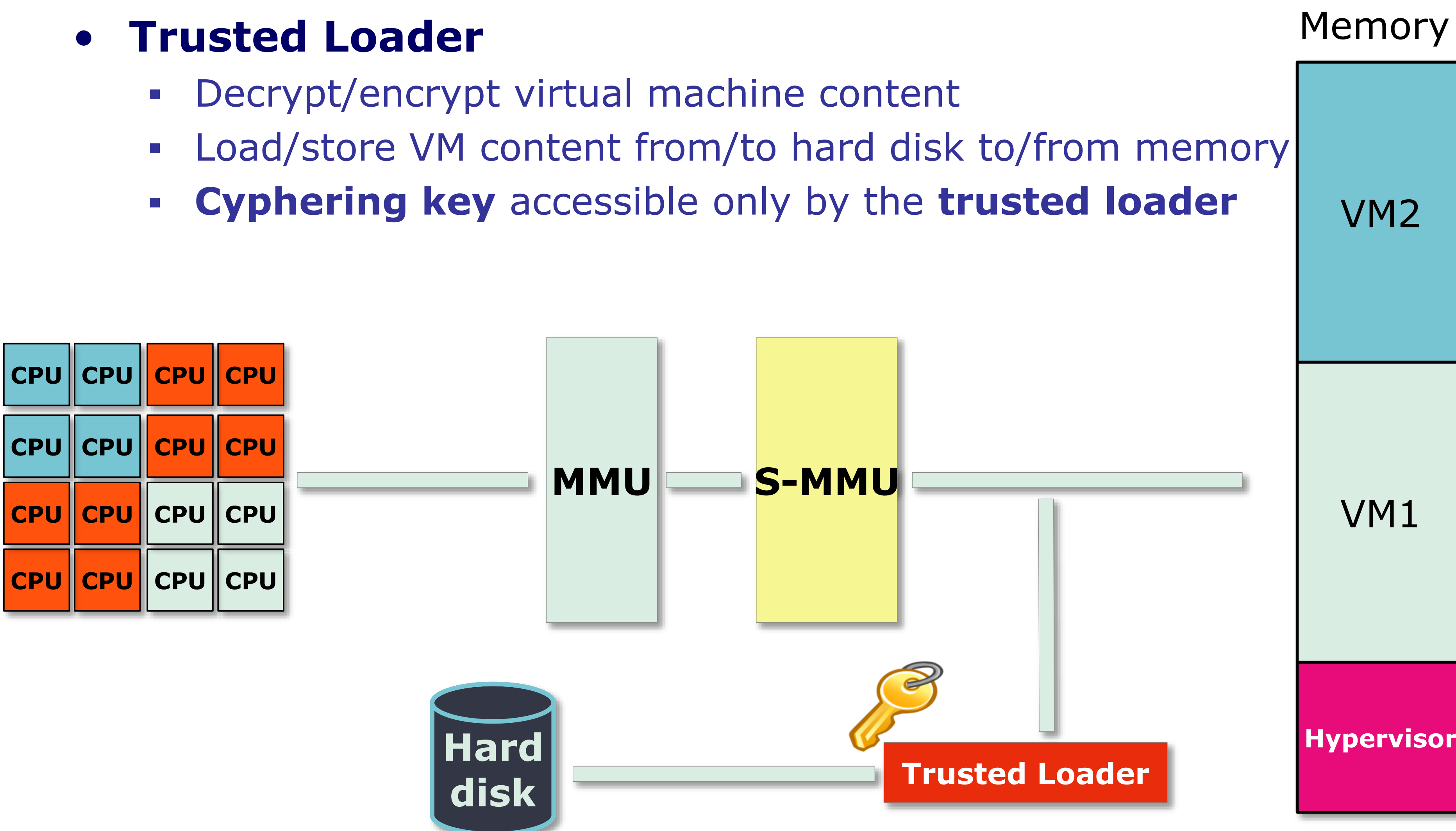
Hardware Extension

- **Secure Memory Management Unit**
  - **Creates** and **isolates** memory partitions for virtual machines and hypervisor
  - **Prevent** hypervisor from accessing virtual machines memory partitions component that should be **trusted**



Hardware Extension

- **Trusted Loader**
  - Decrypt/encrypt virtual machine content
  - Load/store VM content from/to hard disk to/from memory
  - **Cyphering key** accessible only by the **trusted loader**



Blind Hypervisor Functionality

- Hypervisor configures **Secure-MMU**
  - After **activation of secure-MMU**, hypervisor can not access VMs memory partitions
- Hypervisor commands **Trusted Loader**
  - **Decryption** or **Encryption** of virtual machine content **takes places only during** virtual machine **load** or **store**
- Hypervisor starts **virtual machines**
  - **Active** virtual machine accesses only its memory partition

Blind Hypervisor features

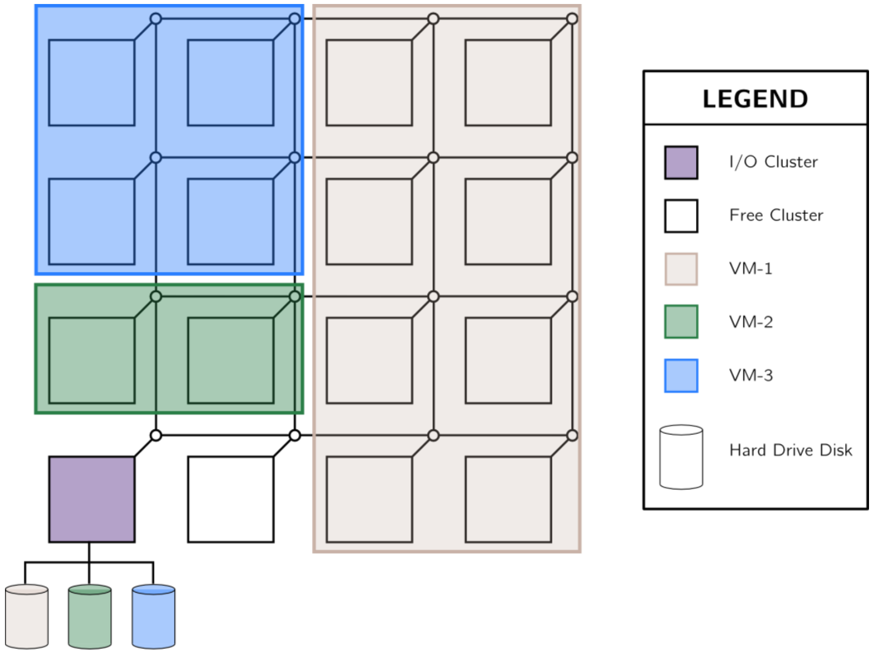
- Hypervisor not **trusted**
- Able to guarantee security properties such as **confidentiality** and **integrity** of virtual machines
- Do not **impact performance**
  - **Code** and **Data** of active virtual machine are loaded in **clear text** into memory
  - No **on-the-fly encryption** of code and data required

- Implementation on the **TSAR** manycore architecture
  - Please see details in the next presentation

Part2: Executing Secured Virtual Machines within a Manycore Architecture

Clément Devigne

Objectives

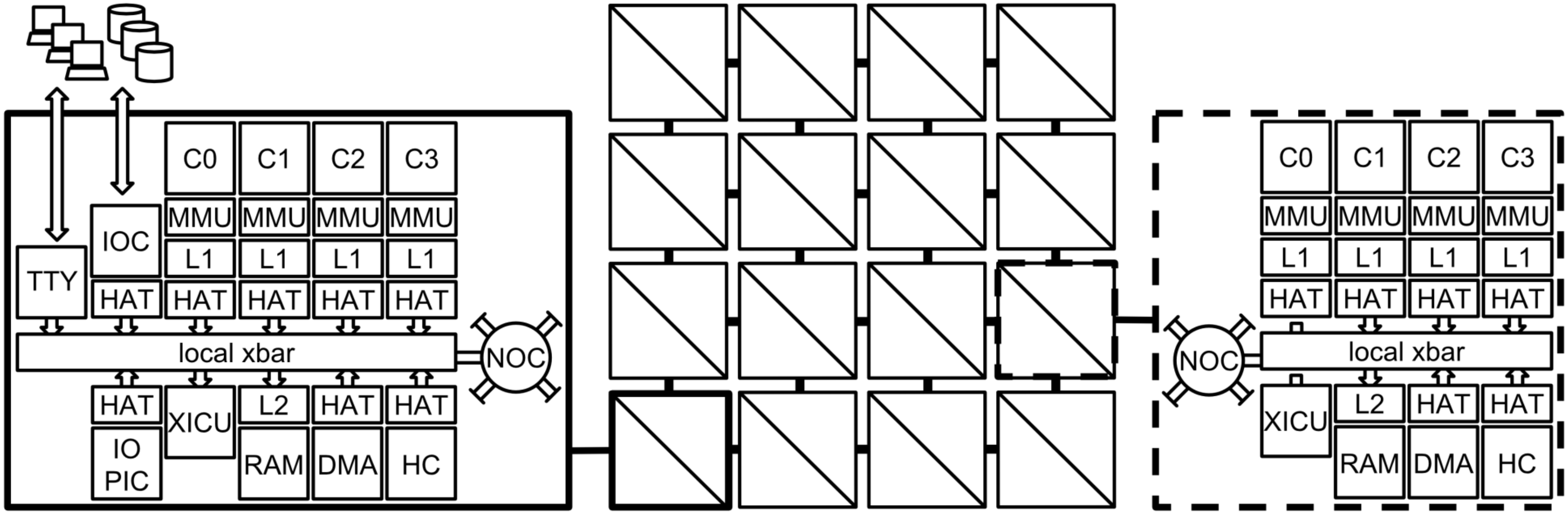


- ***Example of a TSUNAMY platform with 3 virtual machines deployed***
- **Each virtual machine has an exclusive IOC channel.**
- **Each disk contains a bootloader, the kernel code and user applications.**

Context

- The hypervisor manages all the Virtual Machines (VM)
- The hypervisor is blind (i.e. it is not able to access VM resources after their configuration)
- VMs do not share any core or memory bank
- Three address spaces: virtual, physical and machine

TSUNAMY Architecture



- **All clusters contain:**
  - 4 MIPS cores with their first level caches
  - 1 second level (L2) cache
  - 2 internal peripherals: an interrupt controller including timer functions (XICU) and a DMA controller
  - A local crossbar
  - The Hardware Address Translator (HAT)
- **The I/O cluster additionally contains:**
  - A terminal controller (TTY)
  - A hard-drive disk controller (IOC)
  - A Programmable Interrupt Controller (PIC)

Memory Management Unit

- **A MMU generally uses a translation cache (called TLB) to speed up address translation**
  - Non negligible hardware overhead, including the logic to manage the TLB misses
  - Slower to perform address translation because of the TLB misses overhead

- **The hypervisor must create the page table for the memory allocated to a virtual machine and store it into a memory space non accessible by itself nor any virtual machine**

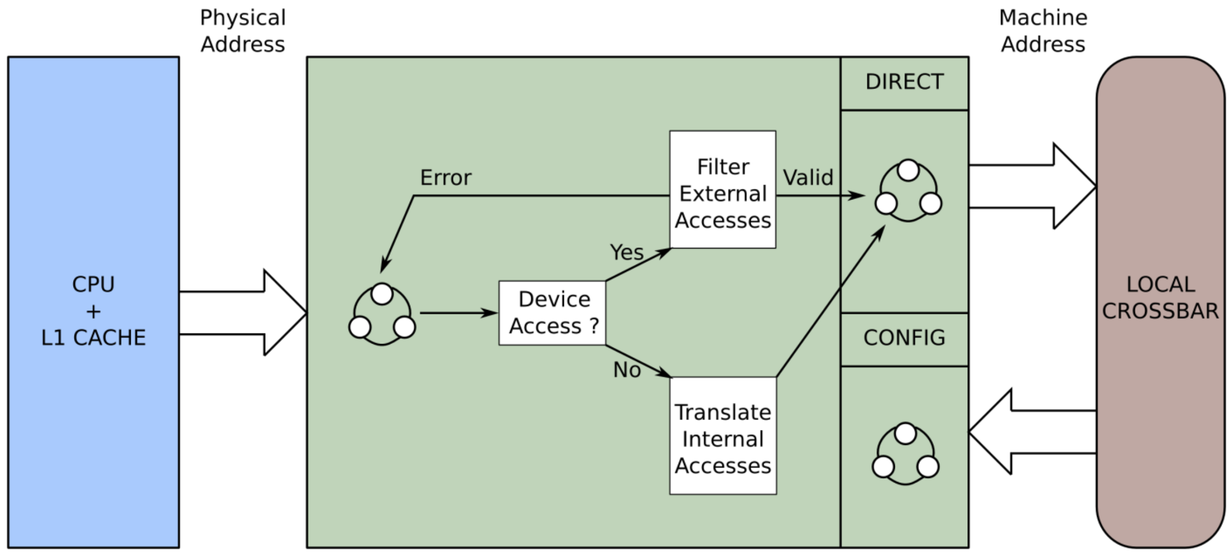
- **Translation with a page granularity (e.g. 4KB)**
  - Useful when virtual machines share memory banks
  - but this is not within our hypothesis to physically isolate the virtual machines

Hardware Address Translator

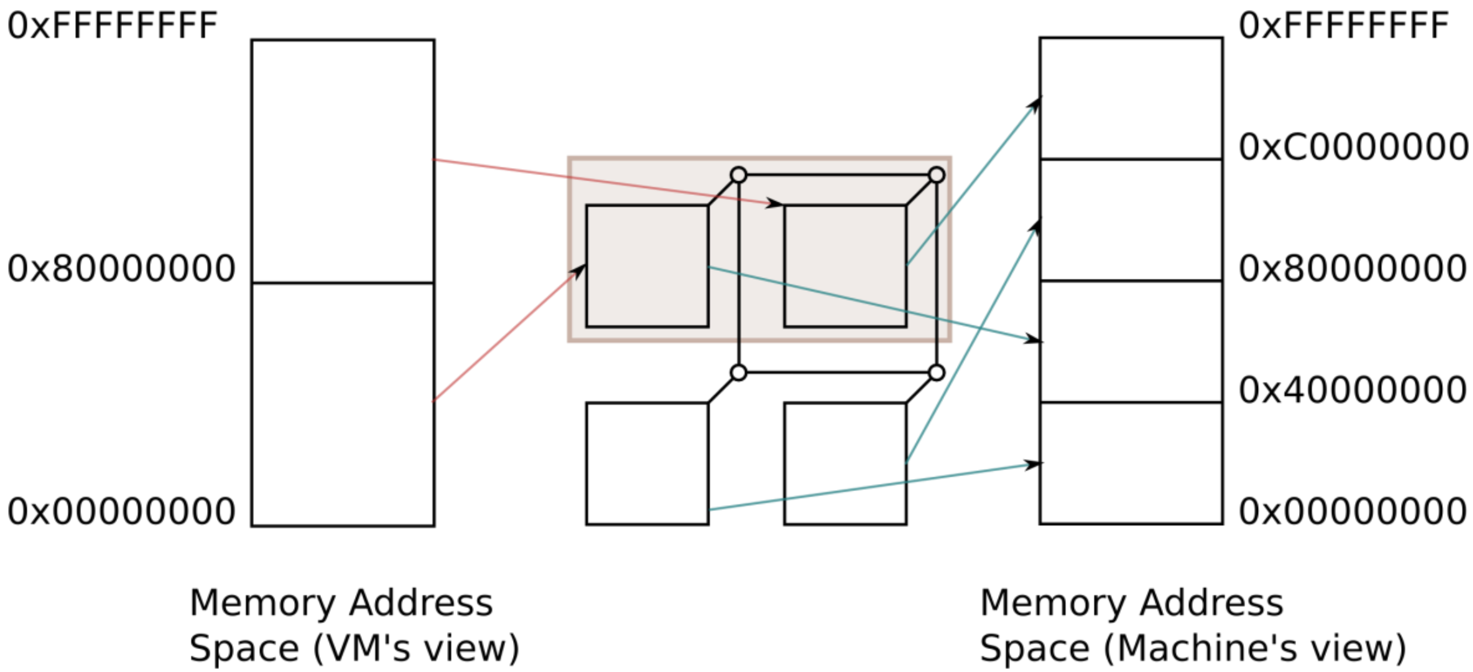
- **HAT performs the translation from physical addresses to machine addresses**
- **Configured once by the hypervisor at the start of an operating system and placed behind each initiator in the architecture**
- **HAT only needs topology information to perform address translation**
- **HAT operates with a coarser granularity (cluster granularity)**

HAT: Overview

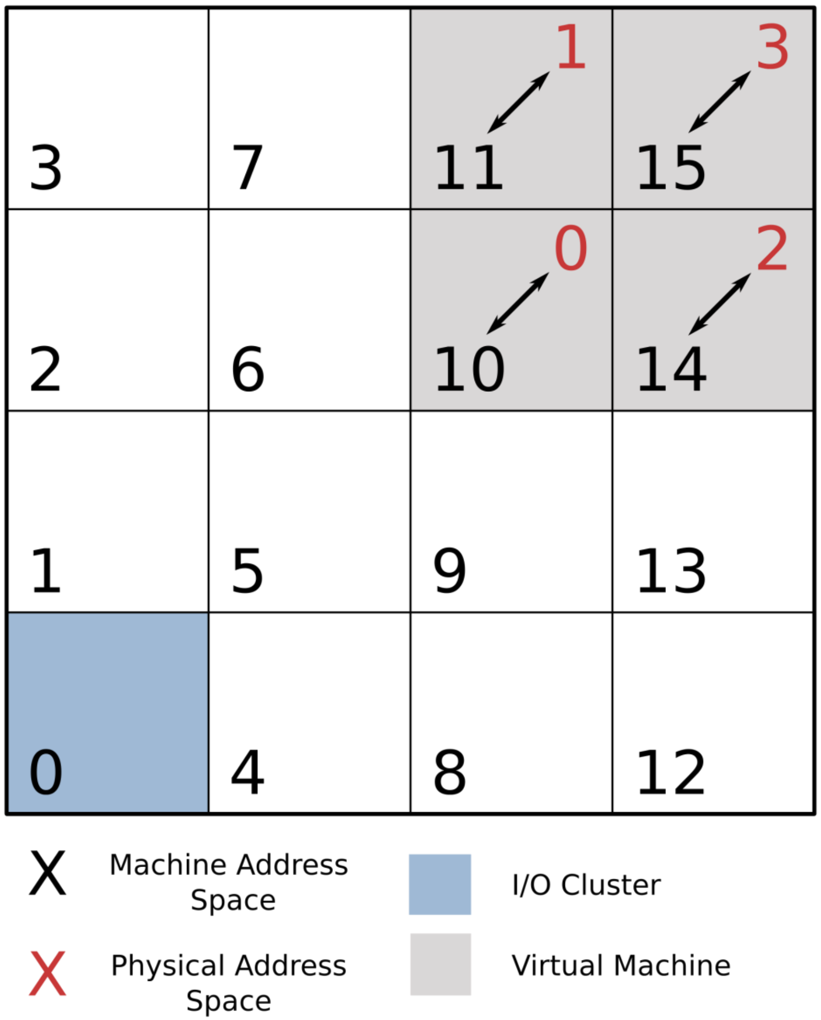
- **Two types of addresses target**
  - module included in a cluster of the same virtual machine (*internal access*)
  - peripheral outside the virtual machine (*external access*)



Memory Space Distribution

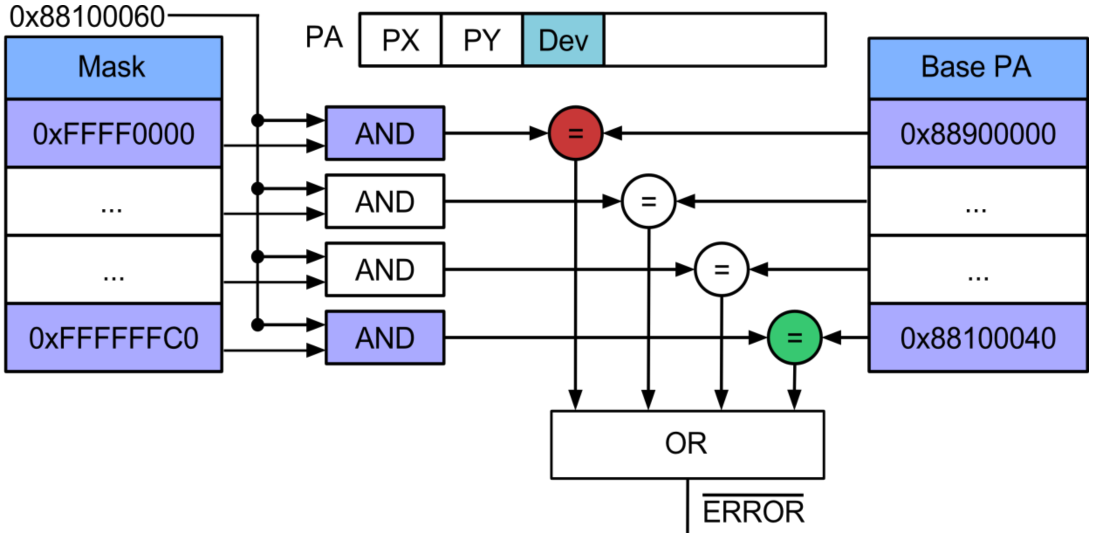


HAT: Internal Accesses Mechanism



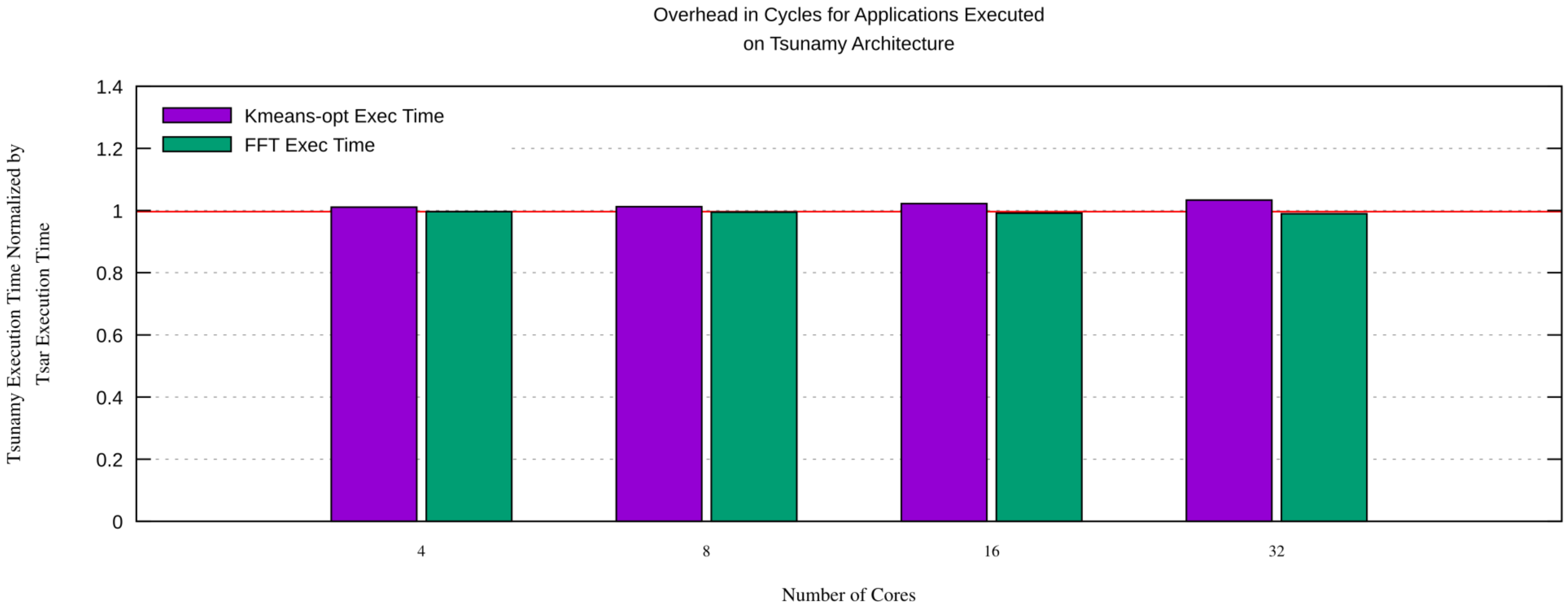
- **Most significant bits (MSB) define the cluster coordinates (X; Y)**
- **The address translation consists only in changing the MSB**

HAT: External Accesses Mechanism



- **1 bit defines if the request targets a peripheral (*DEV* bit)**
- **2 tables into the HAT handle peripheral accesses**
  - Base Physical Address table
  - Mask table

Preliminary results



- **Average overhead : < 3%**

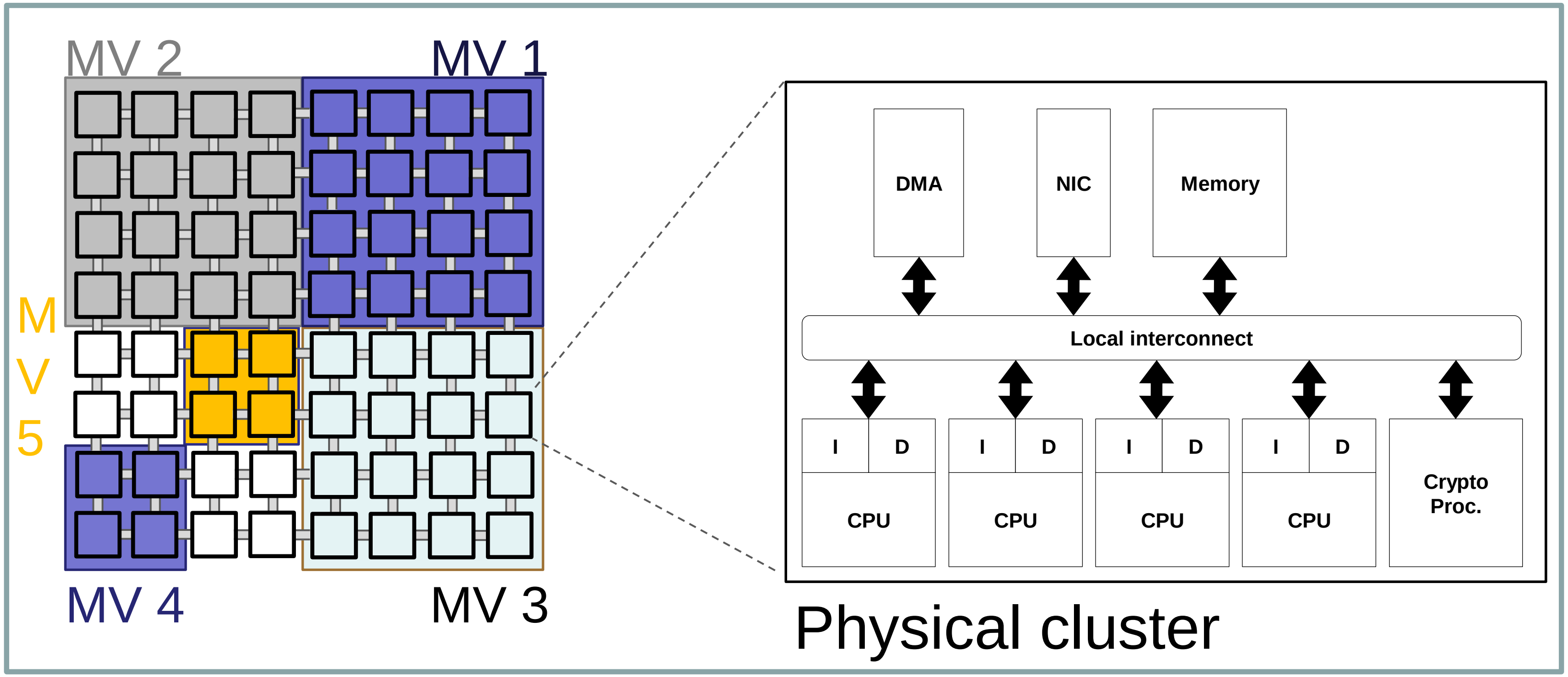
Part3: Secure application deployment

Maria Mendez

Global overview

Trusted Virtual Machines (VM)

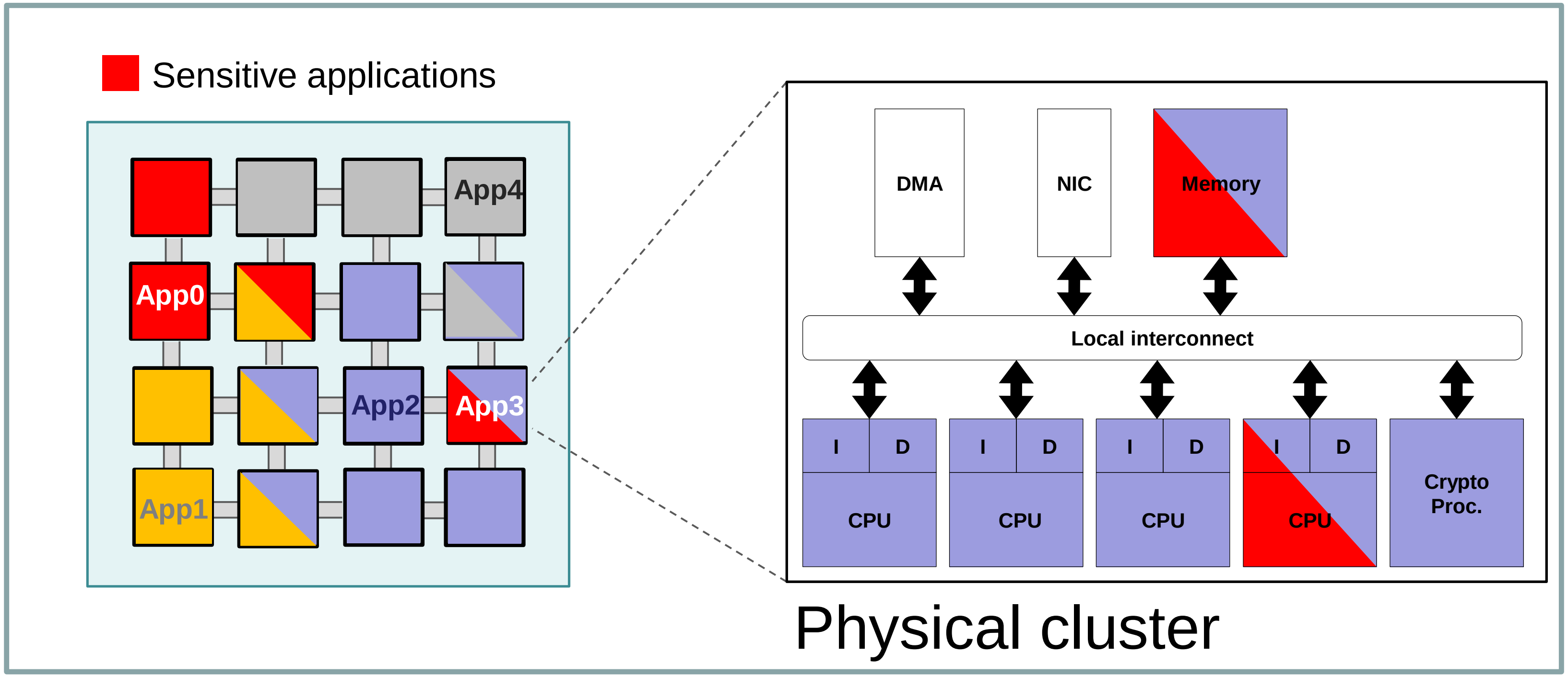
TSAR Architecture



Inside a trusted virtual machine

Are the applications securely deployed?

TSAR Architecture

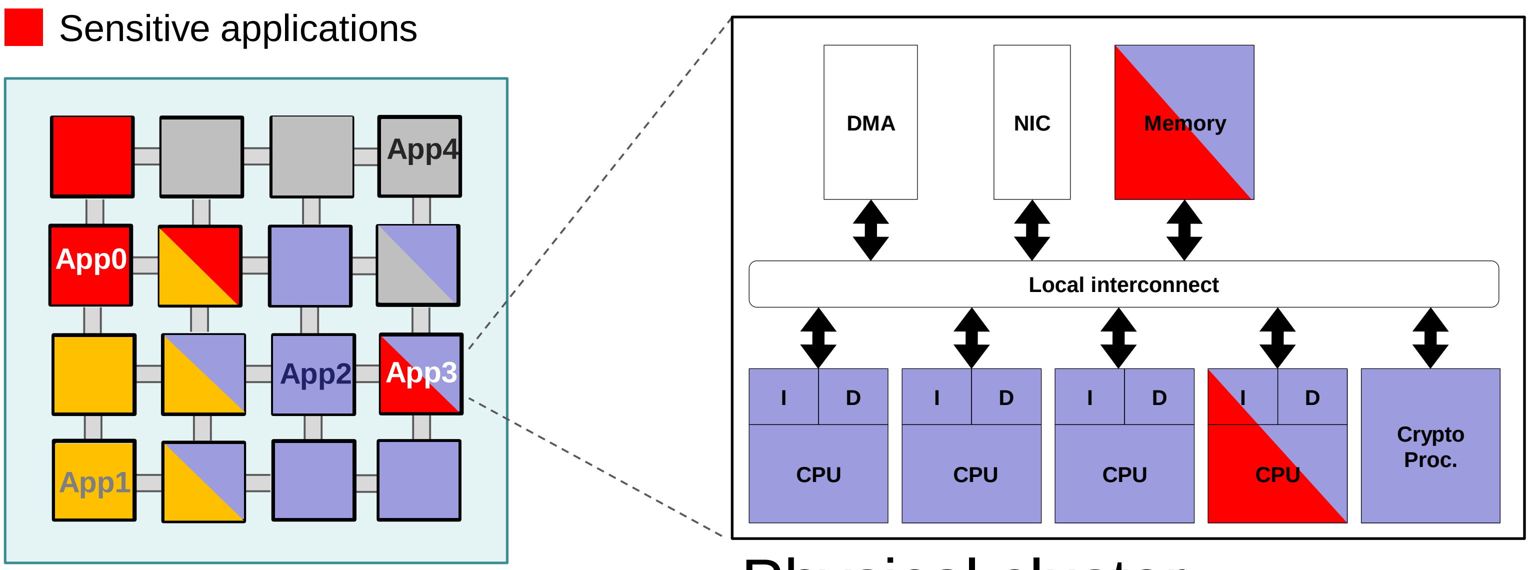


➡ Sharing resources

Threat model

Sharing resources

- Denial of Services attacks (DoS)
- Leakage of Information attacks (Communication and Cache Side-Channel Attacks (SCA))

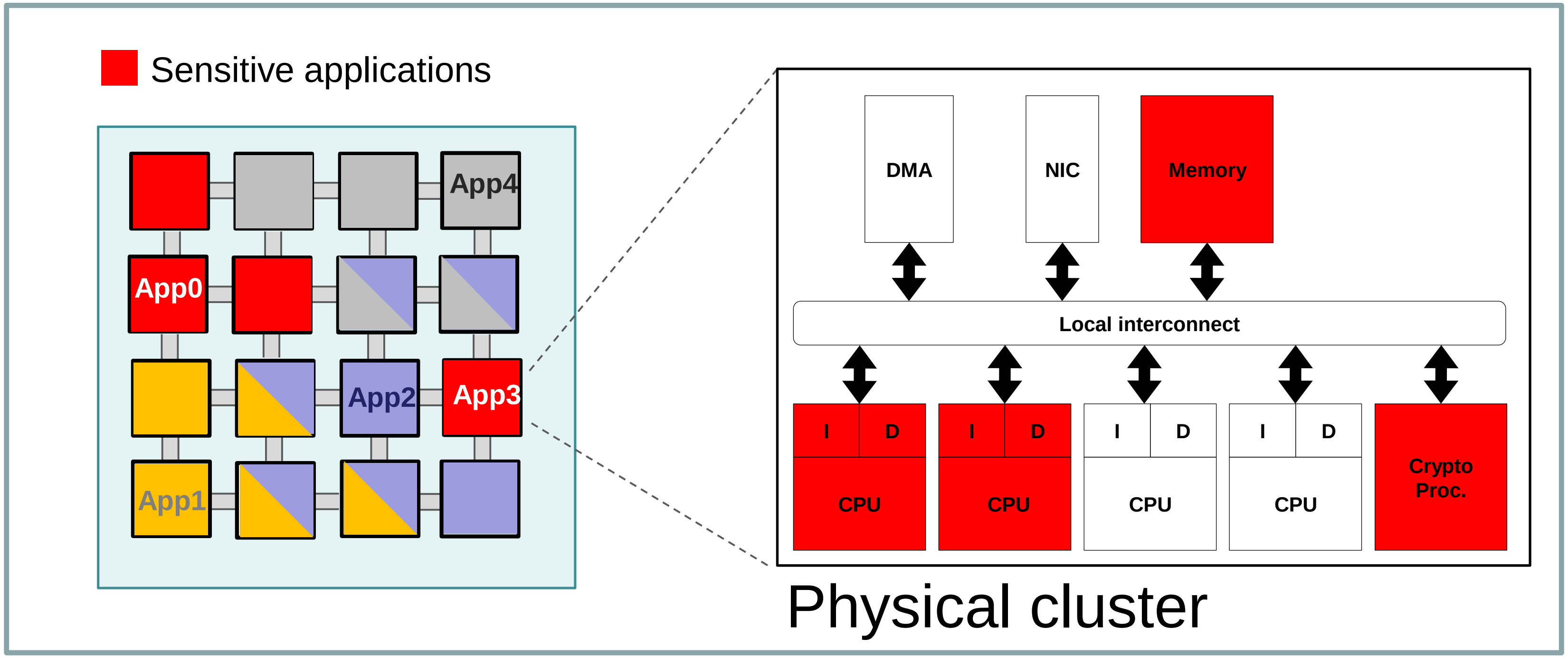


Physical cluster

J. Demme and S. Sethumadhavan. Side-channel vulnerability metrics: Svf vs . csv. In WDDD, 2014
Y. W. and G. E. Suh. Efficient timing channel protection for onchip networks. In NOCS 12 Proceedings of the 2012 IEEE/ACM Sixth International Symposium on Networks-on-Chip, pages 142–151, 2012
M. J. Sepulveda et al. Protection for SoC Time-Driven Attacks," /Embedded Systems Letters, IEEE/ , vol.7, no.1, pp.7,10, March 2015

Physical isolation

Objective: Physically isolate sensitive applications in order to avoid Cache SCA and DoS attacks



Physical isolation implementation

ALMOS services extension

Scheduling

Monitoring

New user application mapping

Task (thread, fork) mapping

Memory allocation (level 2 cache)

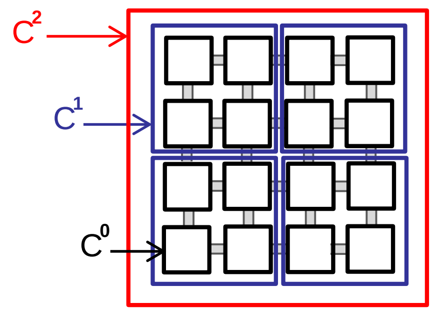
ALMOS. https://www-soc.lip6.fr/trac/almos

Physical isolation implementation

ALMOS services extension

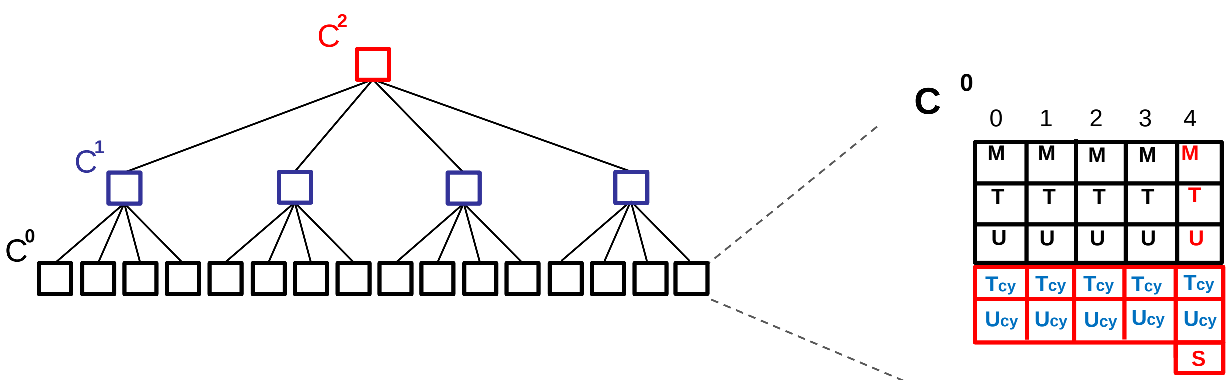
- Scheduling
- **Monitoring**
- New user application mapping
- Task (thread, fork) mapping
- Memory allocation (level 2 cache)

Distributed Quaternary Decision Tree (DQDT) adapted for crypto-processors and secure zones



M : Physical pages number
T : Threads number (Runnable)
U : CPU utilization

Tcy : Crypto tasks number
Ucy : Crypto-processor utilization
S : Secure zone ID



ALMOS. https://www-soc.lip6.fr/trac/almos

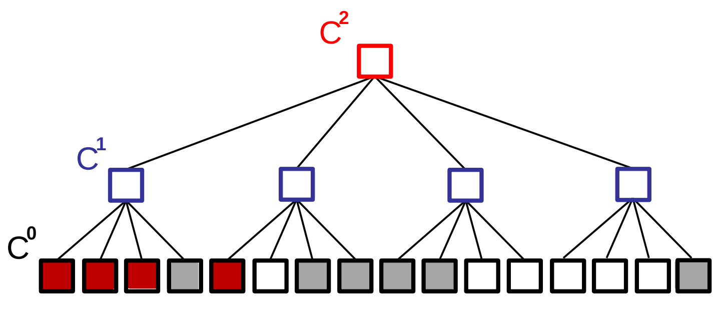
Physical isolation implementation

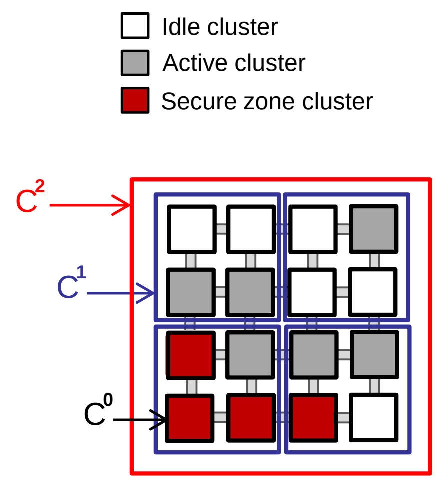
ALMOS services extension

New secure zone creation service

- Scheduling
- **Monitoring**
- **New user application mapping**
- Task (thread, fork) mapping
- Memory allocation (level 2 cache)

Knowing the maximum parallelism of an application → Searching for contiguous physical resources enough → Statically creation of a secure zone





ALMOS. https://www-soc.lip6.fr/trac/almos

Physical isolation implementation

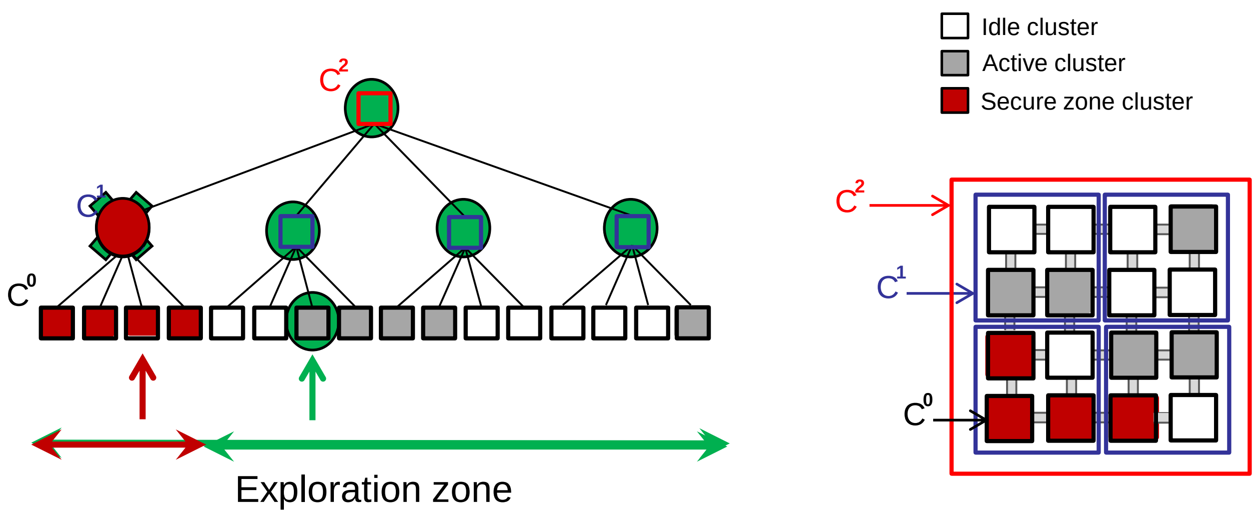
ALMOS services extension

With the mechanism of secure zones, the exploration zone, for physical as well as for logical isolated application is reduced and bounded by the entire platform

- Scheduling
- Monitoring
- New user application mapping
- Task (thread, fork) mapping
- Memory allocation (level 2 cache)

After physical isolation mechanisms

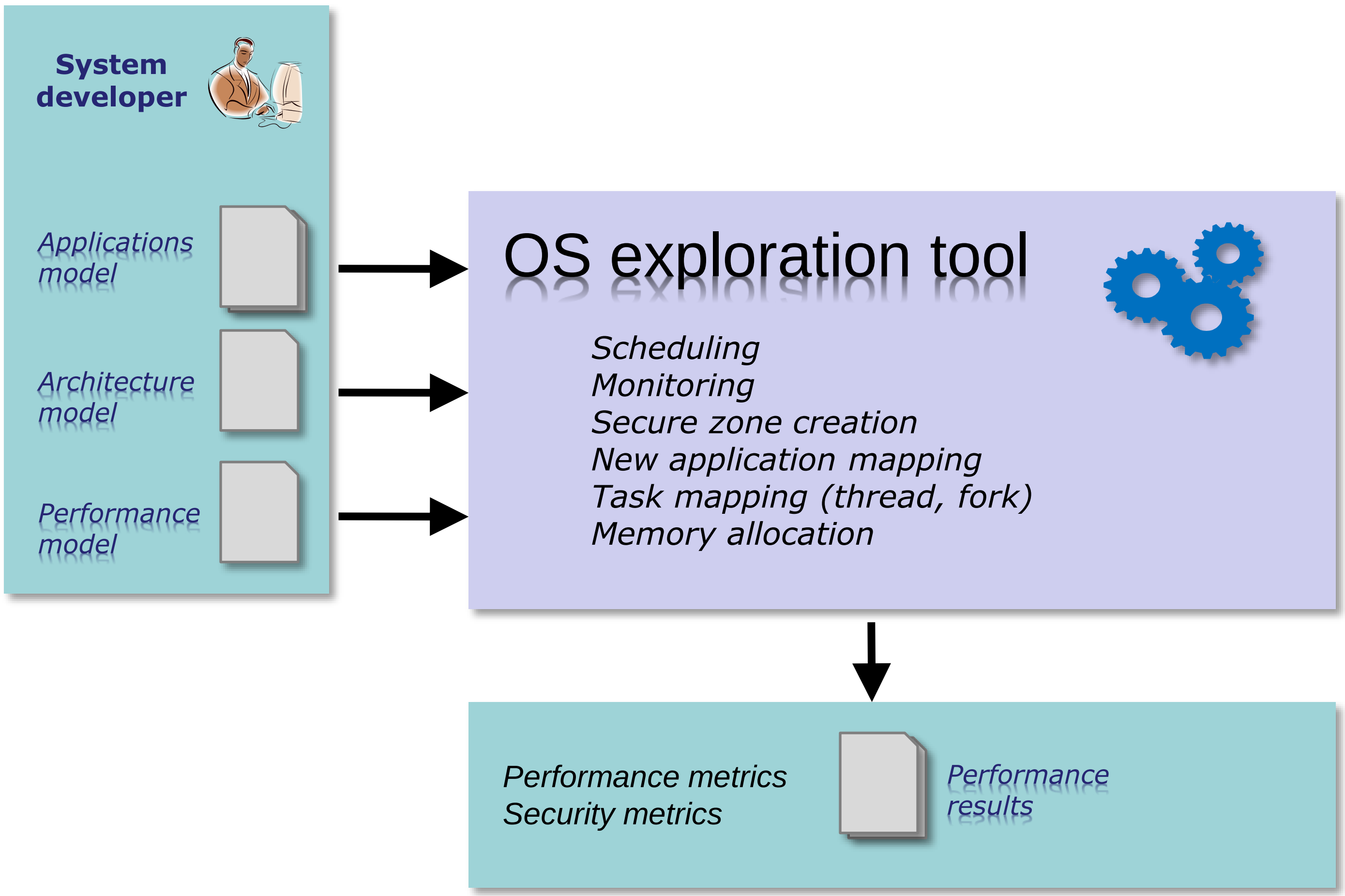
Before physical isolation mechanisms



ALMOS. https://www-soc.lip6.fr/trac/almos

OS exploration tool

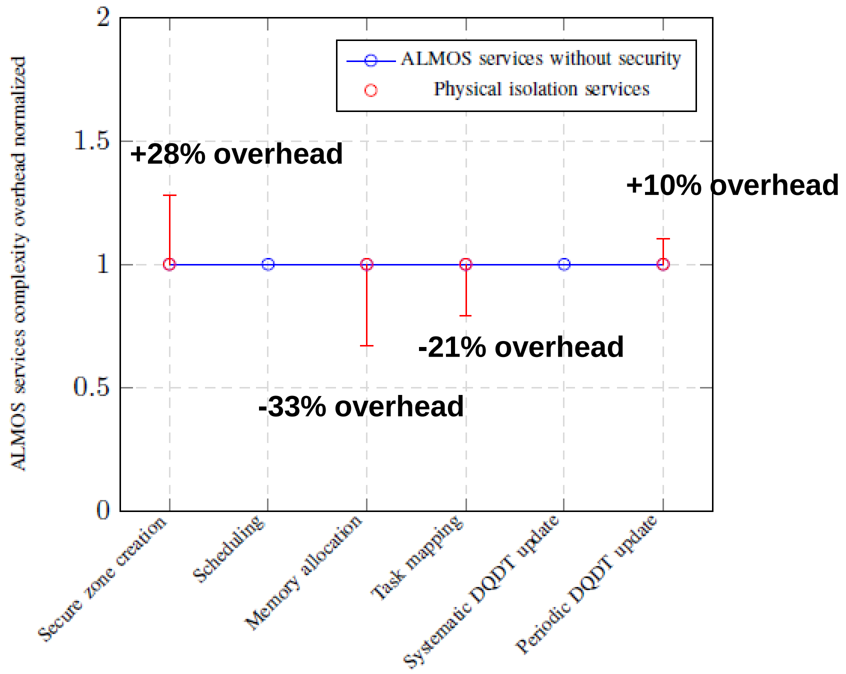
- **Models based exploration tool**



Experimental results

- **Experimental protocol**
  - Synthetic applications
  - Evaluation on 2x2 and 4x4 physical clusters architectures, each physical cluster containing 4 CPUs

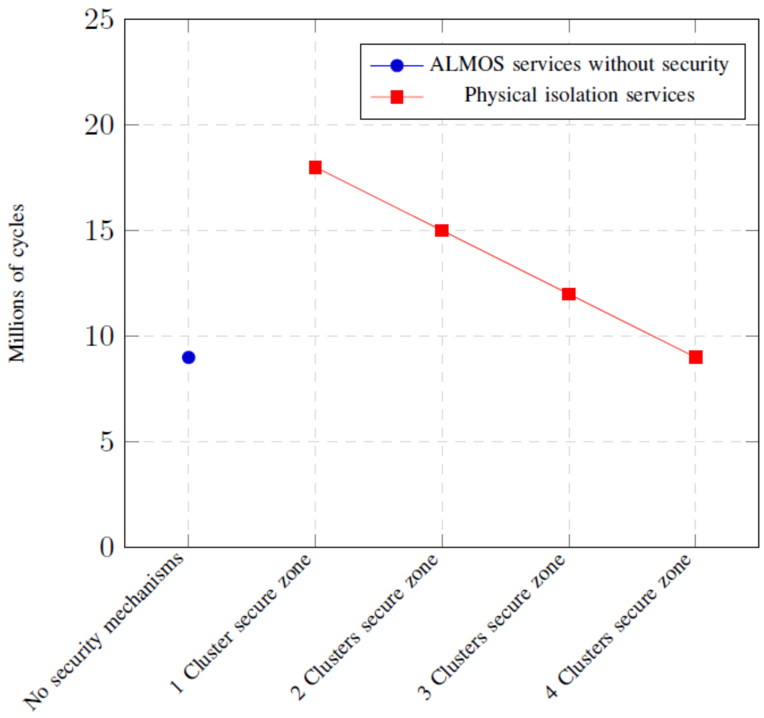
1. Performance overhead evaluation of atomic ALMOS extended services normalized by original ALMOS services



Experimental results

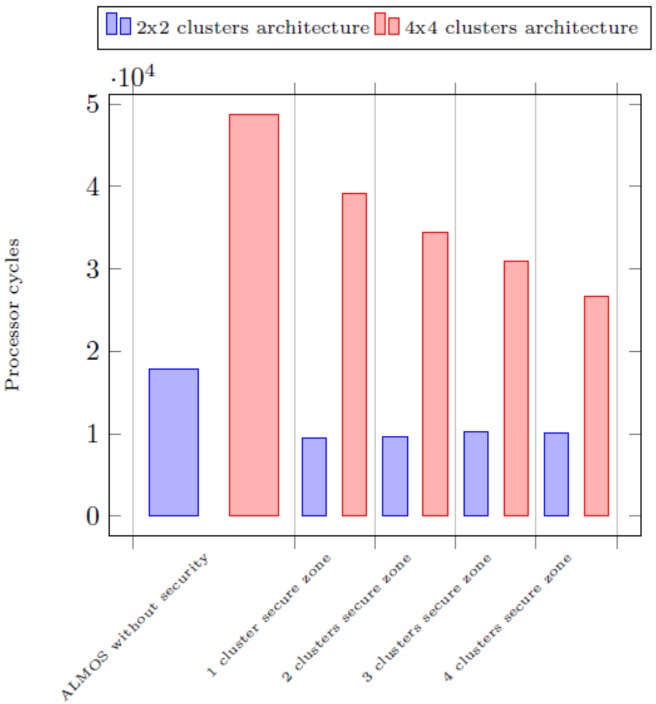
2. Comparison between original and security enhanced ALMOS services with no architecture load.

A. Performance (total execution time) of an application intended to be physically isolated

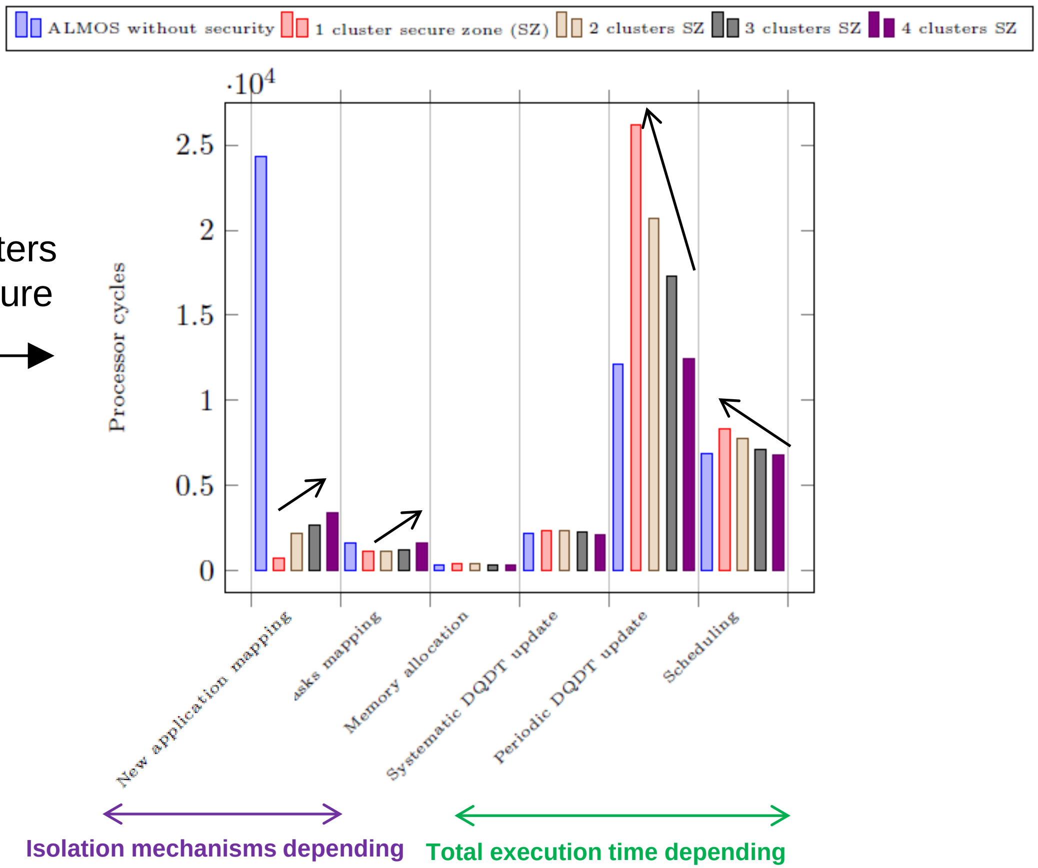


Experimental results

B. Performance of the security enhanced ALMOS services



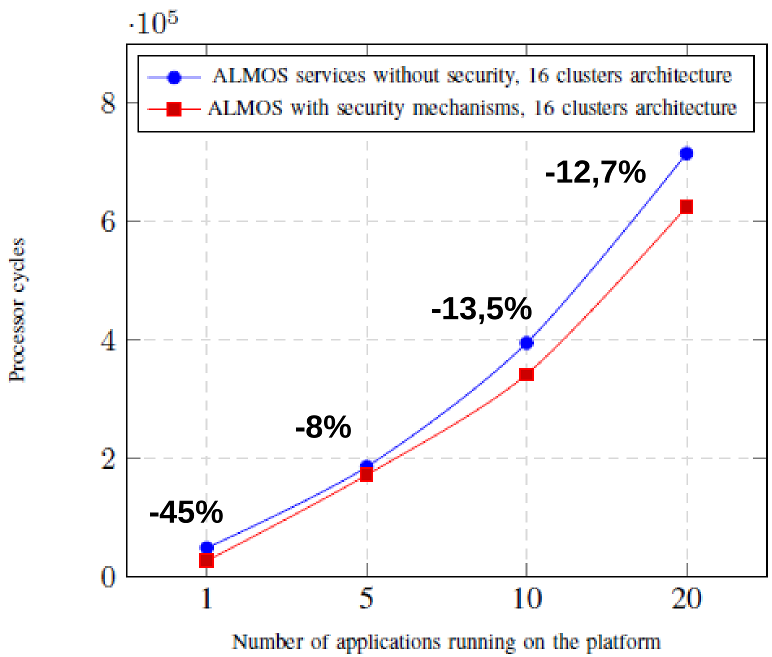
4x4 clusters architecture



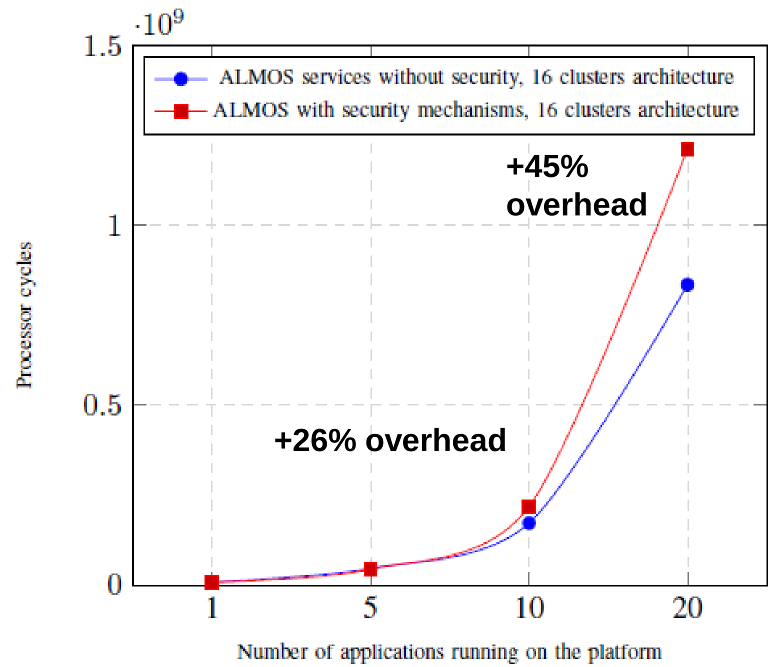
Experimental results

3. Comparison between original and security enhanced ALMOS services according to the number of applications running on the platform (one single application isolated on a 4x4 clusters architecture)

A. ALMOS services performance



B. Execution time of non isolated applications



Conclusion

- **The TSUNAMY project addresses the problem of secure handling of personal data and privacy in manycore architectures**
- **It proposes a solution to execute many independent applications in parallel, safely and ensuring respect for the privacy of users**
- **It proposes mechanisms for logical and physical isolation to ensure execution of partitioned applications**
- **It develops strategies for dynamically distributing applications on a manycore architecture**

Applications security in manycore platform: from operating system to hypervisor

Mehdi Aichouch[1], Clément Devigne[2], Guy Gogniat[3], Maria Mendez[3]

[1]CEA, Saclay, [2]LIP6, Jussieu, [3]Lab-STICC, Lorient

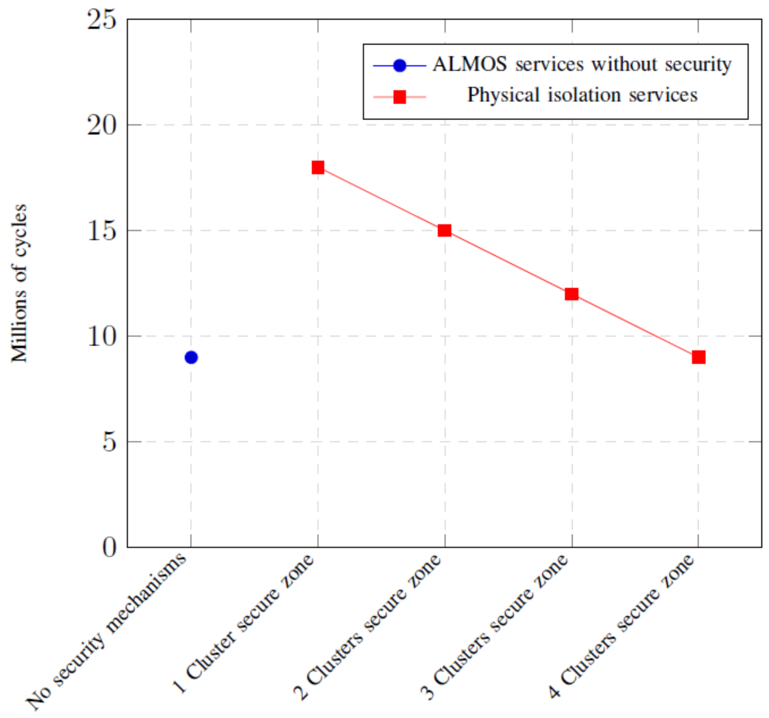
TSUNAMY Project (ANR)

CryptArchi'2015, Leuven, Belgium - June 29th, 2015

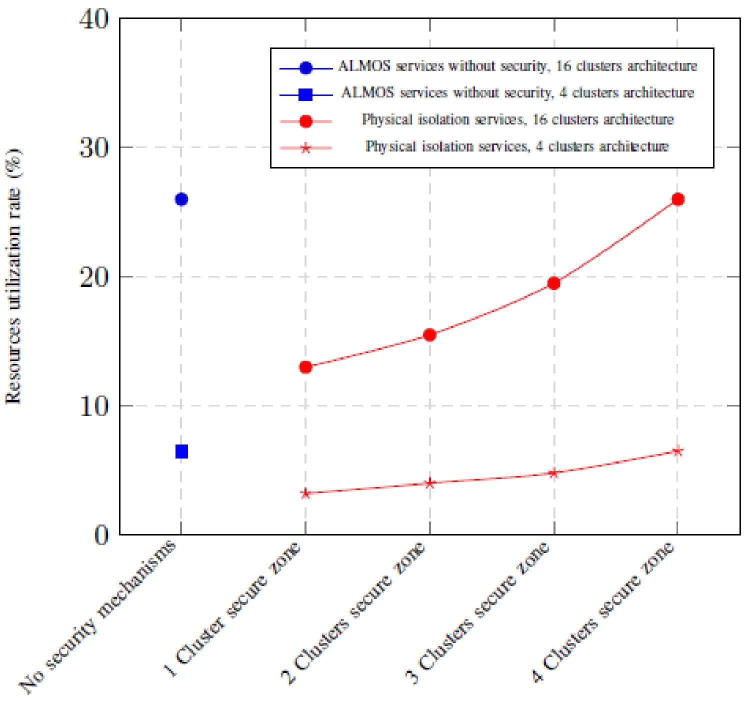
Experimental results

2. Comparison between original and security enhanced ALMOS services with no architecture load.

A. Performance (total execution time) of an application intended to be physically isolated



B. Resources utilization rate according to isolated scenarios



Experimental results

4. Comparison between original and security enhanced ALMOS services according to the number of applications physically isolated, 20 applications running on the 4x4 clusters platform