Déploiement sécurisé d'applications au sein des architectures many-coeurs

Université Bretagne Sud

Lab-STICC

Vianney LAPOTRE

TSUNAMY

- **https://www.tsunamy.fr**
- **From December 2013 to May 2017**

- **Partners**

Lilan BOSSUET
Cuauhtemoc MANCILLAS

Maria MENDEZ
Guy GOGNIAT
Vianney LAPOTRE

Franck WAJSBÜRT
Quentin MEUNIER
Clément DEVIGNE

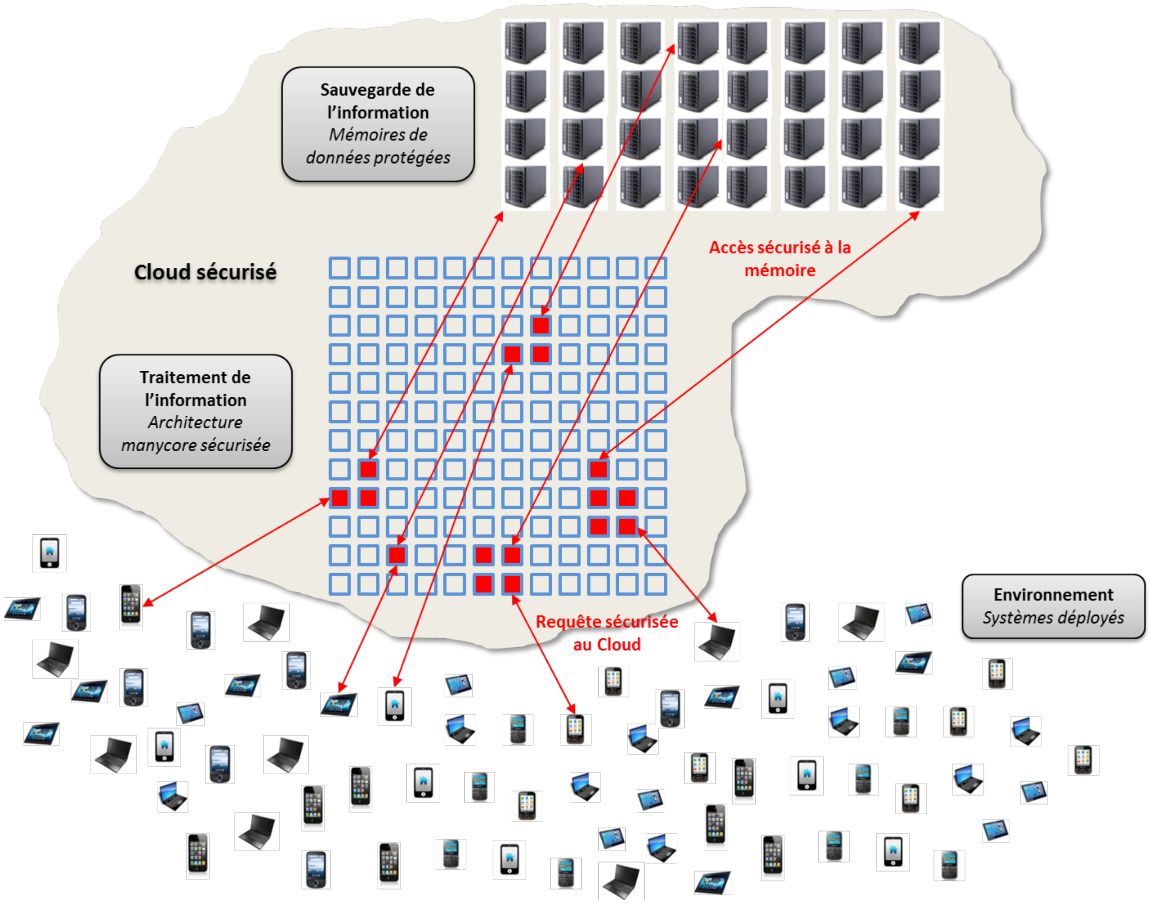
Moha AIT HMID

TSUNAMY – Toward a trusted platform

- ***Cloud computing context***
  - Need for secure requests
  - Secure storage

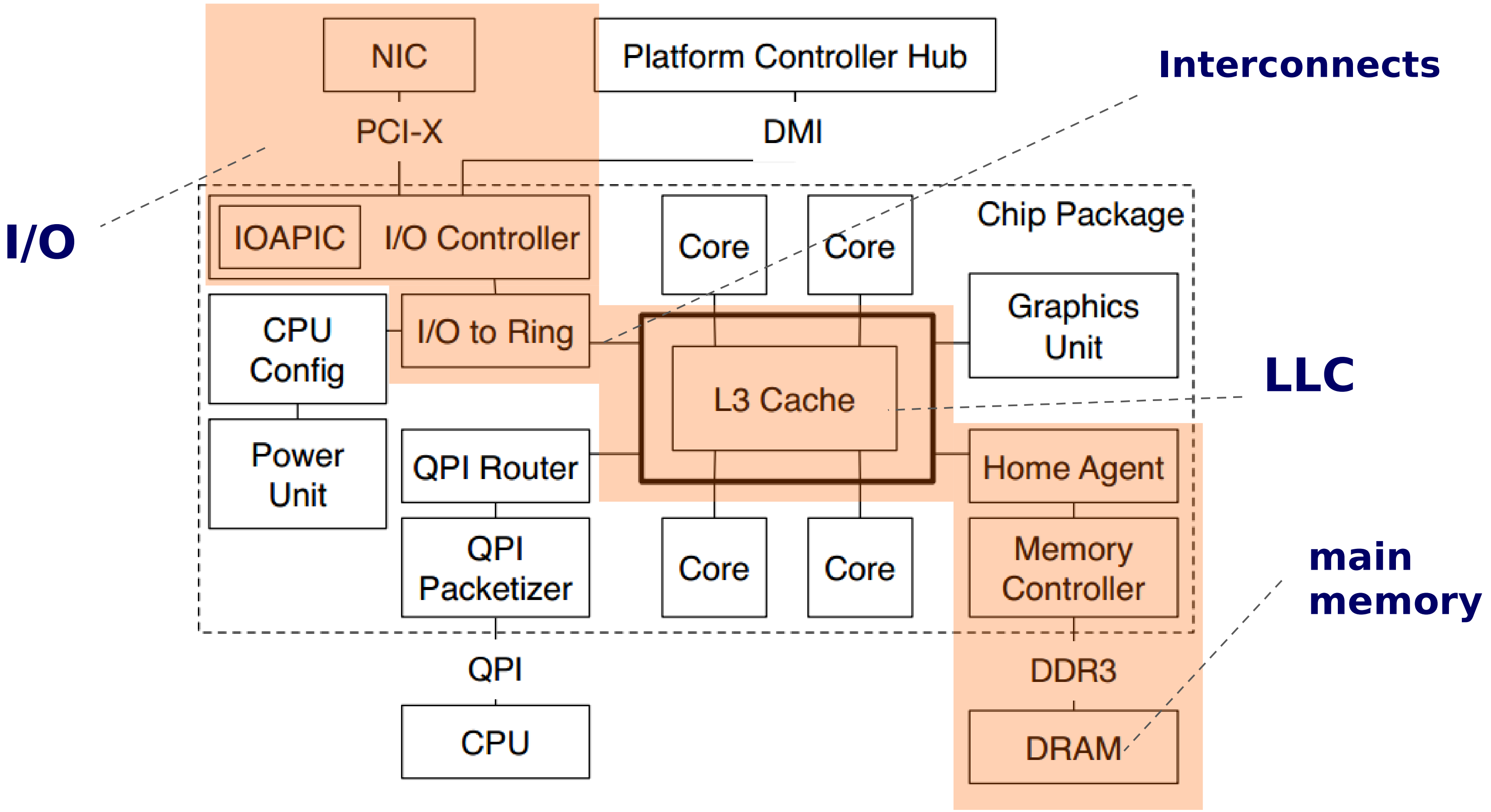
- **Challenges for security and performance**

  1. TSAR extension to integrate crypto-processor
  2. Virtual Machines isolation (i.e. Blind Hypervisor)
  3. Applications Isolation within each Virtual Machine



Why do we need (strong) isolation ?

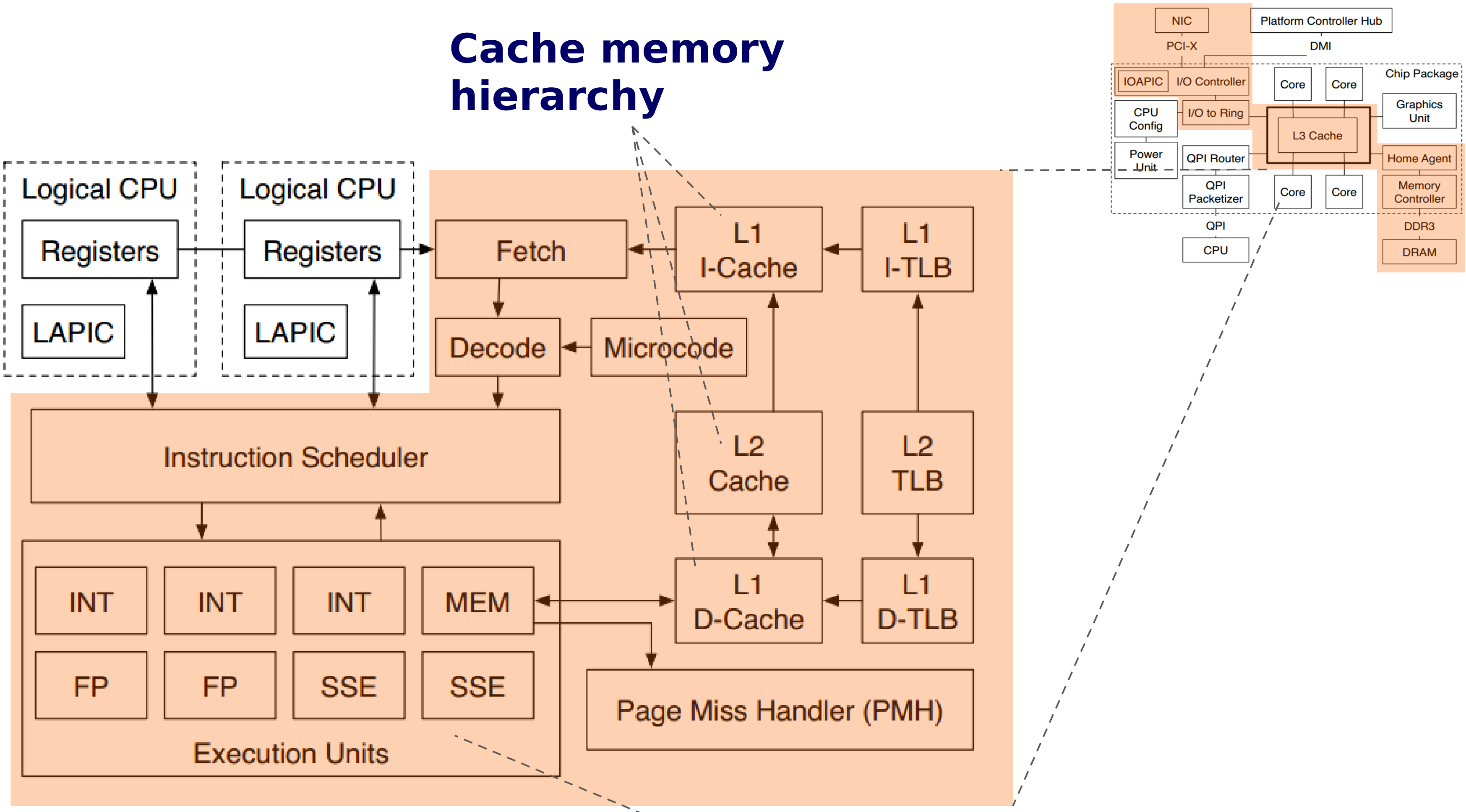
An Intel processor's die



"notable" shared resources

Why do we need (strong) isolation ?

Cache memory hierarchy

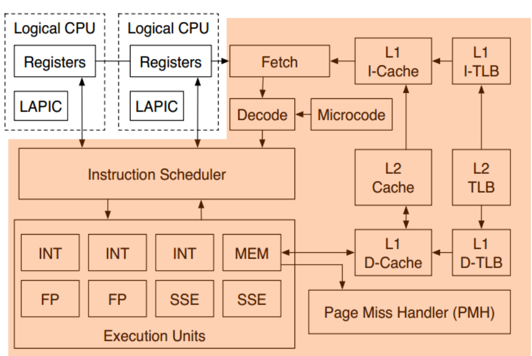


Execution units & branch predictor

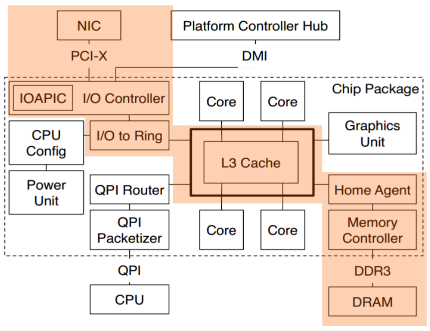
"notable" shared resources

Why do we need (strong) isolation ?

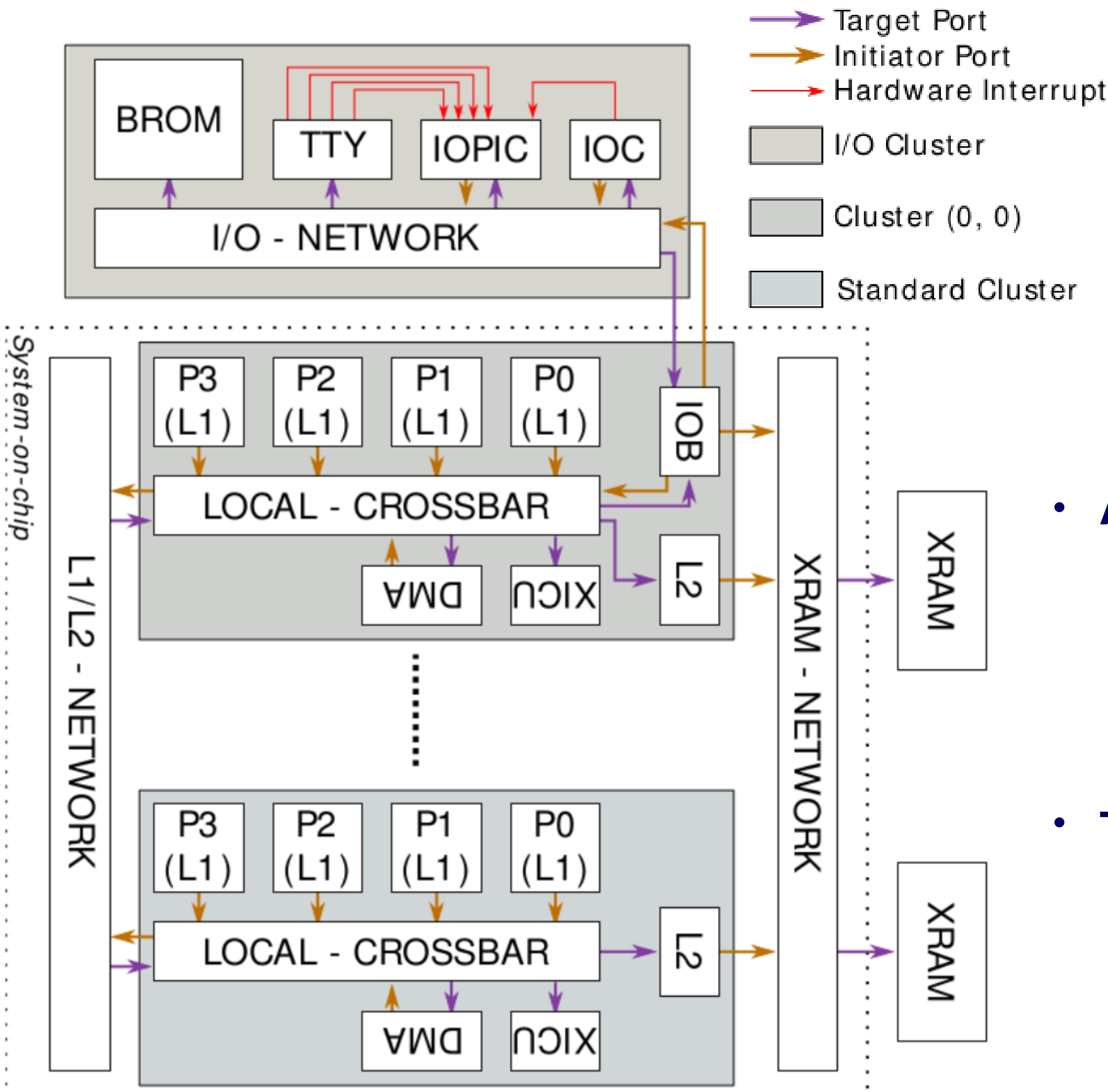
- **Numerous resources are shared in a multi- or many-core system**
  - This leads to multiple threats
    - Recently we can mention both Meltdown and Spectre vulnerabilities

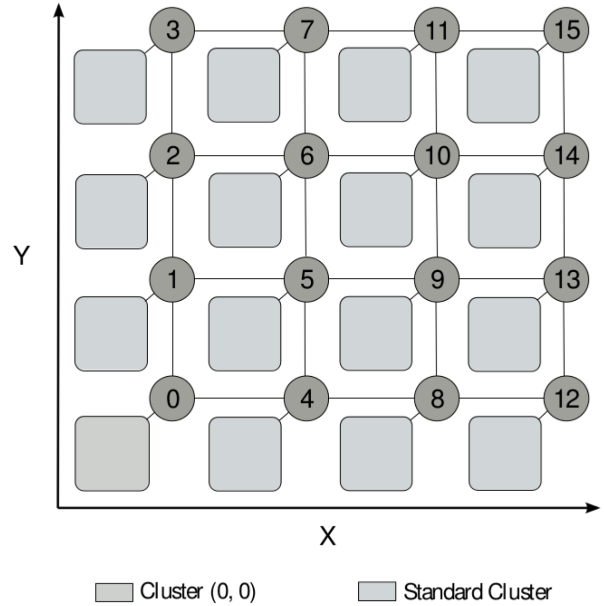


https://spectreattack.com



TSAR architecture



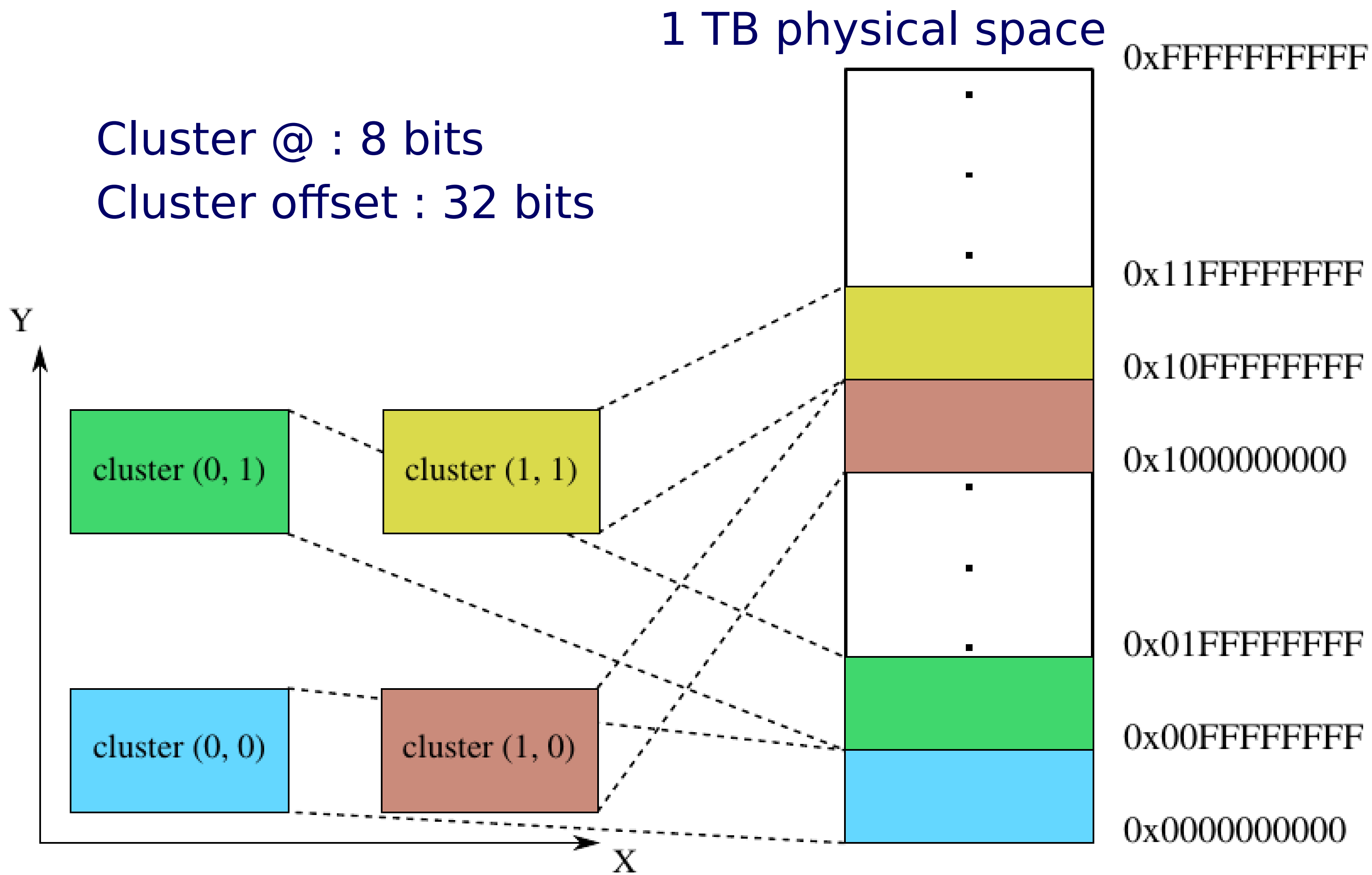


- **All clusters contain:**
  - 4 MIPS cores with their first level (L1) caches
  - 1 second level (L2) cache in charge of a segment of physical memory
  - 2 internal peripherals: XICU, DMA
  - A local crossbar
- **The I/O cluster contains:**
  - A terminal controller (M_TTY)
  - A hard-drive disk controller (M_IOC)
  - A Programmable Interrupt Controller (IOPIC)
  - A boot ROM (BROM)
  - A I/O network with access to the RAMs network

TSAR architecture

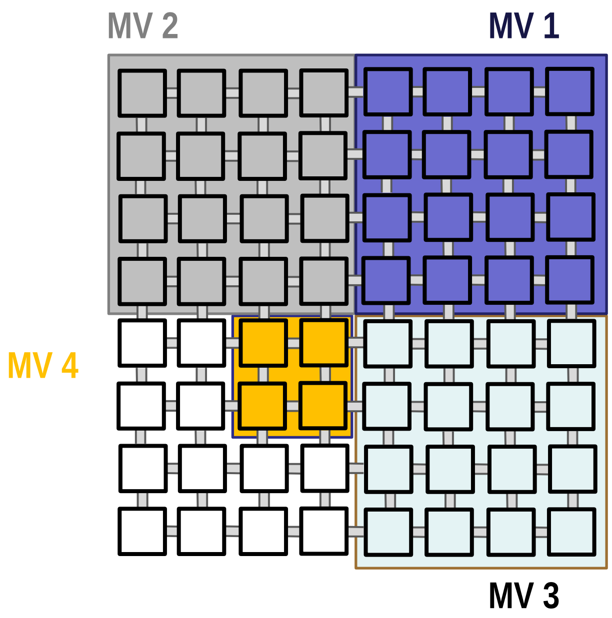
Cluster @ : 8 bits

Cluster offset : 32 bits



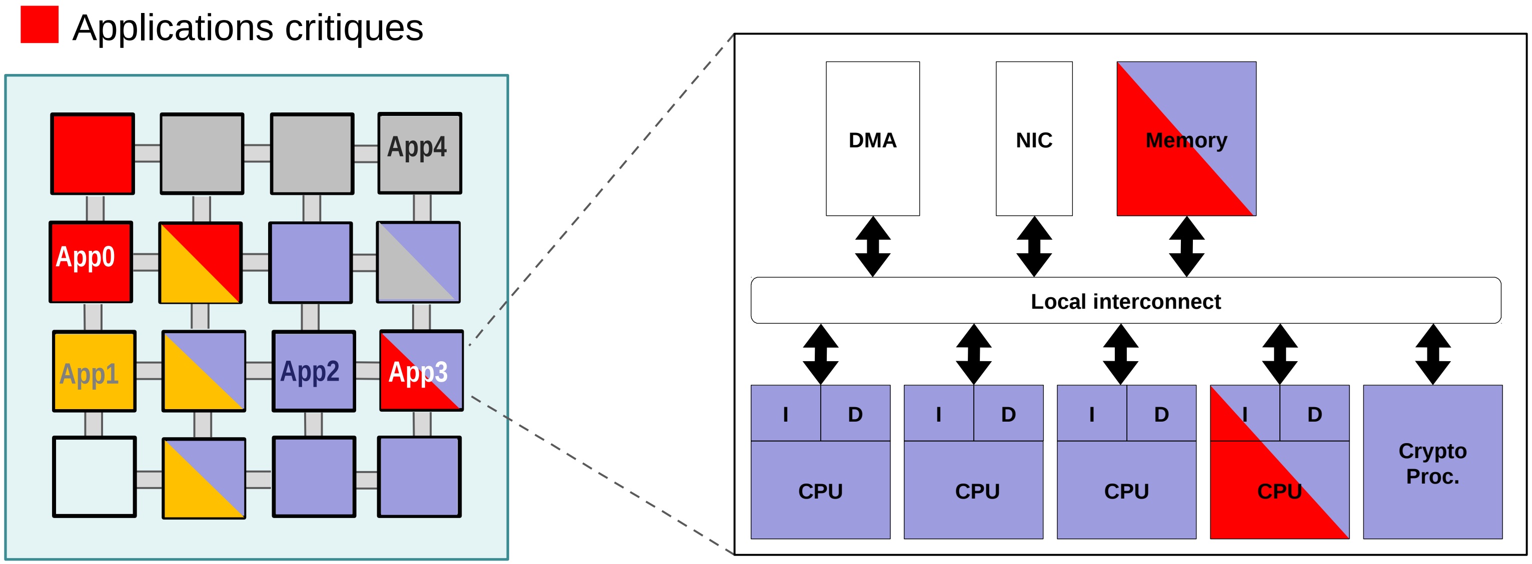
1) Virtual machines isolation

- **Blind Hypervisor**
  - Warrant Virtual Machines confidentiality and integrity
    - *Strong memory isolation between VMs.*
    - *Do not address deny of services (DoS).*
  - High level assurance
    - *Reduce root of trust (TCB) to make formal verification feasible.*
    - *Do not trust hypervisor.*
  - Protection from software attacks
    - *From both other VMs and hypervisor.*
    - *Do not address probing or other physical attacks.*
  - Low performance impact : no on-the-fly encryption
    - *VM are stored in clear text in RAM.*
    - *VM are ciphered outside SoC (eg hard drive, network).*

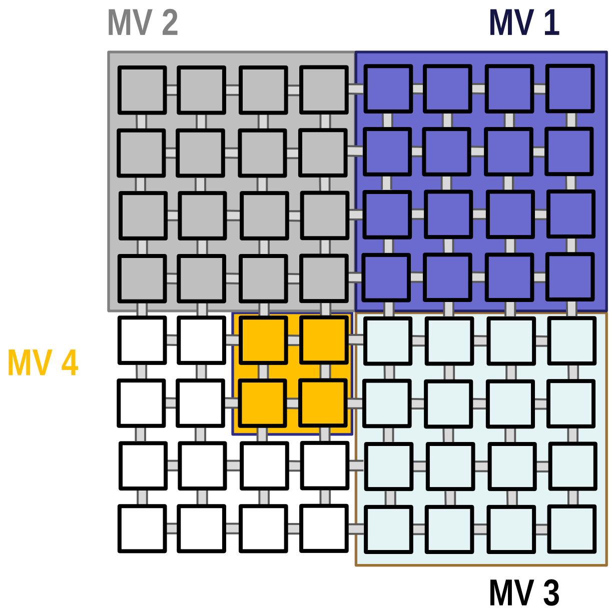


2) Application isolation within a VM

- **Sensitive applications (e.g cryptographic processes) need to be isolated from non-trusted application**

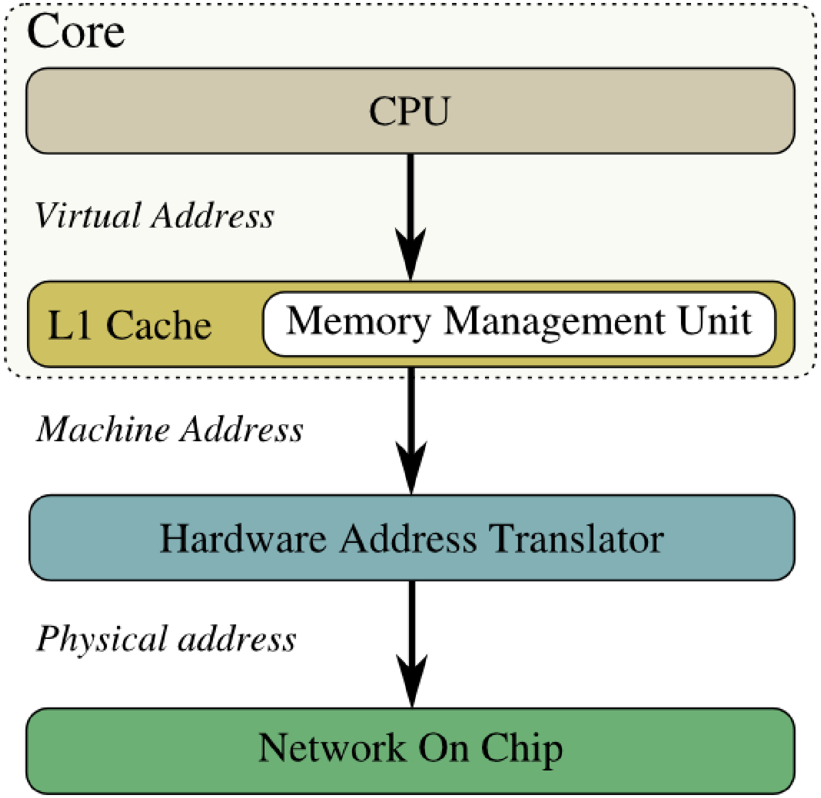


Blind hypervision

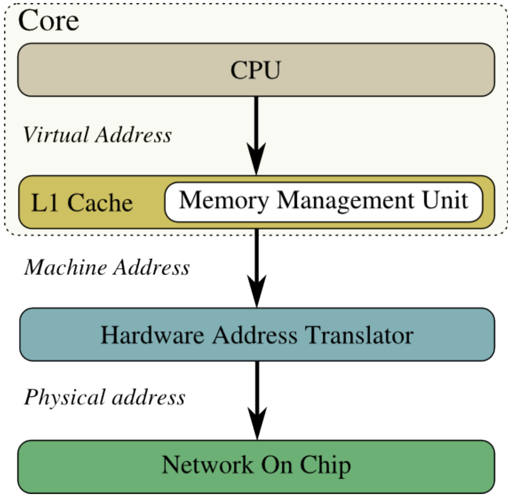


Hypothesis

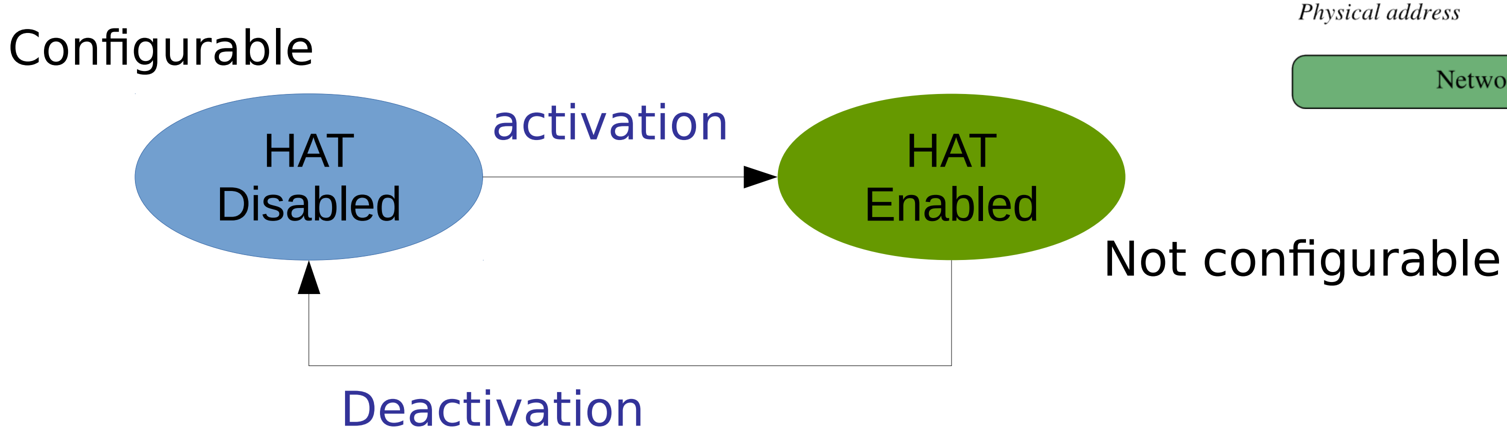
- Our targeted manycore architecture is a clustered architecture with non uniform memory accesses and supports a hardware cache coherence protocol
- Physical attacks are not handled
- Operating Systems running on the platform are untrusted
- The hypervisor manages all the Virtual Machines (VM)
- The 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, machine and physical



HAT - introduction

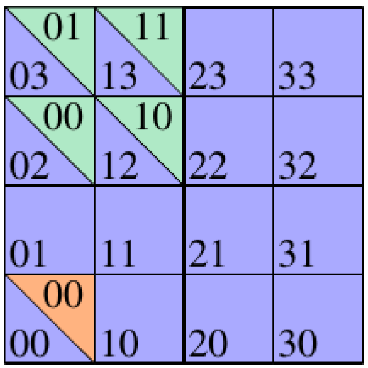


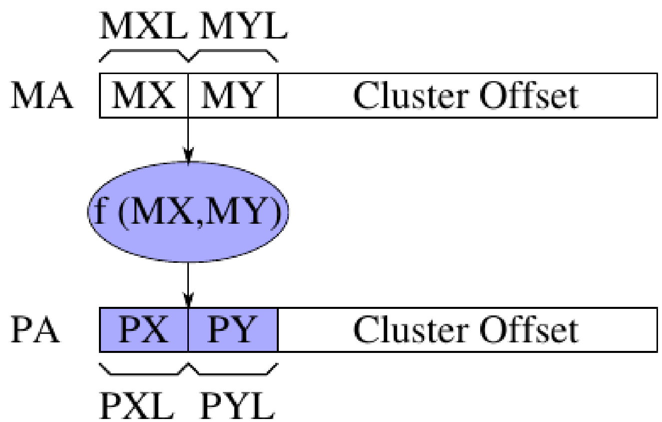
- **Ensure that physical addresses obtained for a VM can only target physical memory or devices located inside the allocated clusters**

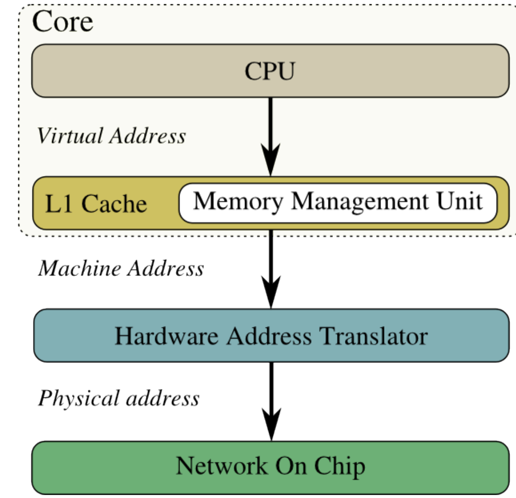


- **Disabled when processor starts**
- **Configured by one of the virtual machine cores**
- **Activated by their own core => not configurable anymore**

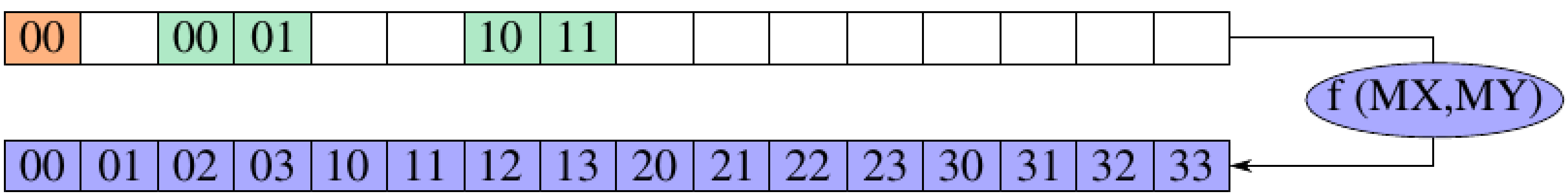
HAT - Internal access





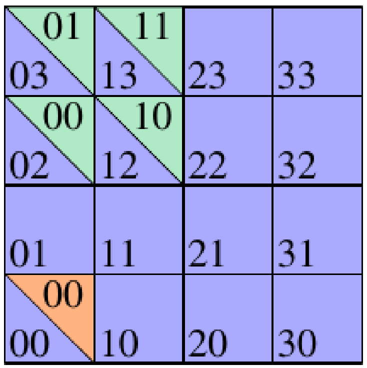


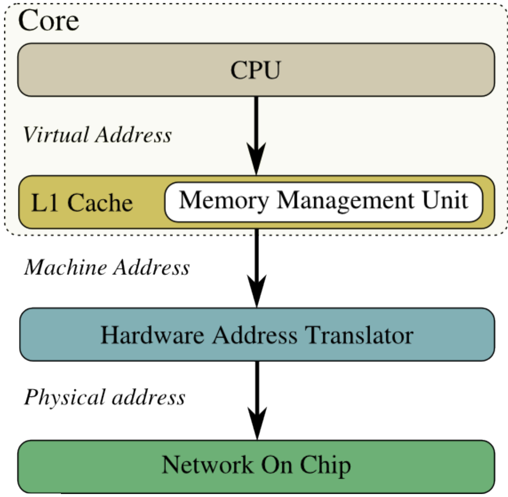
Machine address space

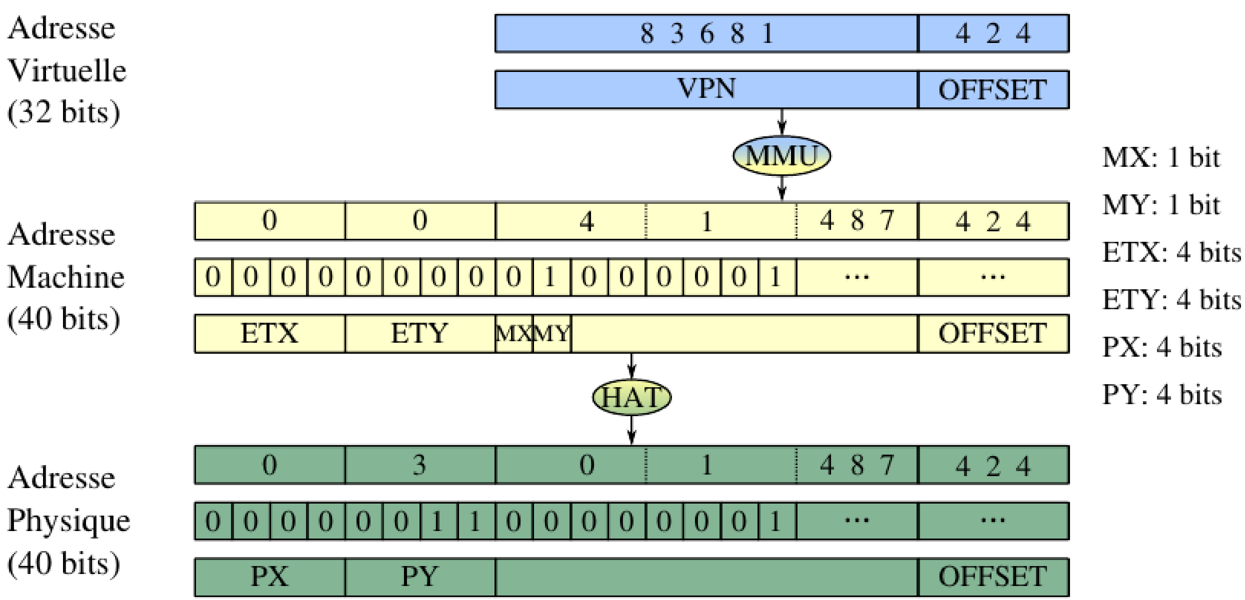


Physical address space

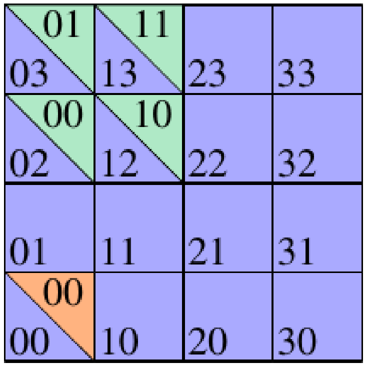
HAT - Internal access

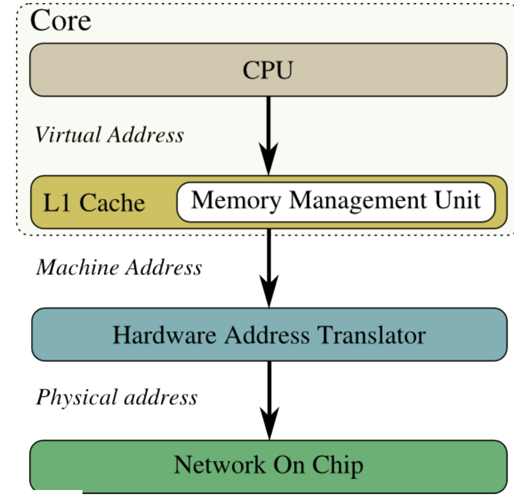


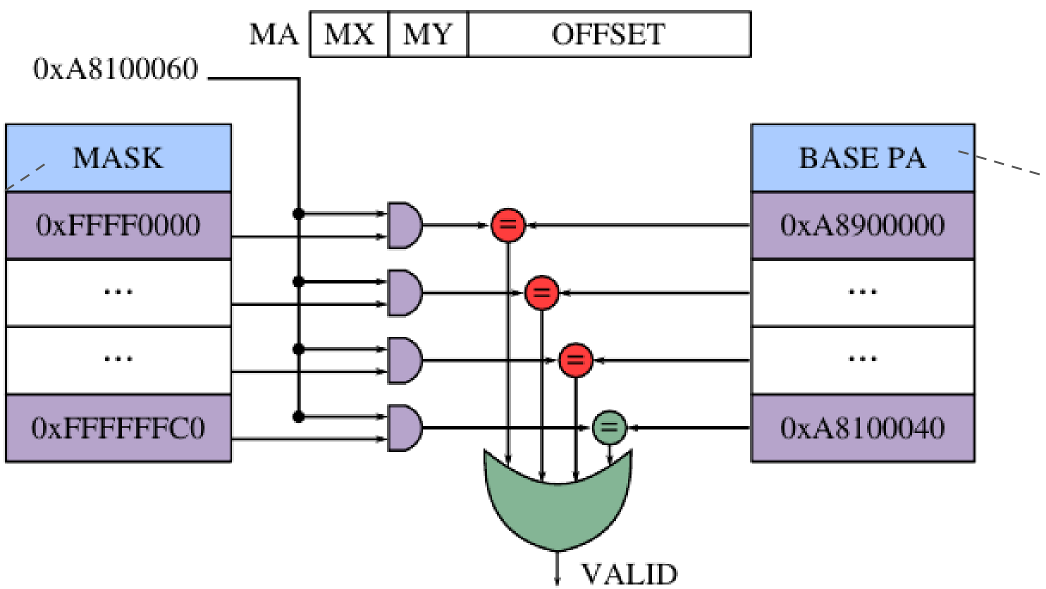




HAT - External access



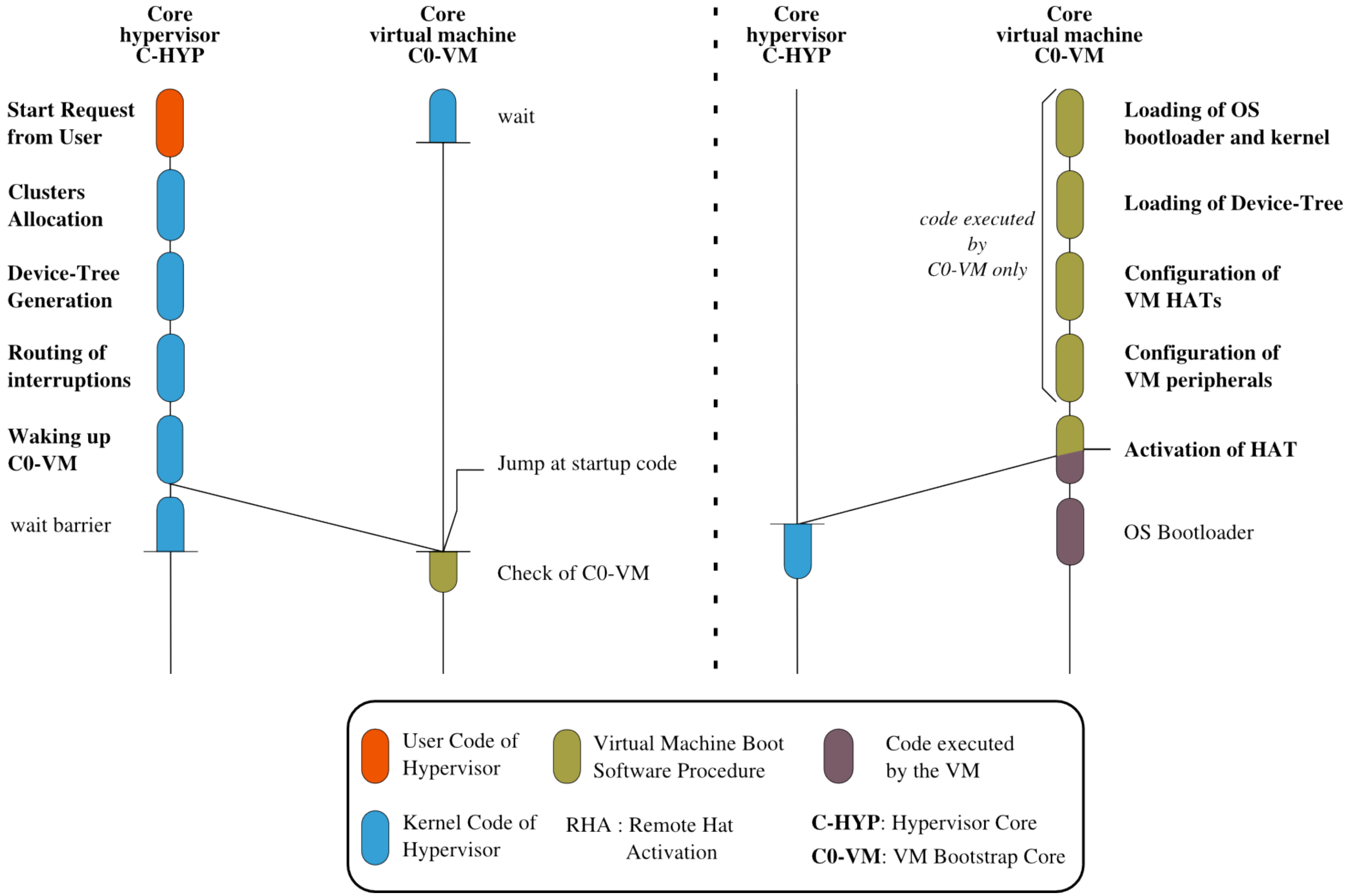




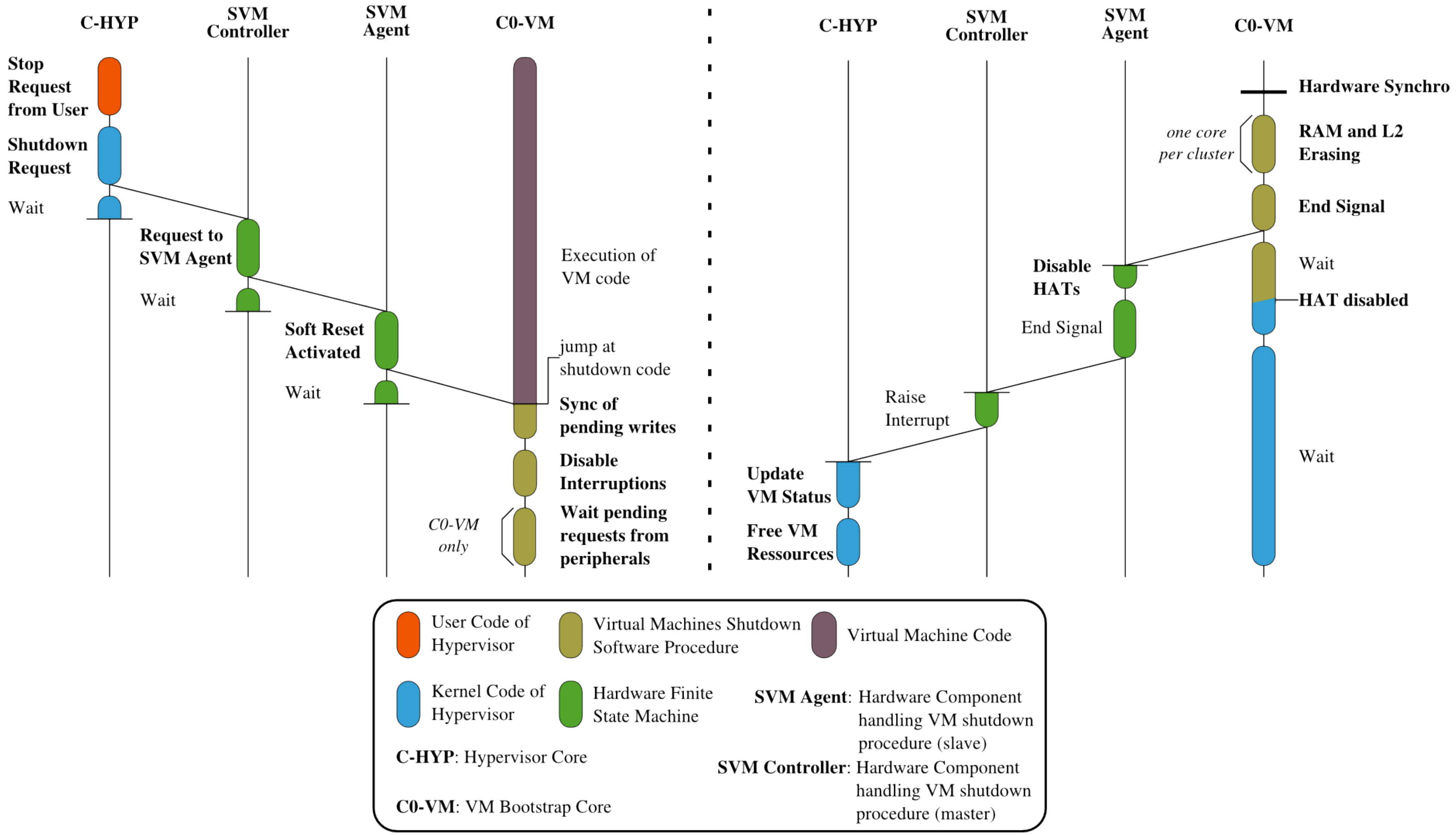
Mask related to the peripheral segment size

Physical segment @ of each peripheral

Virtual Machines Boot Procedure



Virtual Machines Shutdown Procedure

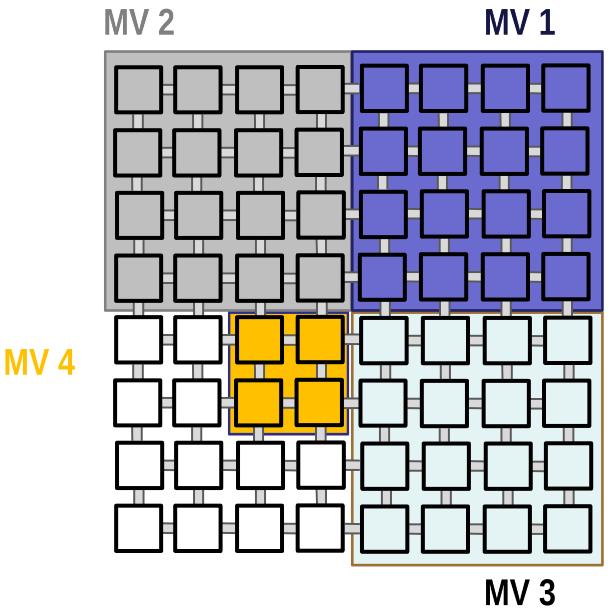


Application isolation

Application isolation

- **Blind hypervisor**
  - Secure deployment of virtual machines (VMs)
  - Non-interference between VMs
  - Non-interference between running VMs and the hypervisor

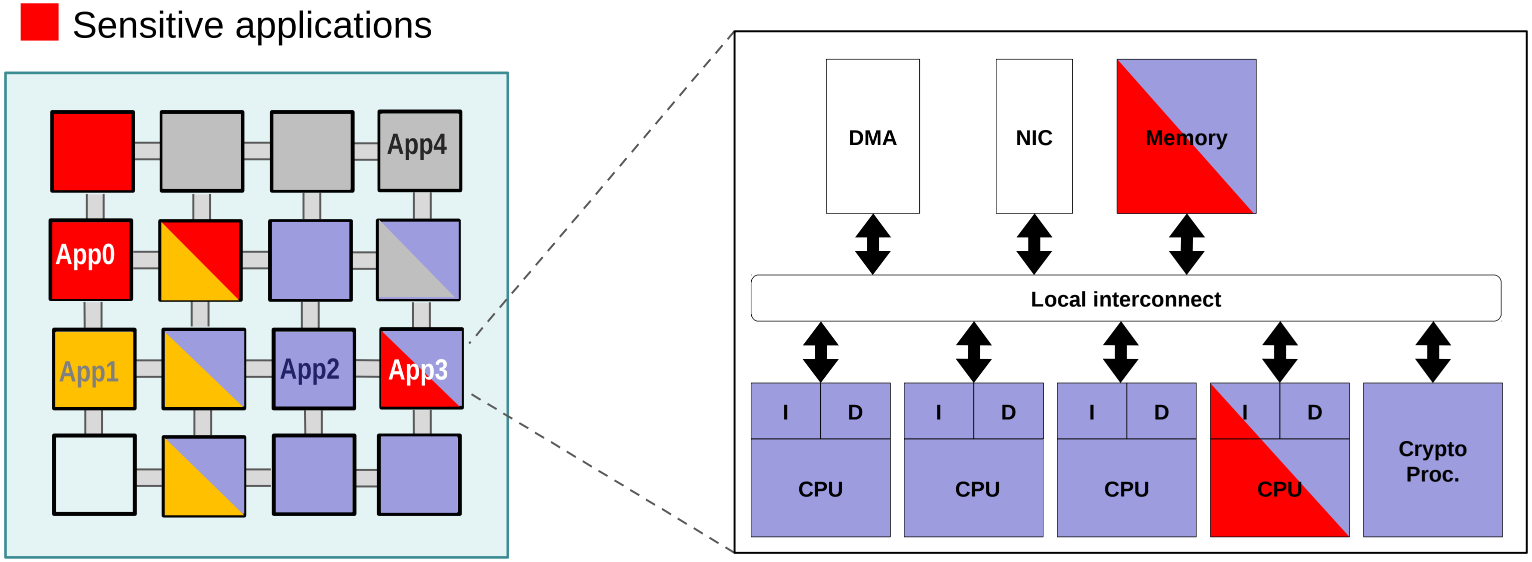
- **What about security within a VM ?**



Motivation

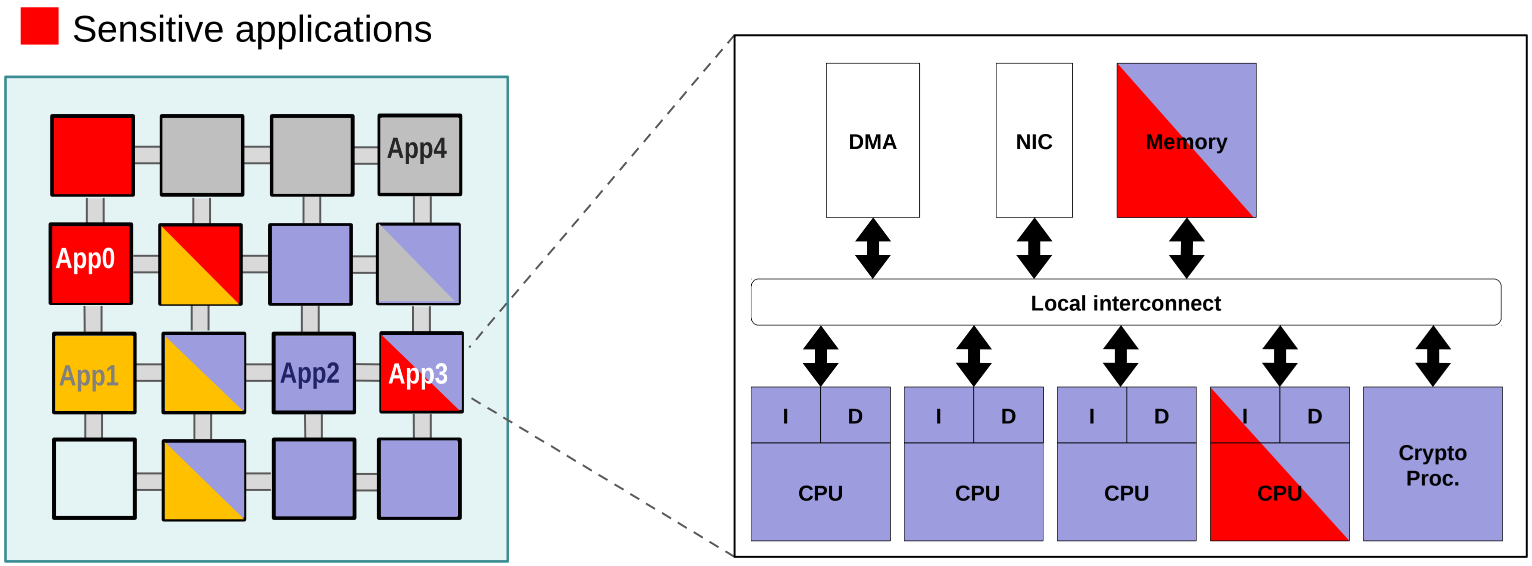
- **Threat model**
  - Sensitive and potentially malicious applications share resources (computing, memory, communication infrastructure)
  - Applications are logically isolated thanks to the MMU

    -> no illegal direct access to the memory



Motivation

- **Threat model**
  - But:
    - *DoS and*
    - *Illegal access to the memory (cache-based and timing-driven Side-channel attacks SCA) between applications are still possible*



Motivation

- **Focus on cache-based SCA**
  - Introduced due to cache sharing (within and across cores)
  - Caches are seen as leakage channels
  - The attacker behaves as a normal process which analyzes its **own activity**
  - Determine cache lines or sets accessed by the victim based on its own memory accesses time
  - Deduce sensitive information
  - Various implementations on different architectures (AES, RSA, ECC, on Intel, AMD, ARM [1-4])

[1] D. J. Bernstein, “Cache-timing attacks on AES”, Technical report, 2005.
[2] Y. Yarom, et al., “Last-level cache side-channel attacks are practical”, in the 23th USENIX Security Symposium, 2015.
[3] Y. Yarom, et al., “FLUSH+RELOAD: A High Resolution, Low Noise, L3 Cache Side-Channel Attack”, in the 23th USENIX Security Symposium, 2014.
[4] D Gruss, et al., “Flush+Flush: A Fast and Stealthy Cache Attack”, in the 13th Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA), 2016.

Motivation cont.

- **Countermeasures against cache-based SCA**
  - Software countermeasures
    - *Changing the implementation of cryptographic algorithms [2]*
  - Hardware countermeasures
    - *Disabling cacheability*
    - *Flushing the cache after each context switch [3]*
    - *Changing the cache design -> Partitioned cache [4]*
    - *Two separate virtual worlds on the same processor [5]*

Application specific

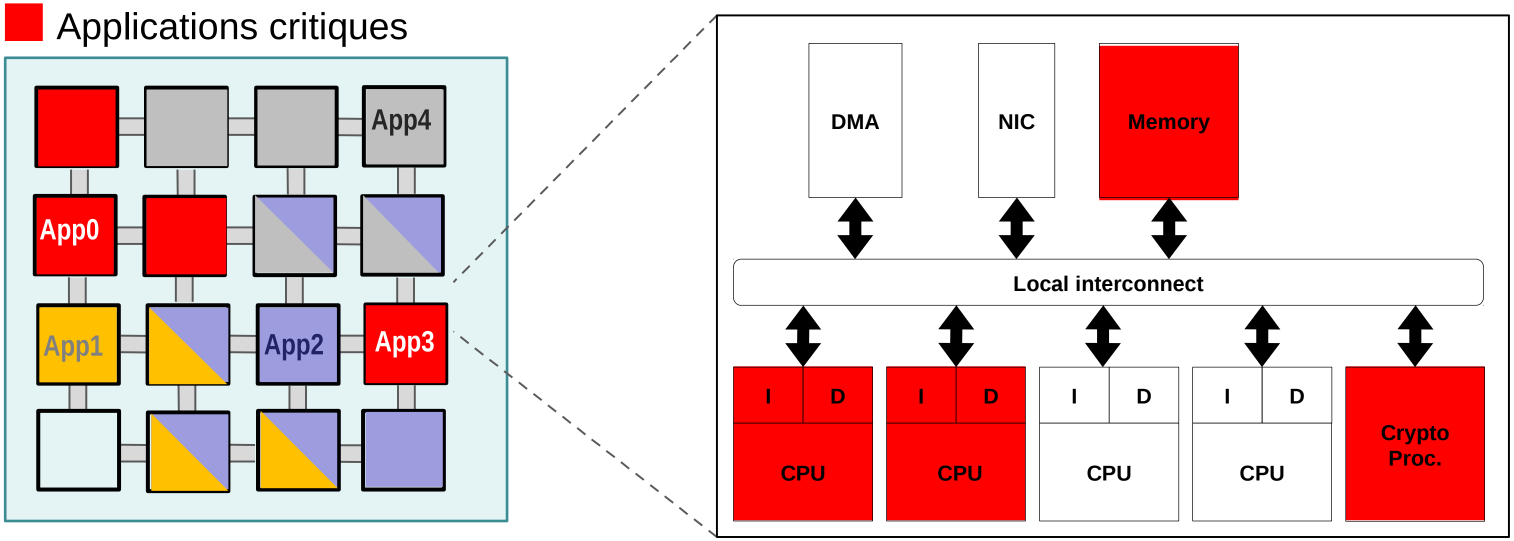
Too expensive

Solution at the processor level

[2] J. Blomer and V. Krummel, "Analysis of Countermeasures Against Access Driven Cache Attacks on AES," Selected Areas in Cryptography,vol. 4876, pp. 96–109, 2007.
[3] Guanciale, et al., "Cache Storage Channels: Alias-Driven Attacks and Verified Countermeasures," in IEEE Symposium on Security and Privacy, 2016.
[4] Wang and R. B. Lee, "New Cache Designs for Thwarting Software Cache-based Side Channel Attacks," in IEEE Symposium on Computer Architecture (ISCA), 2007, pp. 494–505.
[5] www.arm.com/products/processors/technologies/trustzone/

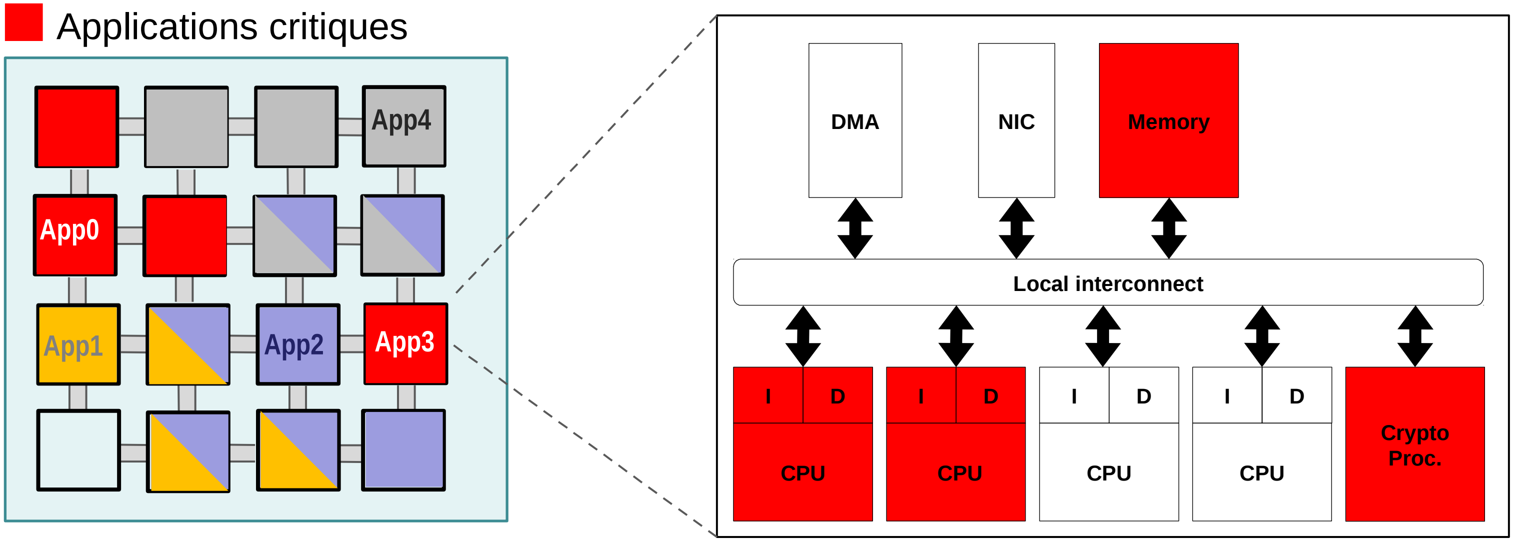
Spatial isolation

- **Isolated execution of sensitive applications**
  - No resource sharing for sensitive applications
  - A trusted entity (OS kernel) is responsible for the dynamic deployment of secure zones
  - Implementation at the deployment and resource allocation level
    - *Application and task mapping,*
    - *resource allocation and*
    - *monitoring services*



Spatial isolation

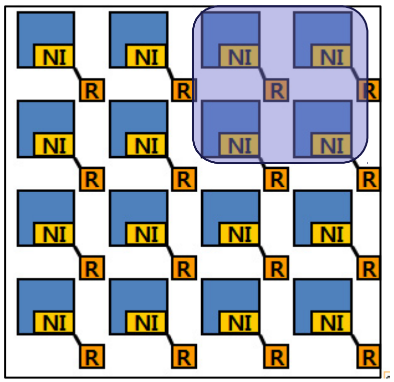
- **Advantages**
  - Non-application specific,
  - Portable
  - Taking advantage of the wide number of resources on many-core
- **But**
  - expected under utilization of resources and thus, performance overhead



Isolation strategies

Static Secure Zone size:

- The size fulfilling all the application needs
- Restrained size

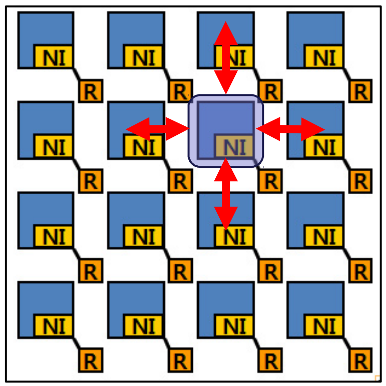


✓ Best isolated apps performance achieved

✗ Isolated apps waiting time before execution

✗ Need to partially know isolated apps

Dynamic secure zone size

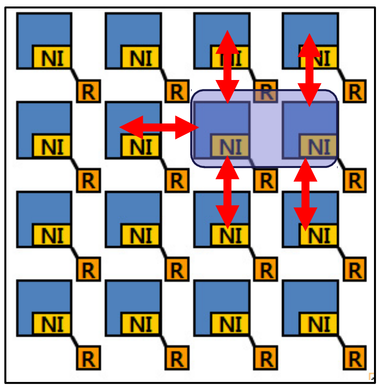


✓ Generic scenario

✓ Best resource utilization rate and minimum performance overhead

✗ Do not prioritize the isolated apps performance

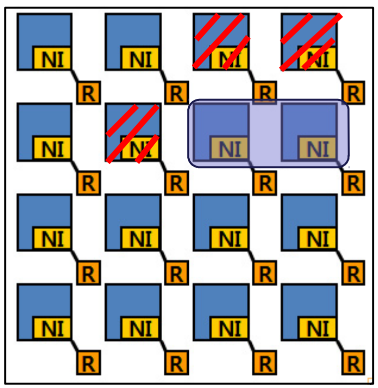
Dynamic secure zone size but guaranteeing a minimum size



✓ Minimum performance guaranteed

✗ Do not prioritize the isolated apps performance

Dynamic secure zone size with resource reservation



✓ Good trade-off resource utilization rate & isolated apps performance

✗ Need for smarter parameters when selecting the reserved resources

Conclusion

- By implementing the concept of blind hypervisor, we avoid that a corruption of the hypervisor leads to a breach of confidentiality or integrity of a virtual machine.

- Sensitive applications within a virtual machine can be isolated by taking advantage of available resources on manycore architectures

References

- **Clément Devigne**
  - **"Exécution sécurisée de plusieurs machines virtuelles sur une plateforme Manycore"**
    - **THÈSE DE DOCTORAT DE L'UNIVERSITÉ PIERRE ET MARIE CURIE**
- **Maria Mendez**
  - **"Spatial Isolation against Logical Cache-based Side-Channel Attacks in Many-Core Architectures**
    - **THÈSE DE DOCTORAT DE L'UNIVERSITÉ BRETAGNE SUD**

Merci pour votre atttention