Executing Secured Virtual Machines within a Manycore Architecture

Clément DÉVIGNE

Jean-Baptiste BRÉJON, Quentin MEUNIER, Franck WAJSBÜRT

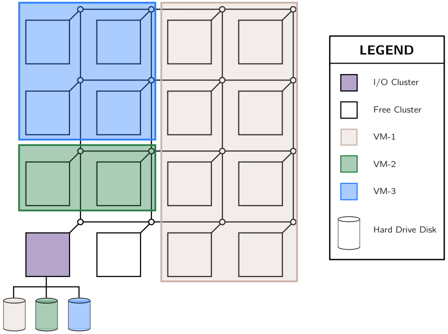
LIP6

October 27, 2015

Objectives

To execute several secured virtual machine on the same manycore

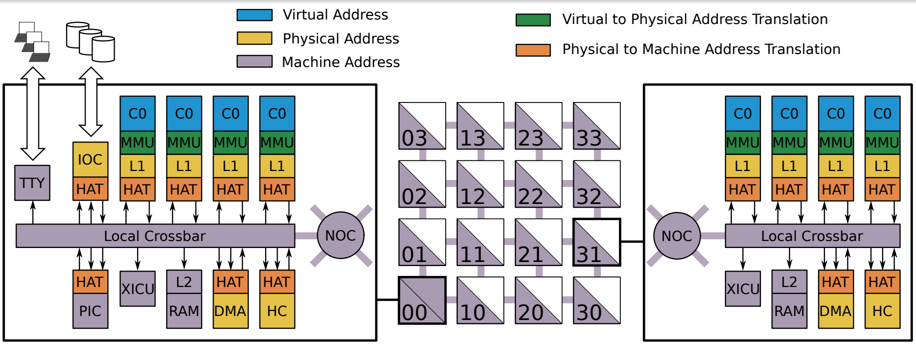
- Example of a TSUNAMY platform with 3 virtual machines deployed
- Each virtual machine has an exclusive Input/Ouput Controller channel
- Each disk contains its own filesystem, a bootloader and the kernel code



Hypothesis

- 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
- Our targeted manycore architecture is a clustered architecture with a non uniform memory access and a hardware cache coherence protocol
- Three address spaces: virtual, physical and machine

Tsunamy Architecture



- All clusters contain:
  - 4 MIPS cores with their first level (L1) caches
  - 1 second level (L2) cache
  - 3 internal peripherals: XICU, DMA, HC
  - A local crossbar
  - The Hardware Address Translator
- The I/O cluster additionally contains:
  - A terminal controller (TTY)
  - A hard-drive disk controller (IOC)
  - A Programmable Interrupt Controller (PIC)

Why Do We Not Use a Memory Management Unit (MMU)

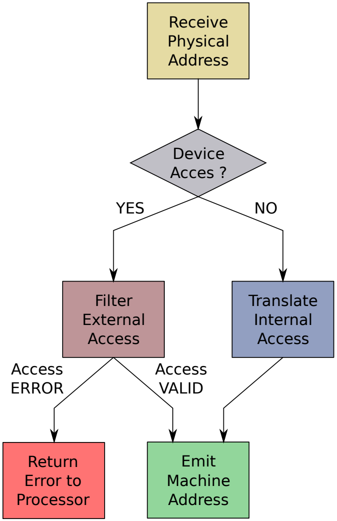
- A MMU uses a Translation Lookaside Buffer (TLB) to speed up address translation
  - Non negligible hardware overhead, including the logic to manage the TLB misses
  - Slower to perform address translation because of the TLB misses overhead
- The hypervisor must create the page table for the memory 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)

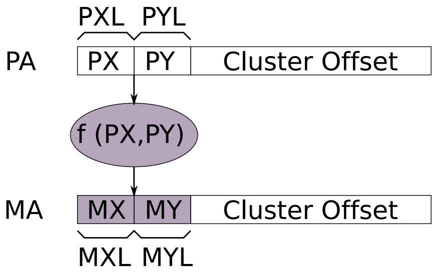
- The HAT performs the translation from physical addresses to machine addresses
- The HAT operates with a coarser granularity (cluster granularity)
- Configured once by the hypervisor at the start of an operating system and placed behind each initiator in the architecture
- The HAT only needs topology information to perform address translation
- The translation mechanism only takes 1 cycle
- The hardware cost involved for a HAT is 175 bytes of memory

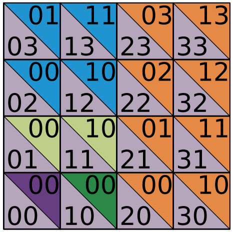
HAT: Overview

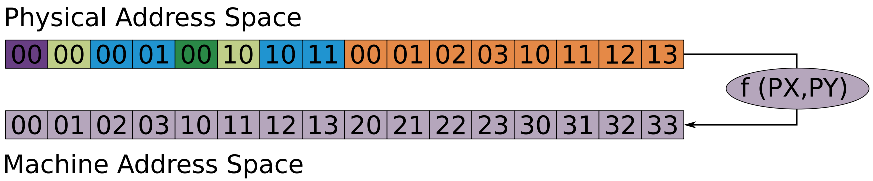
- Two types of addresses target:
  - Module included in a cluster of the same virtual machine (internal access)
  - Device outside the virtual machine (external access)
- An internal access cannot fail and its translation always targets a machine address inside a cluster of the virtual machine



HAT: Internal Accesses Mechanism

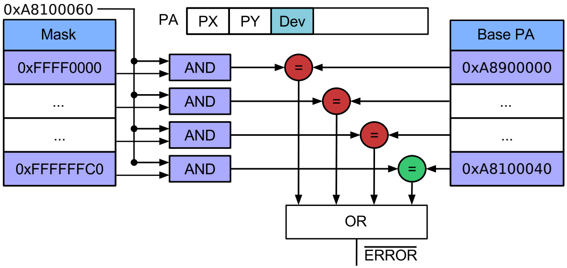




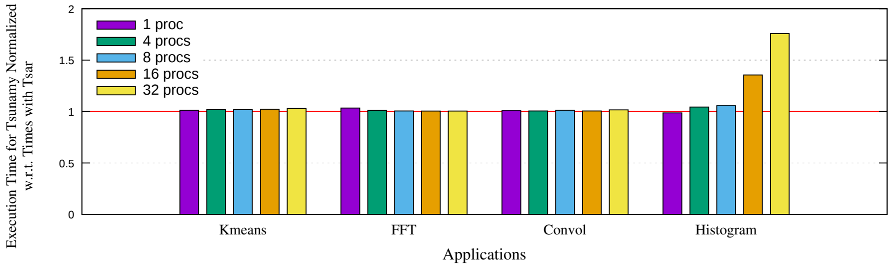


HAT: External Accesses Mechanism

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



Preliminary Results

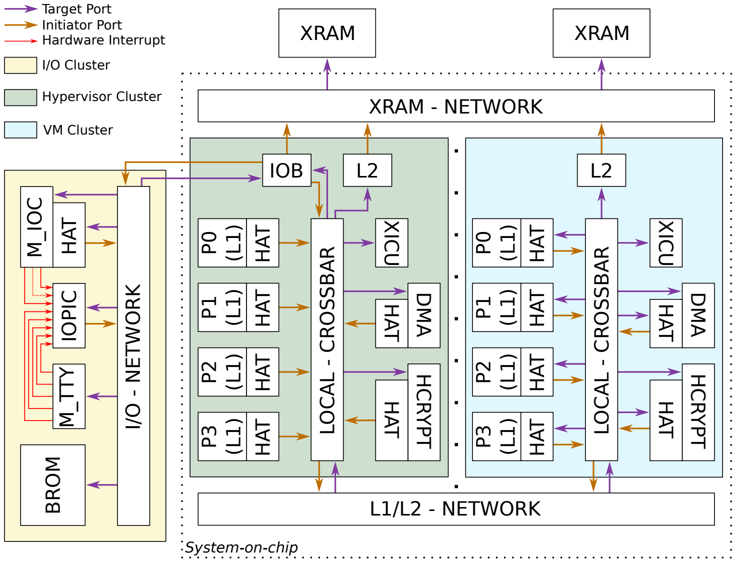


- Average overhead: < 3 %

Conclusion

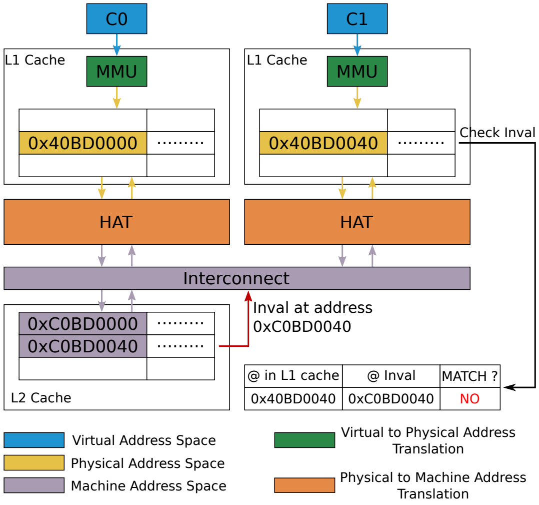
- A platform comprising 16 clusters and supporting hardware cache coherence, running 5 operating system instances on a various number of clusters
- The architecture is described in SystemC at the cycle-accurate level using the SoCLib components library
- Light hardware overhead and very low time overhead
- Future work:
    - To deal with the specification and implementation of the procedure required to stop a running virtual machine
    - To develop another platform to take into account I/O accesses in a more realistic way

Future Work


Target Port
Initiator Port
Hardware Interrupt
I/O Cluster
Hypervisor Cluster
VM Cluster
XRAM
XRAM
XRAM - NETWORK
IOB
L2
L2
M_IOC
HAT
IOPIC
M_TTY
BROM
I/O - NETWORK
P0 (L1)
HAT
P1 (L1)
HAT
P2 (L1)
HAT
P3 (L1)
HAT
LOCAL - CROSSBAR
XICU
HAT
DMA
HAT
HCRYPT
P0 (L1)
HAT
P1 (L1)
HAT
P2 (L1)
HAT
P3 (L1)
HAT
LOCAL - CROSSBAR
XICU
HAT
DMA
HAT
HCRYPT
L1/L2 - NETWORK
System-on-chip

Hardware Cache Coherence Problem

- The cache coherence is fully handle by hardware
- L1 and L2 cache aren't in the same address space
- We need to translate coherence messages



@ in L1 cache	@ Inval	MATCH ?
0x40BD0040	0xC0BD0040	NO