I-MECH: real-time virtualisation for industrial automation

Marco Solieri

Pisa, 2019-05-20

3rd OSPM summit

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Hipert/Lab
High Performance Real Time Lab

EVIDENCE®
EMBEDDING TECHNOLOGY

I-MECH project

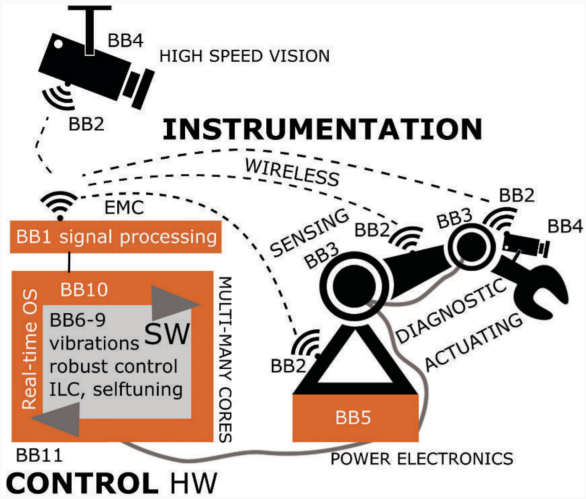
I-MECH consortium

Members:

Applications machines:

- Generic substrate carrier
- 12” wafer stage
- Inline filling and stoppering and tea bag machines
- Medical manipulator
- Modular robotic arm
- …

I-MECH objectives



Smart Control Layer:
"Modular, unified, Hardware and Software motion-control building blocks implementing a service-oriented architecture paradigm."

I-MECH control platform

Control platform building block

Hardware

- industrial x86 architecture
- $\geq$ quad-core
- $\leq$ 1.5 KEUR

Application

- motion control cycle $\leq$ 500 μs
- EtherCAT sample rate $\geq$ 20 *KHz*
- loop latency $\leq$ 100 μs

Real-time operating system

- Simulink integration
- EtherCAT support

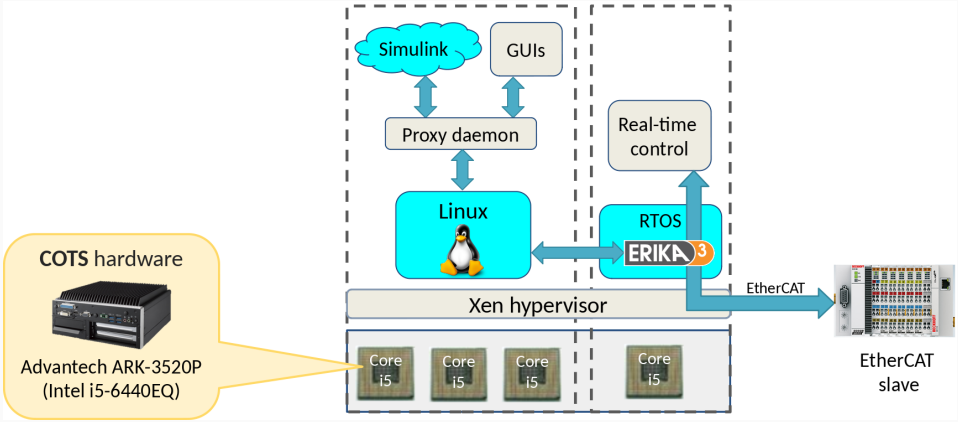
Hypervisor

- Linux, VxWorks, Windows
- mixed criticality: control & HMI

RTOS: ERIKA Enterprise

- Developed by Evidence for automotive ECUs
- Minimal footprint (few KB) and multi-core support
- Certifications: OSEK/VDX, ISO26262 (ASIL B in-progress)
- Reference standards: MISRA-C, AUTOSAR OS
- Open-source: GPL (optional linking exception with a fee)
- Used by several companies and research projects,
  e.g. Magneti Marelli, Vodafone Automotive, Ariston, Piaggio
- `https://www.erika-enterprise.com`

I-MECH extension of Erika



Real-time virtualisation

Hypervisor: Xen

- Developed by Linux Foundation
- Wide support for unmodified guest OSs
- Free and open-source licensed
- Born for multi-tenant servers or general purpose consolidation, now leaning towards embedded/ safety-critical applications (e.g. null scheduler, `dom0less` on Arm)

Cache sharing and memory access predictability

Last-level cache (LLC) is shared in x86 processors

- Shared cache allows different cores to contend for storing data
- Hit probability depends on other cores

LLC miss delay from tens to hundreds cycles

- Jitter and average response time increase
- Pessimism in worst case estimations explodes

LLC partitioning

Three solutions for three processor families

1. Intel Xeon
2. Recent non-Xeon Intel
3. AMD and old Intel

Cache partitioning on Intel Xeon

Cache Allocation Technology (CAT)

- Available on Xeon D, E3 v4, E5 v3, E5 v4, Scalable families
- Direct Xen (and Linux) support from Intel
- Hardware support for way partitioning
  - `xl psr-cat-cbm-set [OPTIONS] domid cbm`
  - Associate a capacity bitmask, i.e. assign ways (non-exclusively), to a given domain
- Negligible inter-core LLC interference within domU
- Smaller LLC availability

Cache partitioning on recent non-Xeon Intel

Some commercial real-time hypervisor and OS on some 6th gen (default or “low-latency” undocumented configuration):

- Negligible inter-core LLC interference between real-time guests/tasks
- Smaller LLC availability

Cache partitioning on recent non-Xeon Intel

Some commercial real-time hypervisor and OS on some 6th gen (default or “low-latency” undocumented configuration):

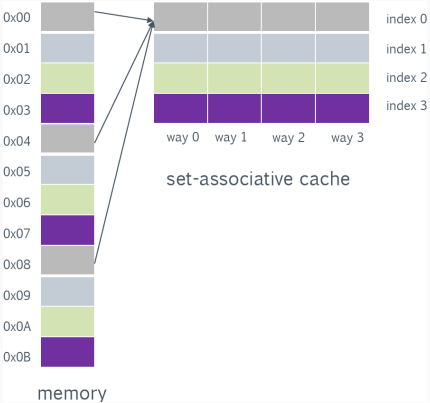
- Negligible inter-core LLC interference between real-time guests/ tasks
- Smaller LLC availability

CAT is there, hidden

- hidden CAT support available
- work in progress: probing functionalities

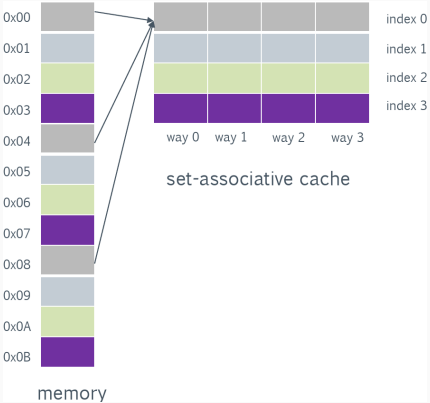
Cache partitioning on AMD and old Intel

Partition address space such that different partitions $\rightarrow$ different sets

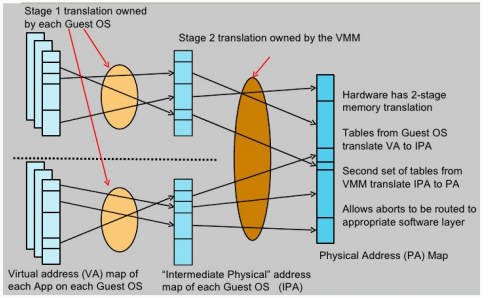


Cache partitioning on AMD and old Intel

Partition address space such that different partitions → different sets



Virtualised, two-stage translation partitions colours to system domains



Challenges of cache colouring on Intel

- Cache mapping function may change
  $\rightarrow$ patiently discover it
  $\rightarrow$ less colours available
- Cache colour bits overlap with DRAM banks addressing bits
  $\rightarrow$ extend/restrict colouring
  $\rightarrow$ reconfigure memory controller if possible
- DMA does not translate with MMU
  $\rightarrow$ colour also IOMMU translations

Challenges of cache colouring on Xen

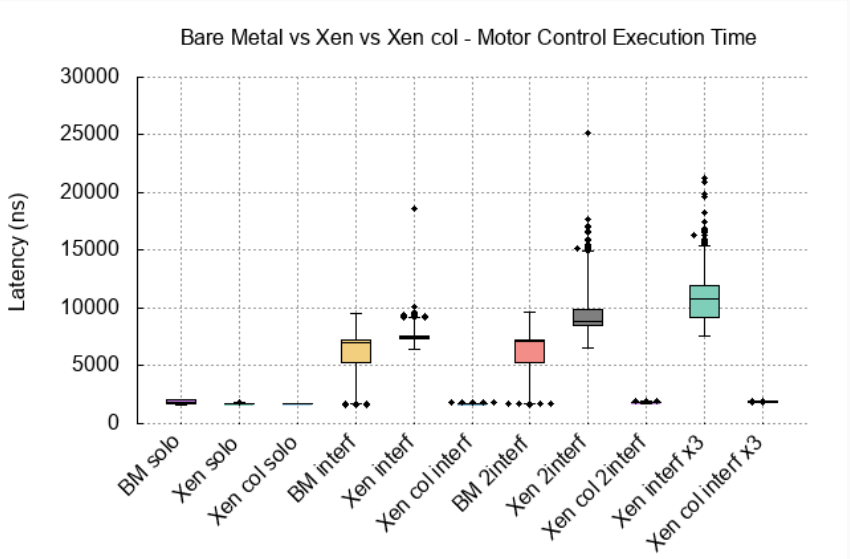
- memory page pool data structure broken (the “buddy”)
  - big pages cannot be used – colours would be mixed
  - memory-contiguity orientation useless – memory is colour-striped

  $\rightarrow$ one data structure per colour
  - linked lists of fixed-length array?
  - red-black trees?
- Xen needs cache colouring too

  $\rightarrow$ dynamic recolouring procedure
- Memory ballooning with `dom0`

  $\rightarrow$ restrict usage so to avoid colour mixing

Evaluation of cache colouring on Xen



Conclusion

Further works

- DRAM bandwidth arbitration
  - Intel hardware RDP support
  - Hypervised throttling
    already experimented on Arm by project HERCULES
- Memory co-scheduling
  1. PREM-like task model: coarse-grained memory/ compute phases
  2. Hypervised scheduling of memory access
     already experimented on Arm platform by project HERCULES
- Hidden CAT
  - More extensive analysis of Intel processors
  - Propose upstream

Thanks

I-MECH live demo by Evidence at lunchtime:
DC motor control via EtherCAT from Erika on Xen