Predictability in Memory Hierarchies for Real-Time Multi-Core Embedded Systems

Marco Solieri

2018-09-13

Joint work with T. Kloda, L. Miccio, R. Mancuso, N. Capodieci, M. Bertogna

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Hipert/Lab
High Performance Real Time Lab

Partially supported by: HERCULES project (Innovation Action, H2020), and Xilinx inc.

High-Performance Embedded Systems for Real Time Application

Commercial off-the-shelf multi-core system-on-chip products

- NVIDIA Tegra: X1, X2 (GPGPU)
- Xilinx UltraScale+: XCZU7, XCZU9 (FPGA)

Memory hierarchy and architecture jeopardise predictability,
thus applicability and flexibility

1. Cluster-shared last-level cache without hardware lockdown
   $\Rightarrow$ cores contention increases both average latency and jitter
2. Random replacement policy
   $\Rightarrow$ worst-case access time extremely pessimistic
3. System-shared SDRAM subsystem
   $\Rightarrow$ contention increases both average latency and jitter

Real-Time Assisting Hypervisor

Hypervisor

- Software fills hardware design gaps
- Seamless integration with legacy solutions

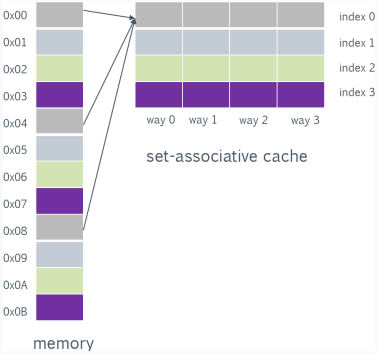
Jailhouse hypervisor

- Siemens maintained, reasonably good maturity
- targets critical systems: resource partitioning
- small codebase (hence easily certifiable)
- free and open-source licenced (hence merrily hackable)

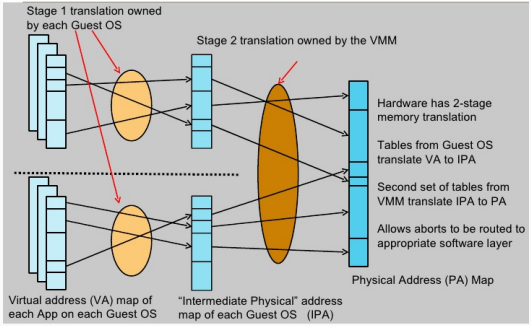
Contention to Last-Level Cache

Cache colouring support

Page colouring

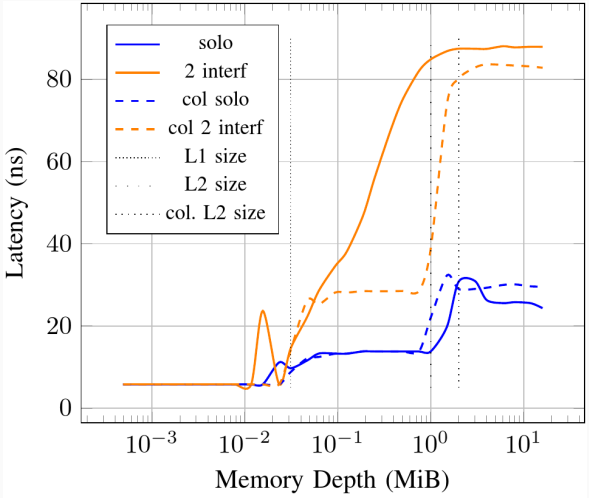


Virtualised, two-stage translation



- User interface: Jailhouse configuration
- See also XVisor support (Modica, Biondi, Buttazzo, Patel; 2018)

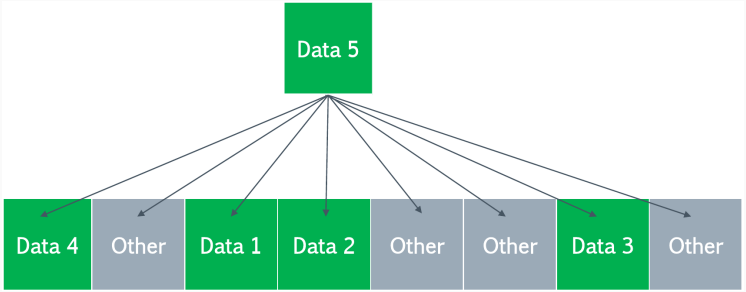
Cache colouring evaluation on access latency



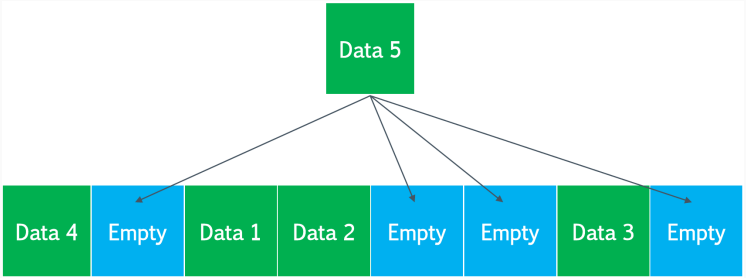
(a) Sequential Access (64 B stride).

Pseudo-Random Cache Replacement Policy

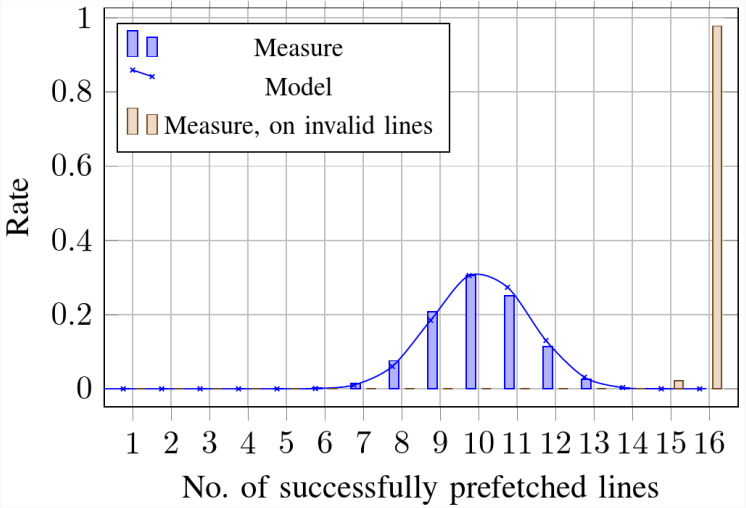
Random replacement and self-eviction



Explicit replacement support: Jailhouse hypercall



Explicit replacement evaluation on prefetch: micro-benchmark

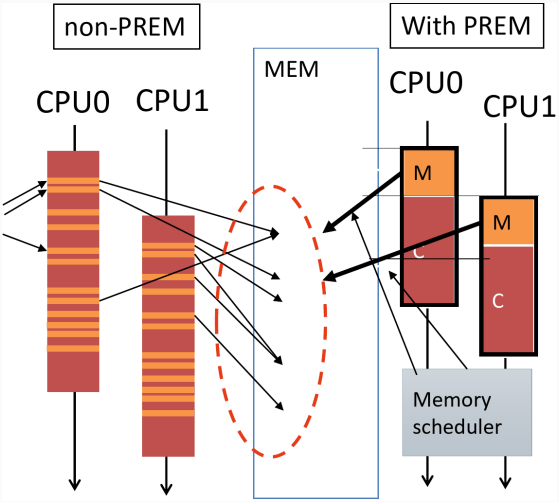


Explicit replacement evaluation on prefetch: benchmarks

Application	*Conf.*	*Time (Kcycles)* *maximum*	*average*	*L2 miss per line*
Heap sort	base	8,089.871	7,692.389	2.1973
	pr.inv.	7,600.765	7,528.418	1.0127
Seq. iterator	base	99.344	91.393	1.5107
	pr.inv.	80.492	70.820	1.0000
Rand. iterator	base	272.061	255.475	1.5078
	pr.inv.	214.077	193.972	0.9990
SHA-1	base	808.105	791.198	1.0000
	pr.inv.	800.823	783.435	1.0000
Convolution	base	14,215.224	14,203.511	1.0010
	pr.inv.	14,236.808	14,209.904	1.0000

Contention to Shared DRAM

Predictable Execution Model and Jailhouse support



- (Pellizzoni, Betti, Bak, Yao, Criswell, Caccamo, Kegley; 2011)
- User interface: Jailhouse hypercalls lock/unlock a memory mutex

PREM co-scheduling evaluation: TBD

Not available, yet.

Conclusion

Summary

What we have

1. Cache colouring to address mutual evictions on caches
2. Explicit replacement to address self-evictions on caches
3. PREM support to address DRAM contention

What else the HERCULES consortium have on Jailhouse

1. MemGuard by CTU in Prague
2. PREMising compiler by ETH in Zurich

Further works

1. Progress by keeping up-to-date with
   - NVIDIA SoCs with cluster-shared L2 and isle-shared L3 caches
   - ARM v8.4 definition with more advanced cache technologies
2. Widen applicability
   - Public release and upstream integration
   - Exploit accelerators (FPGA and GPGPU) to extend possibilities