NXP | FTF 2016 TECHNOLOGY FORUM

BEST VIRTUALIZATION PERFORMANCE WITH KVM ON ARM BASED QORIQ SOCS

FTF-DES-N1887

DIANA CRĂCIUN
SOFTWARE ENGINEER
FTF-DES-N1887
MAY 16, 2016

Software Products and Services

Visit us in the Tech Lab – #247

Development Tools

- CodeWarrior

Runtime Products

- VortiQa Software Solutions

Solutions Reference

- IOT Gateway
- OpenWRT+

Integration Services

- Security Consulting
- Hardened Linux

Linux® Services

- Commercial Support
- Performance Tuning

Accelerate **Customer Time-to-Market**

Deliver **Commercial Software, Support, Services and Solutions**

Simplify **Software Engagement with NXP**

Create **Success!**

Find us online at www.nxp.com/networking-services

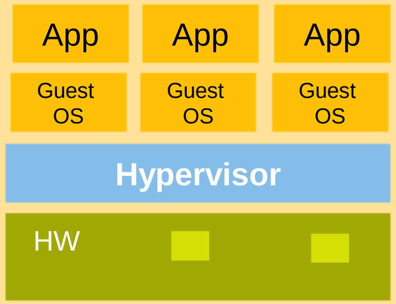
AGENDA

- Introduction
- KVM/QEMU
- ARM Virtualization Extension
- KVM on ARM
- Results
- Conclusions

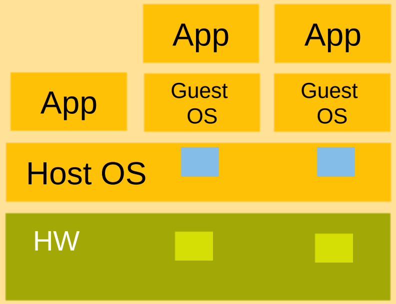
INTRODUCTION

Virtualization and Hypervisors

- **Virtualization** – Hardware and software technologies that provide an abstraction layer that enables running multiple operating systems on a single computer system
- A **hypervisor** is a software component that creates and manages virtual machines which can run guest operating systems



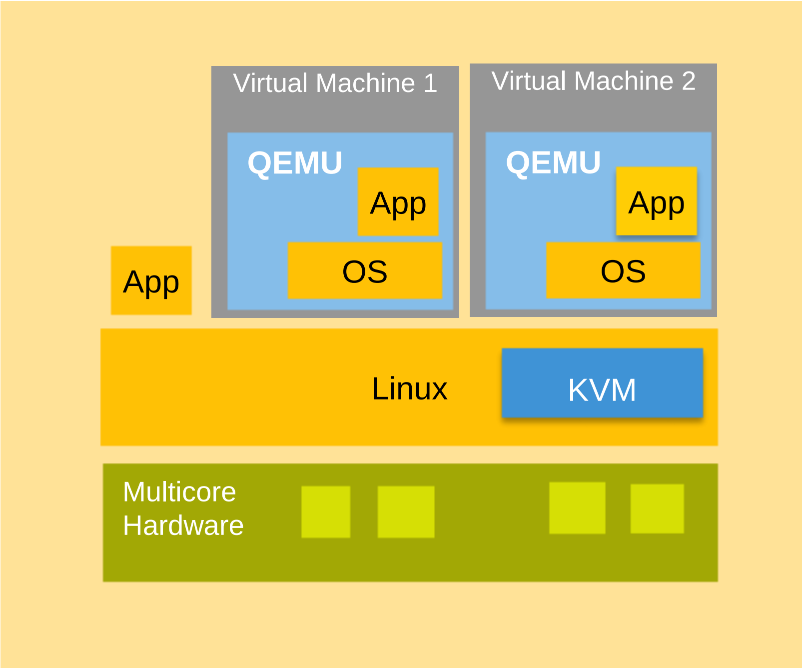
- Hypervisor runs “bare metal”



- Hypervisor is integrated in Host OS
  - Reuses OS infrastructure
- Host OS runs other applications

KVM/QEMU

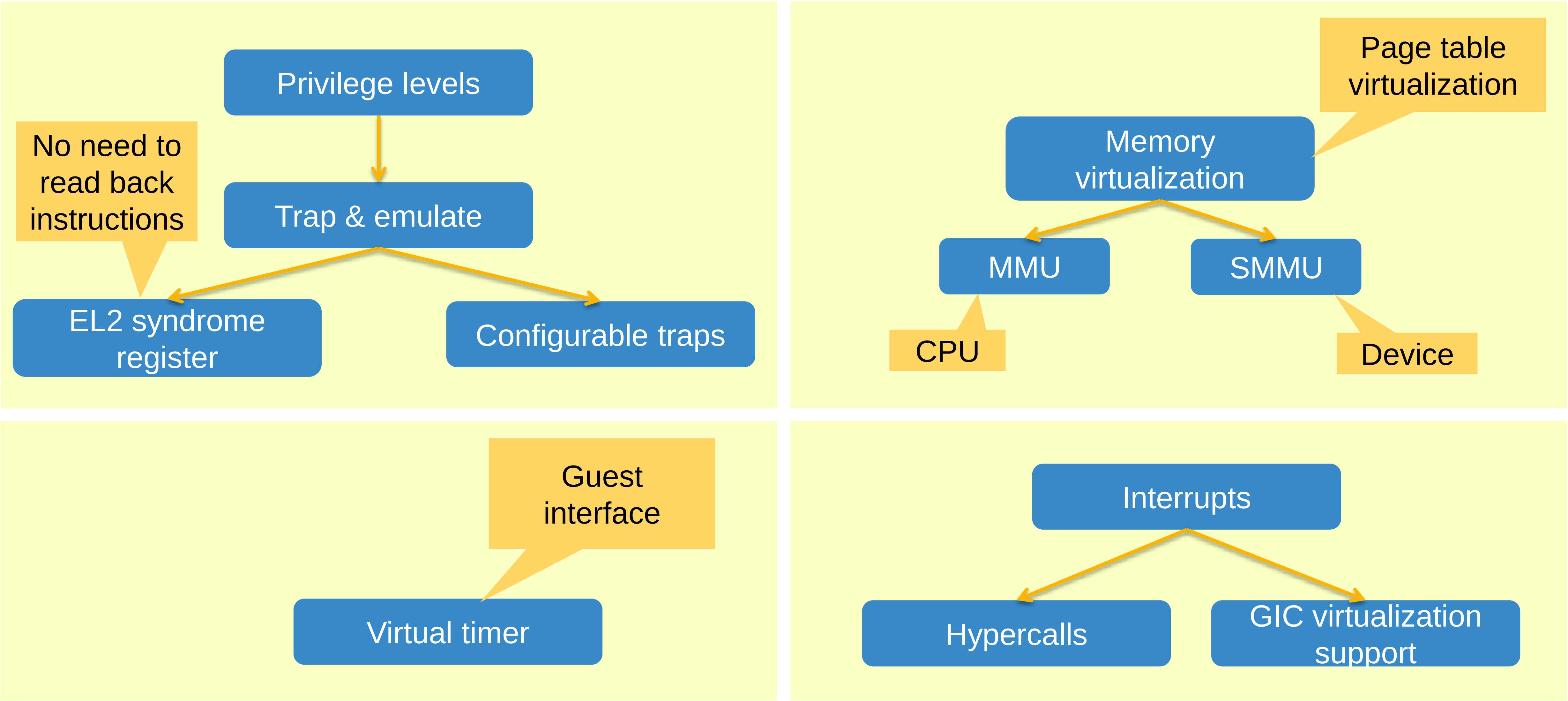
KVM/QEMU



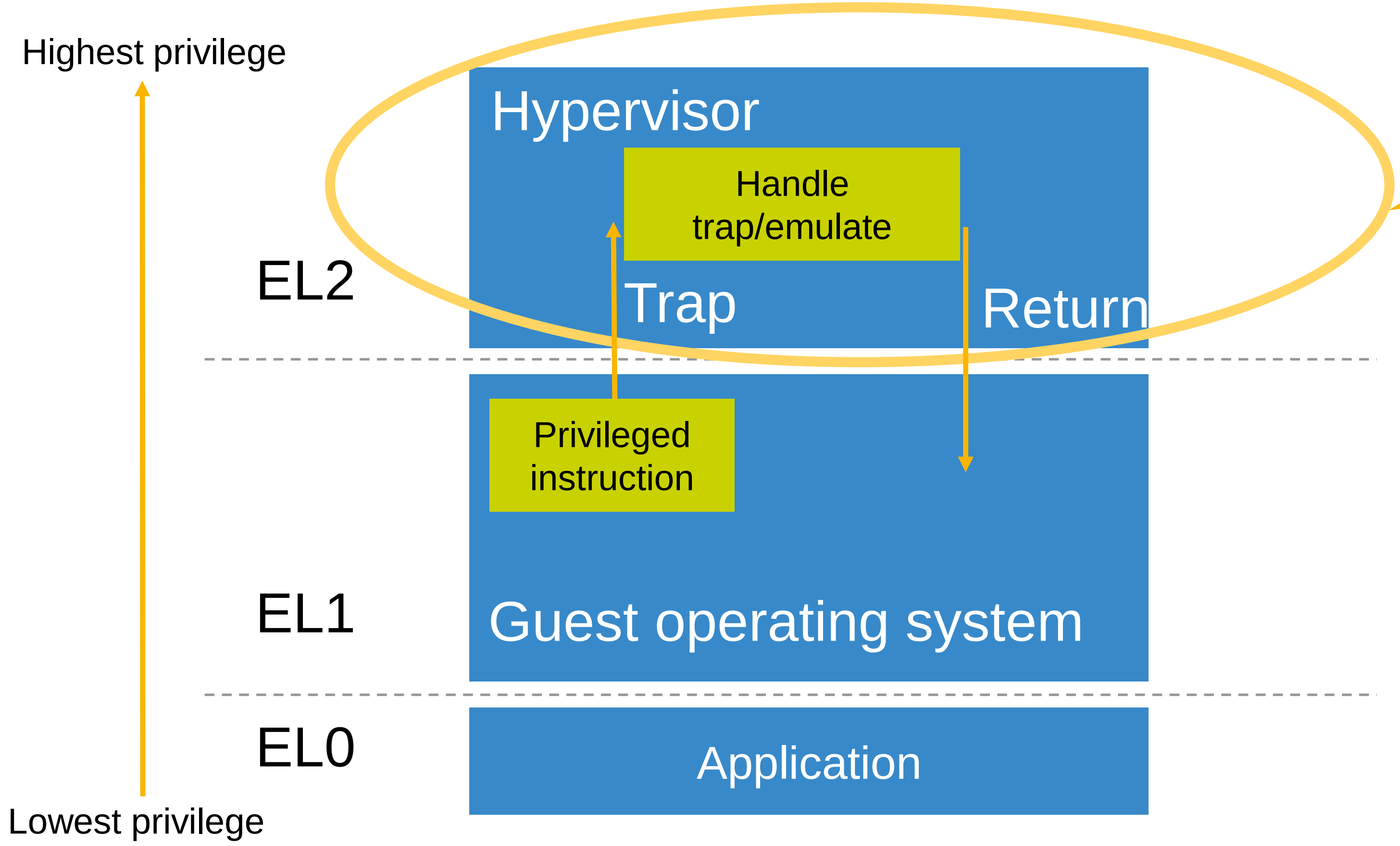
- KVM/QEMU – open source virtualization technology based on the Linux® kernel
- KVM is a Linux kernel module
- QEMU is a user space emulator that uses KVM for acceleration
- Run virtual machines alongside Linux applications
- No or minimal OS changes required
- Virtual I/O capabilities
- Direct/pass thru I/O – assign I/O devices to VMs

ARM VIRTUALIZATION EXTENSIONS

ARM Virtualization extensions



Privilege Levels

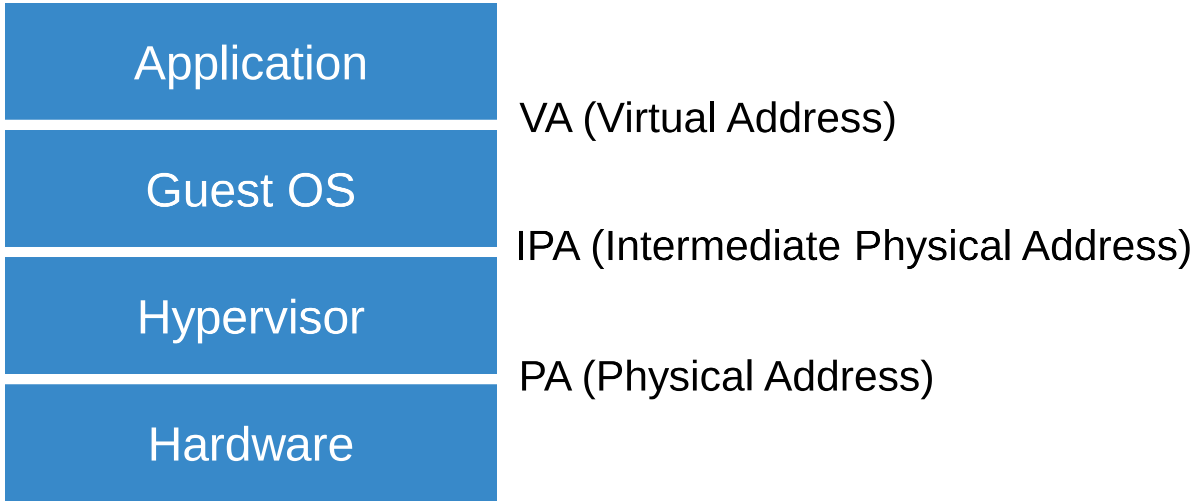


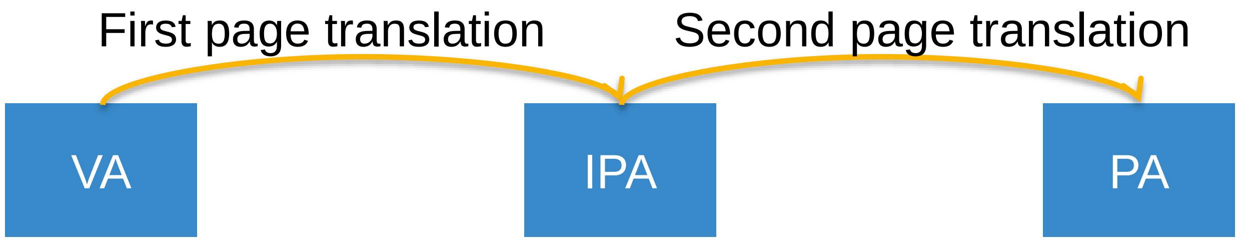
Transition to EL2 via exception
- Trap & emulate, interrupts
- Hypercalls

Separate stack pointer for each EL

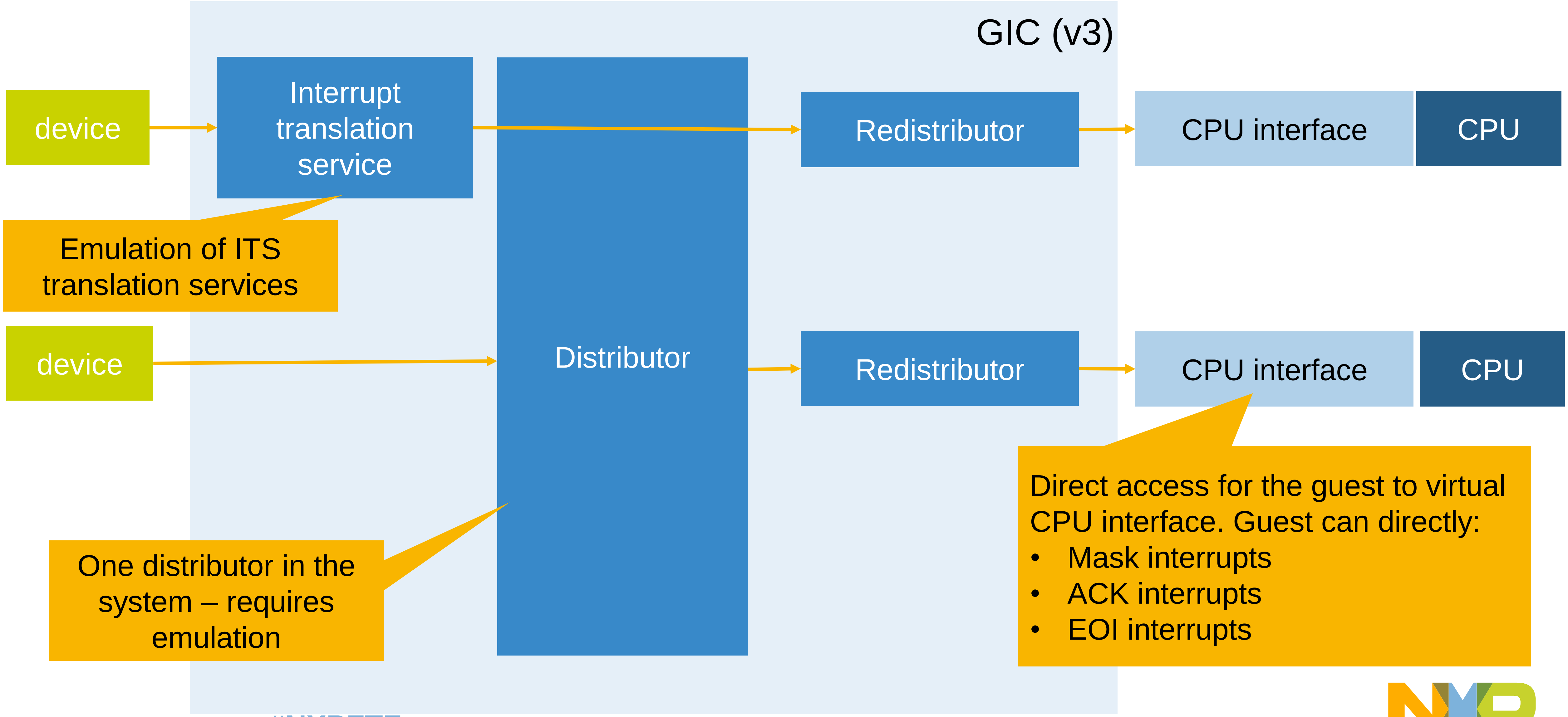
Memory Virtualization

- 2 stage memory translation
- First page translation translates memory from VA to IPA
  - Owned by the guest
- Second stage translation translates from IPA to PA
  - Tables maintained by the hypervisor

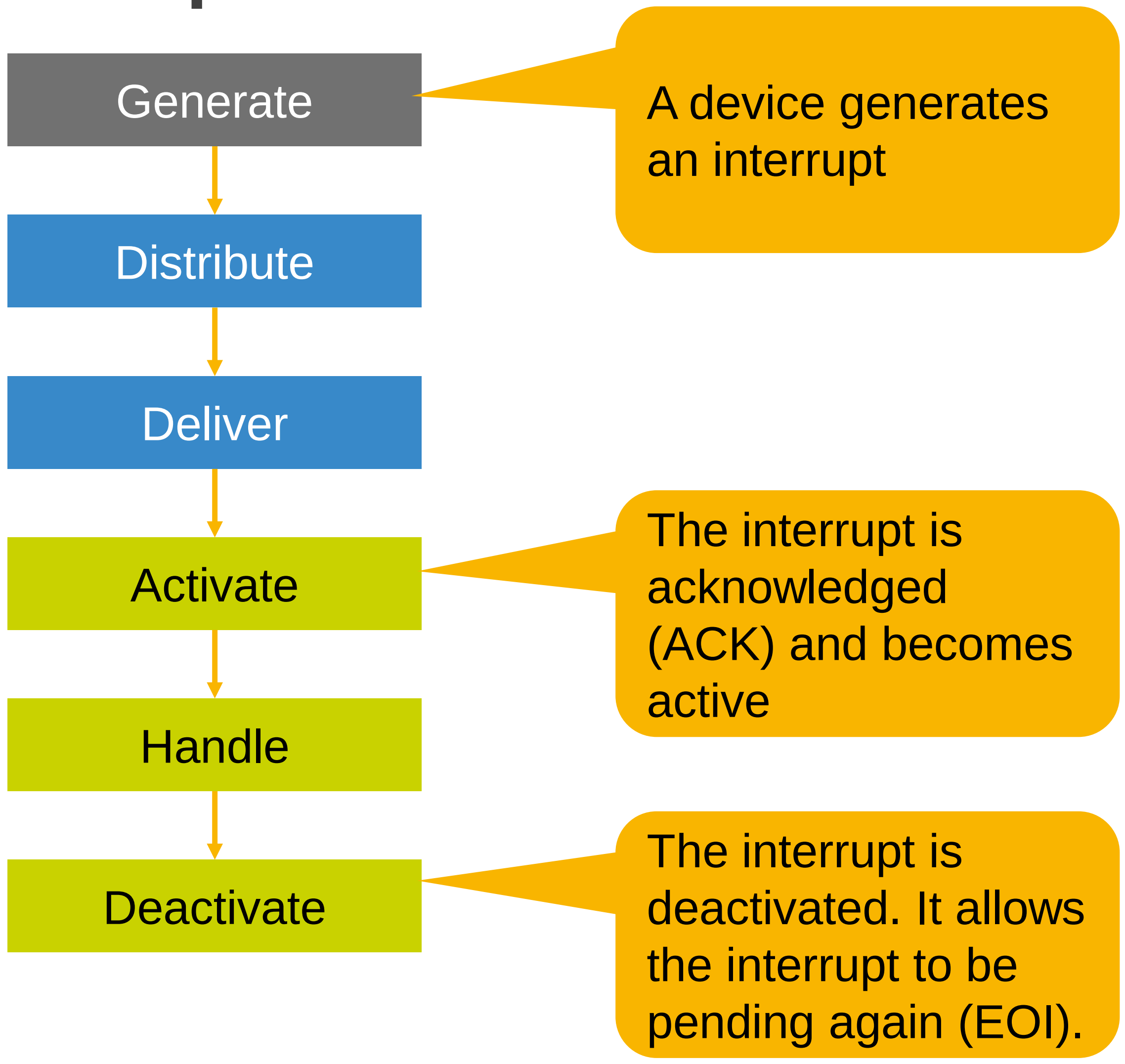


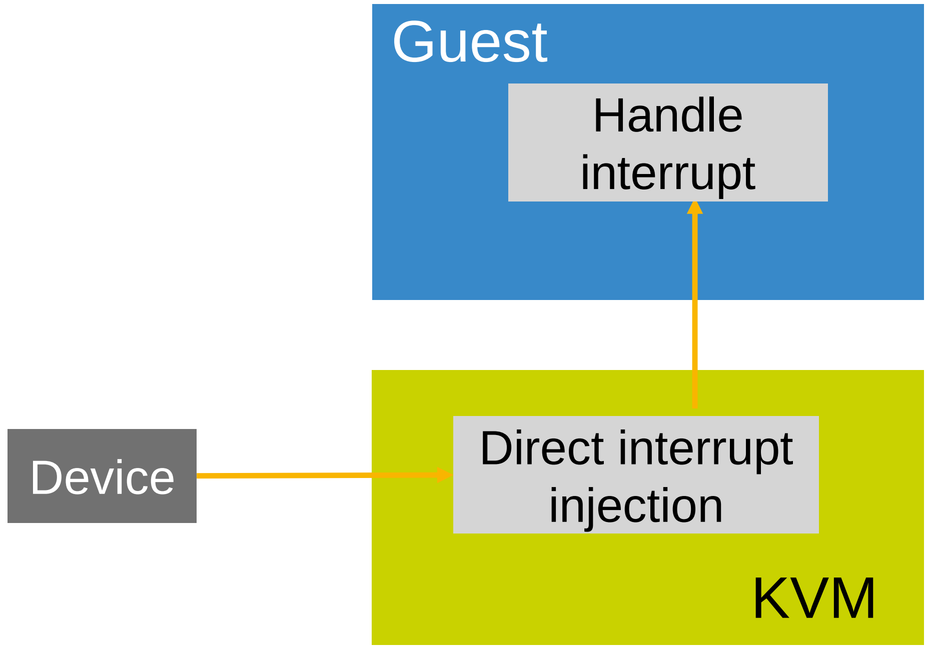


Interrupt Virtualization



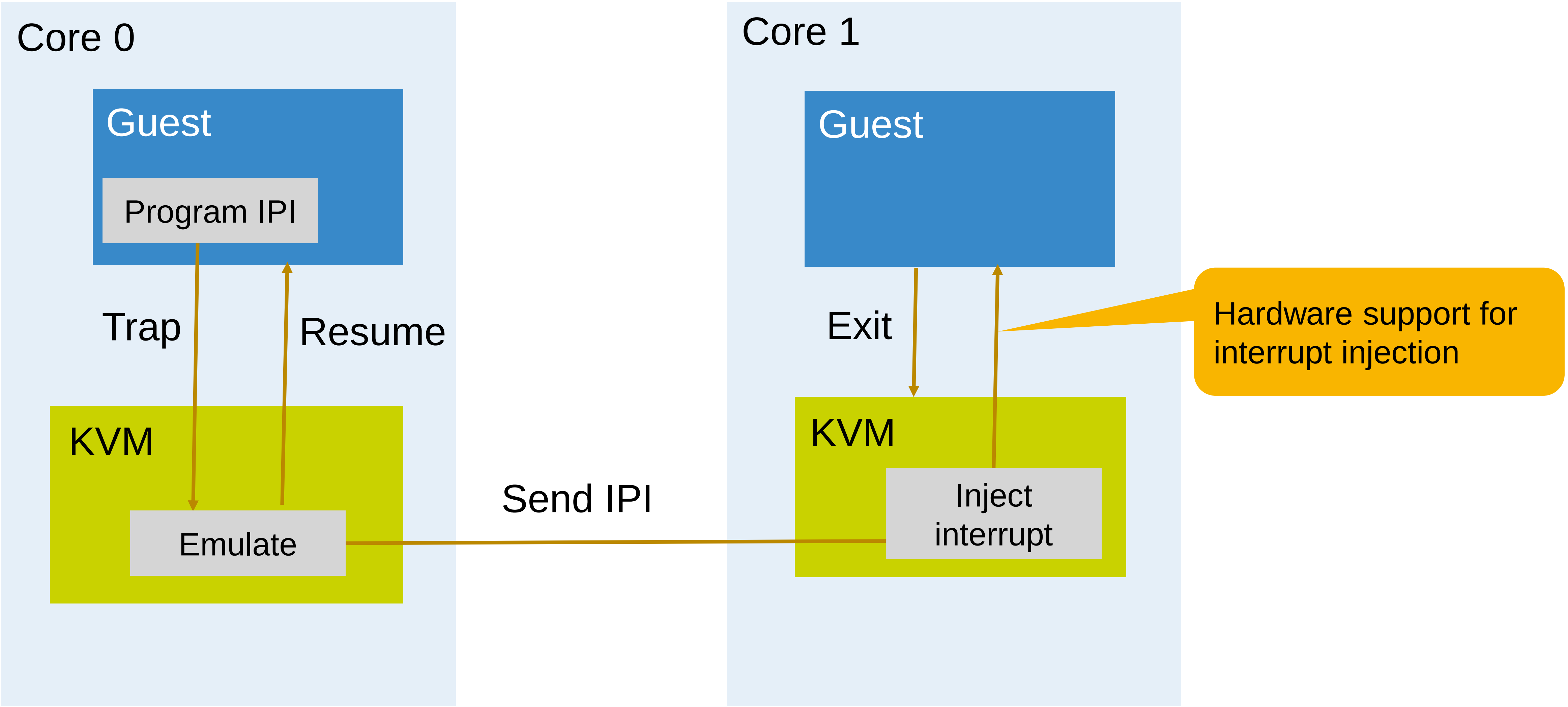
Interrupt Flow





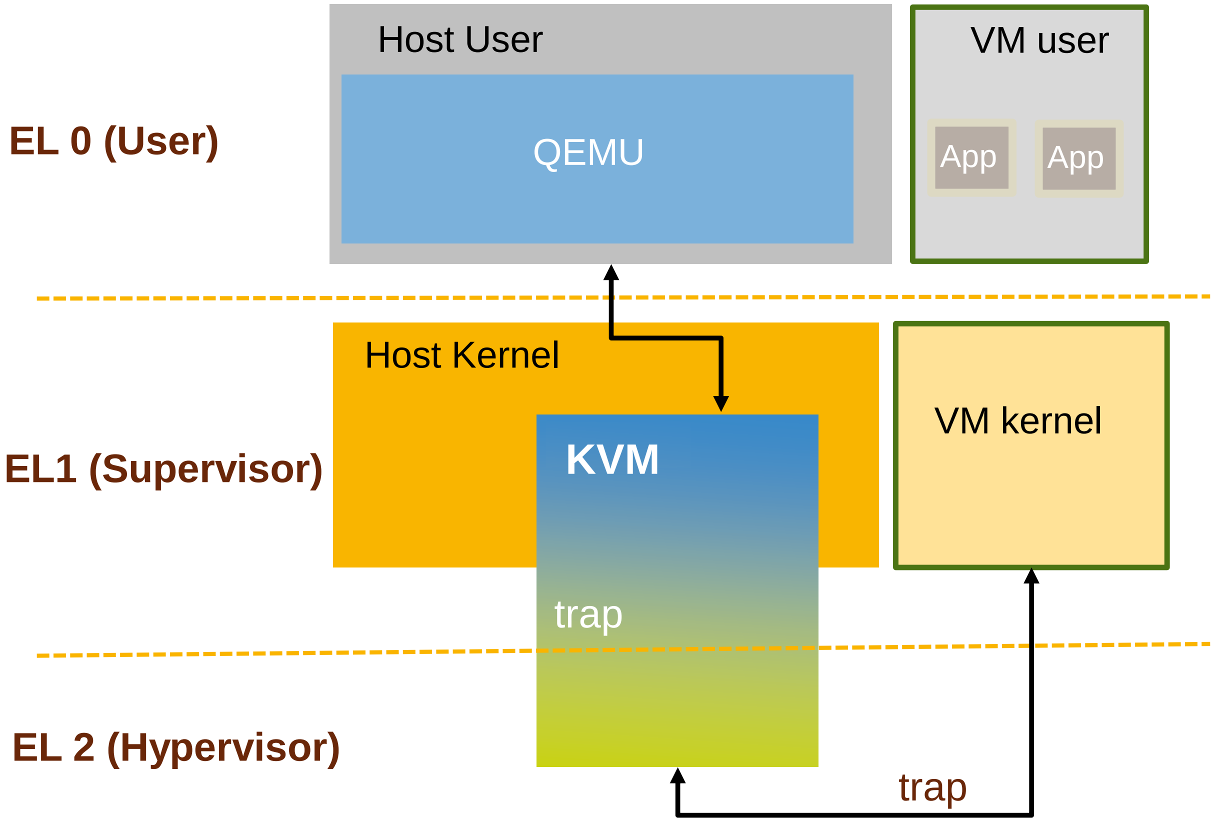
- Guest is interrupted when the interrupt is received
- Hardware support for interrupt injection
- There might be additional exits in certain situations (but they should be rare).

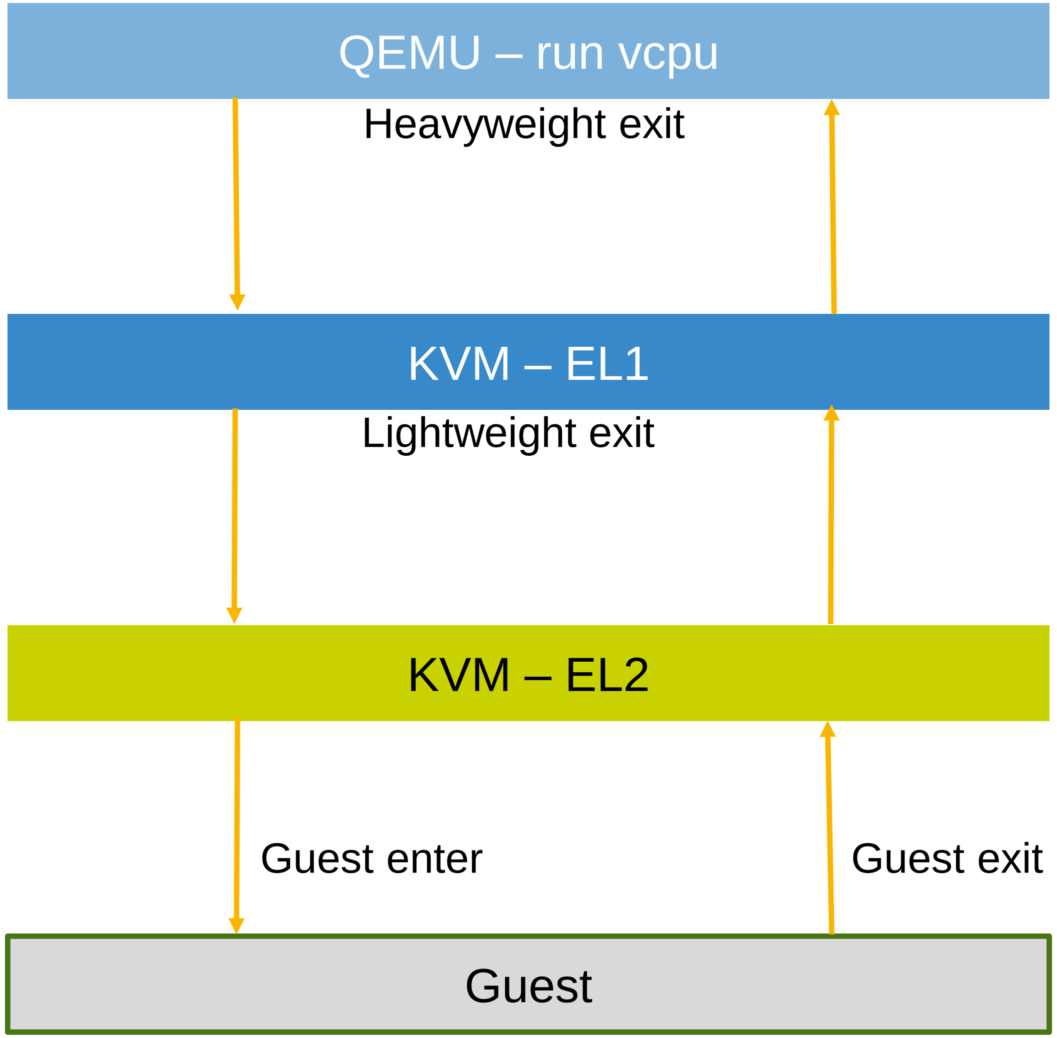
IPI Flow


Core 0
Guest
Program IPI
Trap
Resume
KVM
Emulate
Send IPI
Core 1
Guest
Exit
KVM
Inject interrupt
Hardware support for interrupt injection

KVM ON ARM

KVM/QEMU on ARM





RESULTS

Overhead Sources

- Virtualization may come with a cost: overhead

- But what causes the overhead when we have hardware extensions?

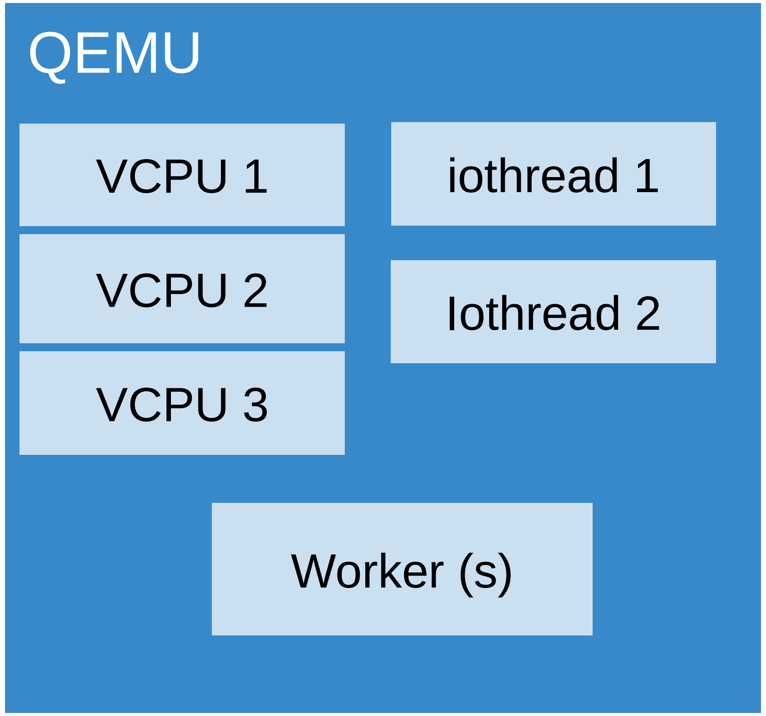
- Overhead due to guest exits
  - Traps, interrupts
- Guest speed
  - More steps in memory translation
  - TLB/cache pollution/contention
  - Lock contention
- Application latency
  - Latency sensitive applications may behave differently in a virtualized environment

Guest Exits – Example

- Exit timing framework
  - For each type of exit reports the time spent in the hypervisor

type	count	min (cycle)	max (cycle)	mean (cycle)	sum (cycle)	std_deviation (cycle)	count	sum
WFX	0	0	0	0	0	0	0	0
CPC15_32	0	0	0	0	0	0	0	0
CPC15_64	0	0	0	0	0	0	0	0
CP14_MR	0	0	0	0	0	0	0	0
CP14_LS	0	0	0	0	0	0	0	0
CP14_64	0	0	0	0	0	0	0	0
HVC32	0	0	0	0	0	0	0	0
SMC32	0	0	0	0	0	0	0	0
HVC64	0	0	0	0	0	0	0	0
SMC64	0	0	0	0	0	0	0	0
SYS64	0	0	0	0	0	0	0	0
IABT_LOW	0	0	0	0	0	0	0	0
DABT_LOW	0	0	0	0	0	0	0	0
DABT_IO_MEM	0	0	0	0	0	0	0	0
DABT_USER_MEM	0	0	0	0	0	0	0	0
DABT_IO_MEM_IPI	157225	10090	57218	13385	2.104.548.054	172	15722,5	210.454.805,40
INTERRUPT	159395	4963	39654	6792	1.082.746.418	226	15939,5	108.274.641,80
TIMEINGUEST	316620	163	356454	47376	15.000.203.563	1230	31662	1.500.020.356,30
DESCHEDULED	2	7036	7036	7036	14072	0	0,2	1.407,20

KVM Benchmark Considerations



?

- VM scaling
- Clustering
- QEMU threads affinity
- CPU scaling
- Idle/busy host
- Reproducibility
- Interrupt affinity

Testing Methodology and Analysis Tools

- Benchmarks
  - Coremark
  - Lmbench
- Analysis tools
  - Exit timing measurements
  - Perf counters (hardware counters)

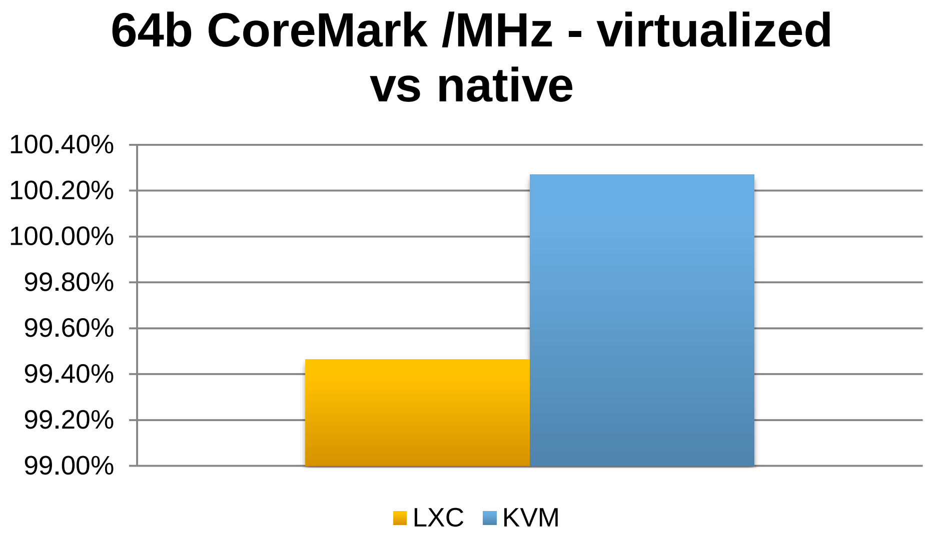
- Platform
  - LS2080 QorIQ hardware

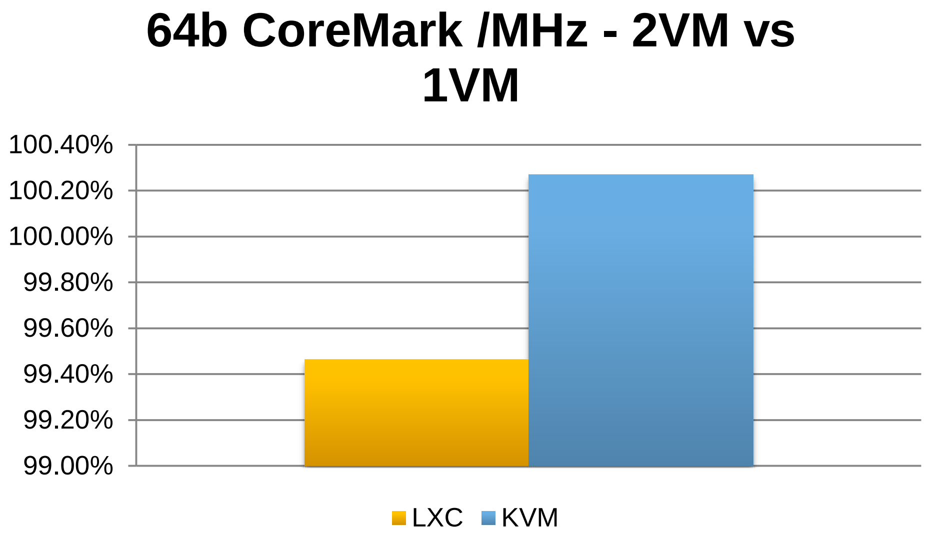
Coremark

- Microbenchmark
- Core centric

Coremark – Results

guest/native [%]

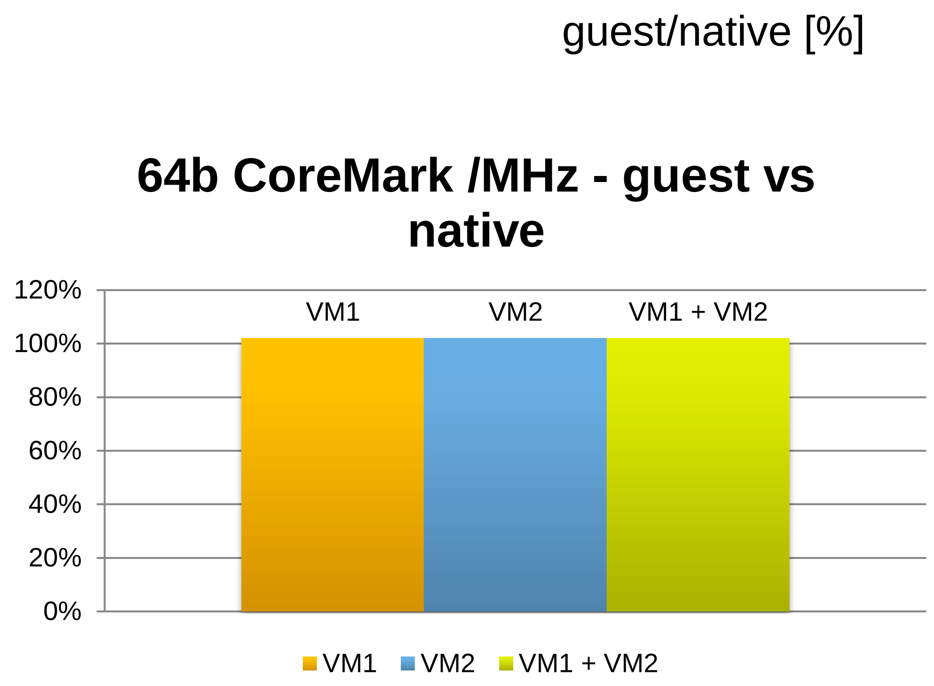




CoreMark – Concurrent VMs

Oversubscription

- Host: 2 CoreMark processes run on the same CPU
- Guest: 2 VMs (VCPUs) running on the same CPUs, each running a CoreMark instance on the same CPU



LMbench

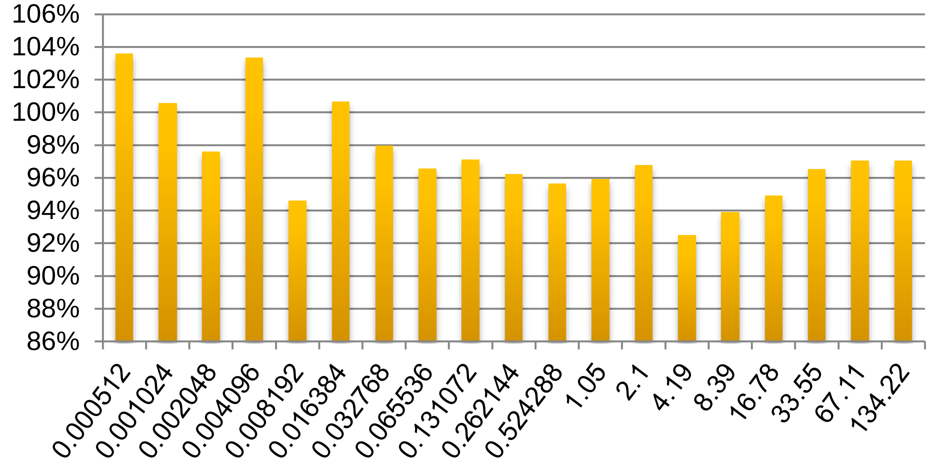
- Synthetic microbenchmark
  - Bandwidth benchmarks
    - Memory bandwidth
    - IPC bandwidth
    - Cached I/O bandwidth

- Latency benchmarks
  - Memory read
  - Signal handling
  - Processes creation
  - Context switch
  - Interprocess communication
  - File system

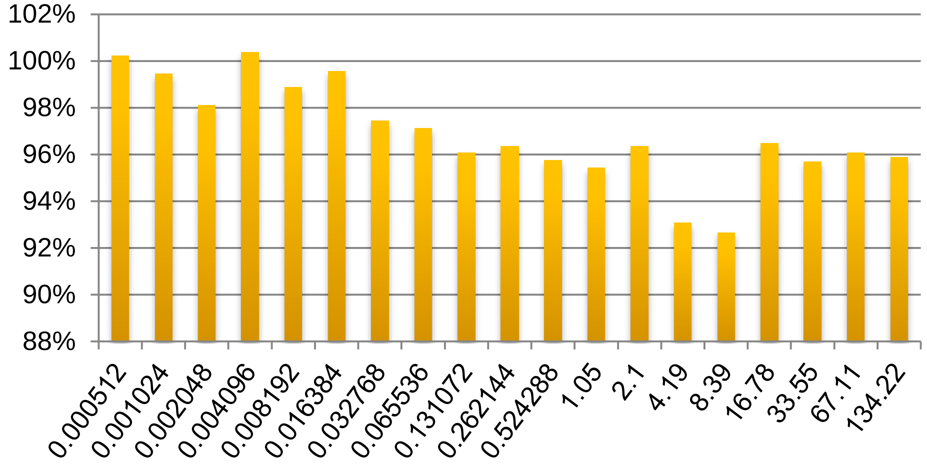
LMBench – Communication Bandwidth

guest/native [%]

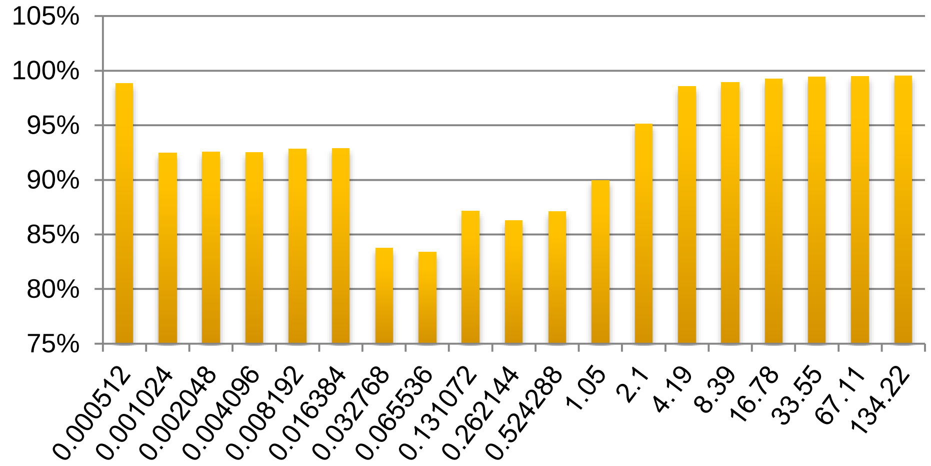
File read bandwidth



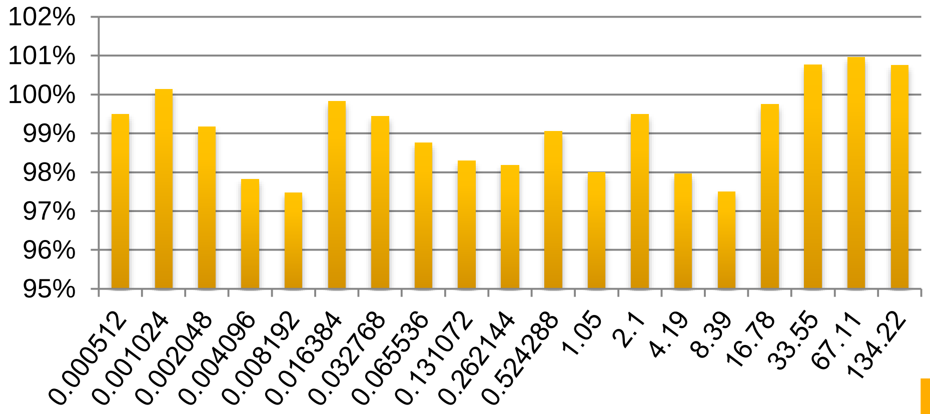
Read open2close bandwidth



Mmap read bandwidth

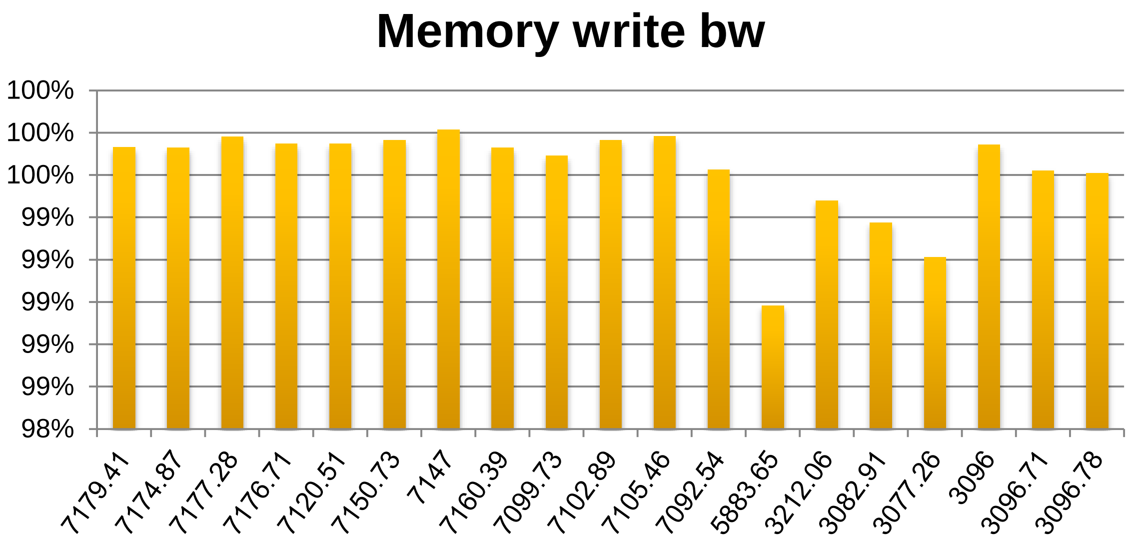


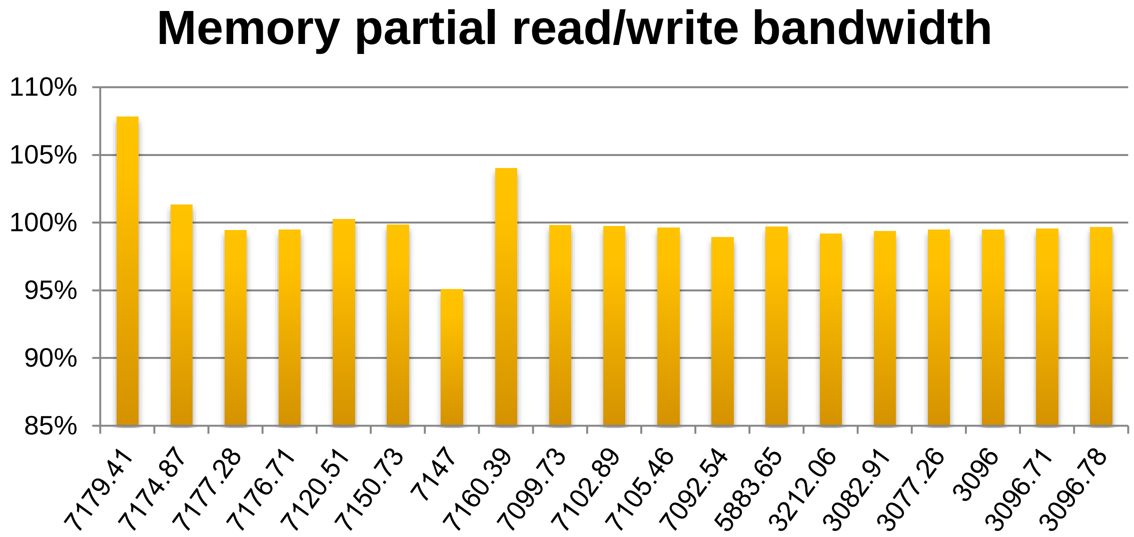
Mmap read open2close bandwidth



LMBench – Memory Bandwidth

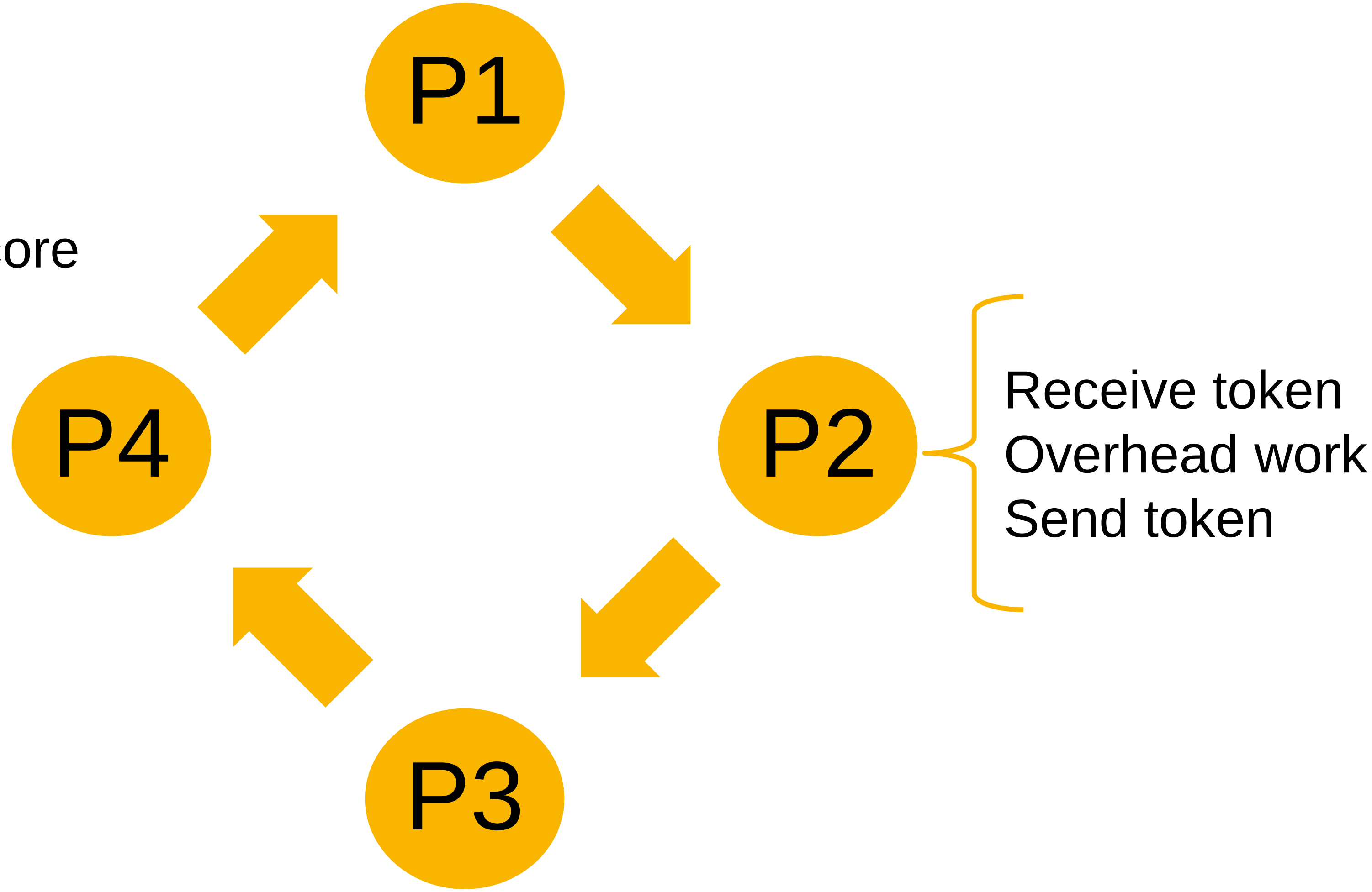
guest/native [%]





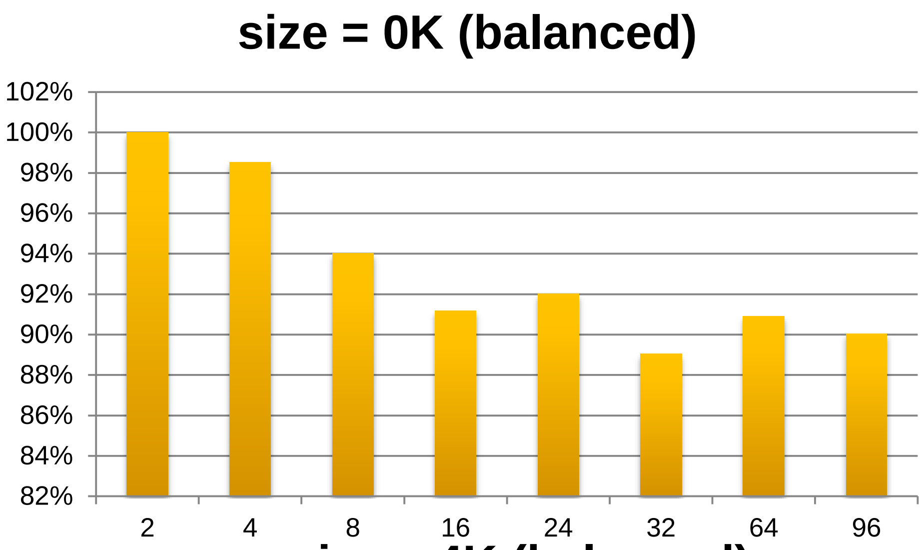
LMBench – Context Switching sub-benchmark

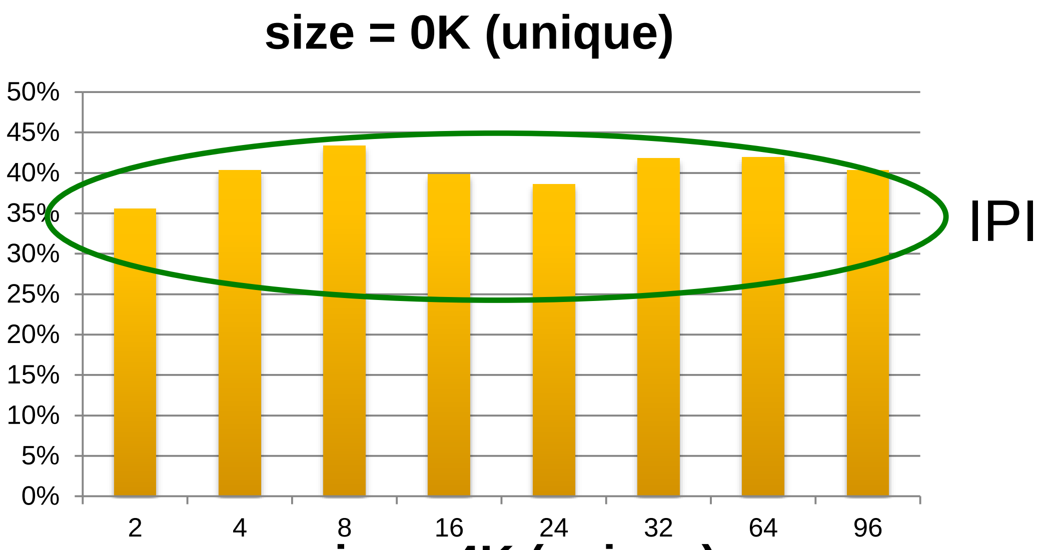
- Balanced
  - All processes on the same core
- Unique
  - Each process is on a different core

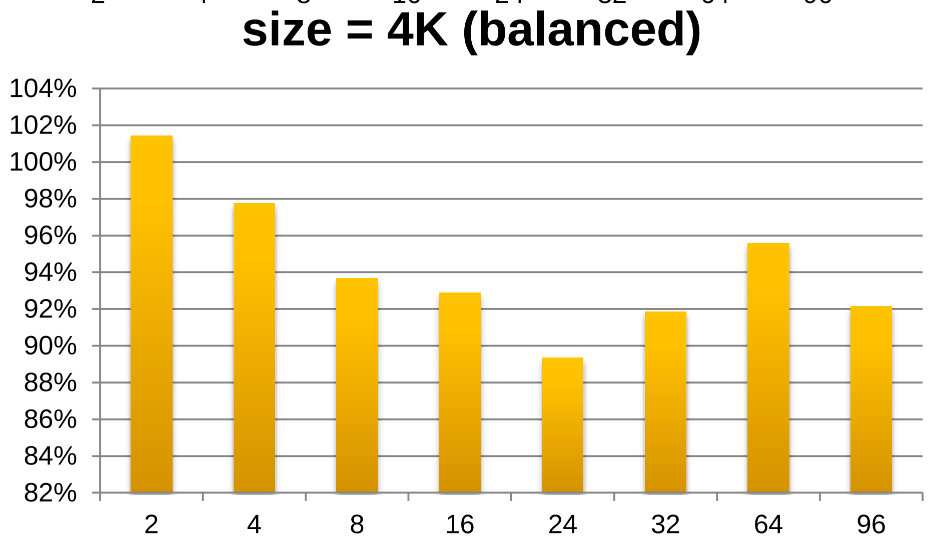


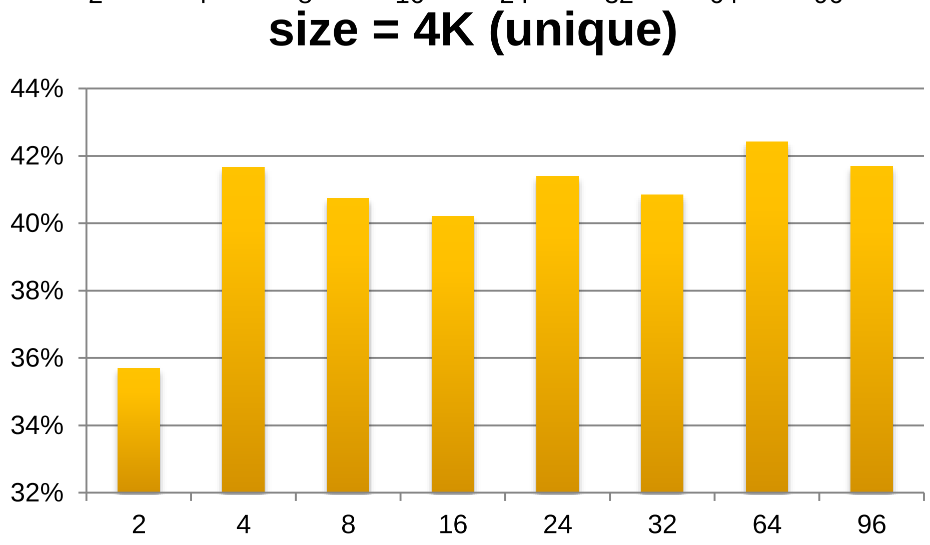
LMBench – Context Switching Latency

guest/native [%]





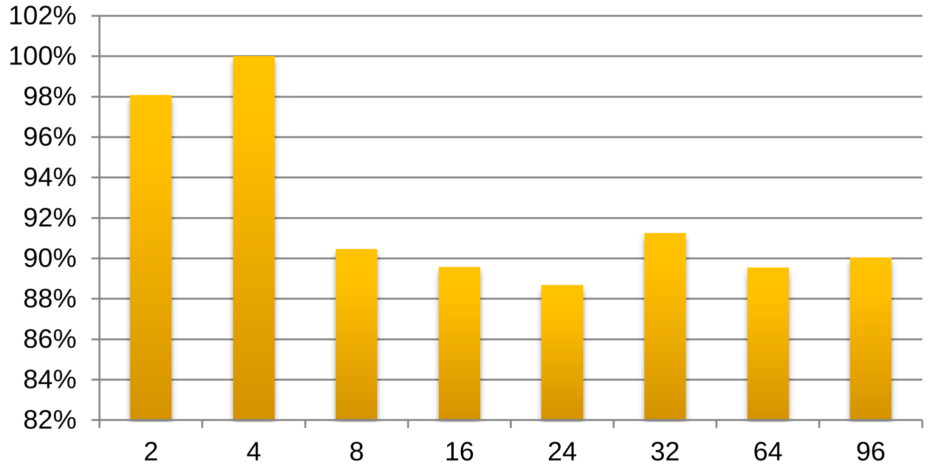




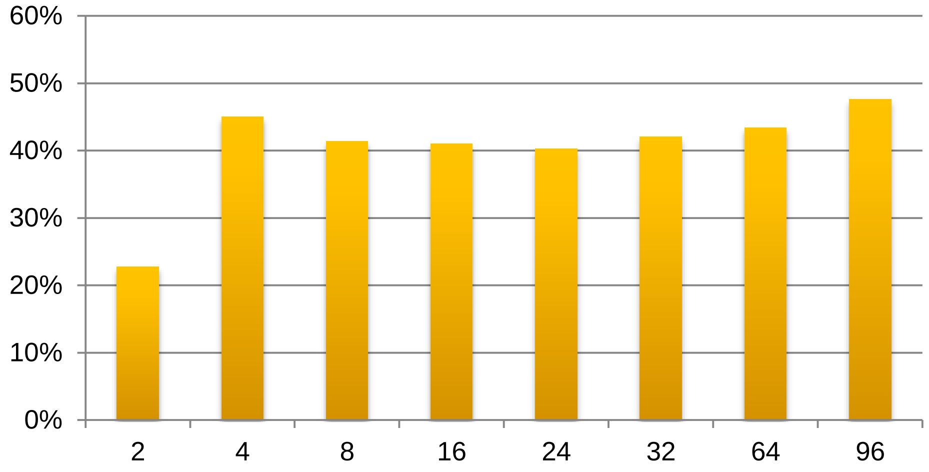
LMBench – Context Switching Latency – Scaling

guest/native [%]

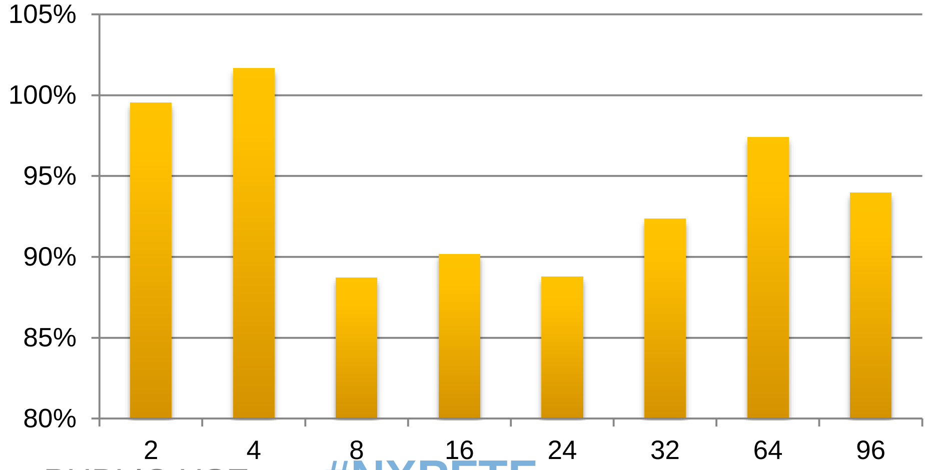
size = 0K (balanced)



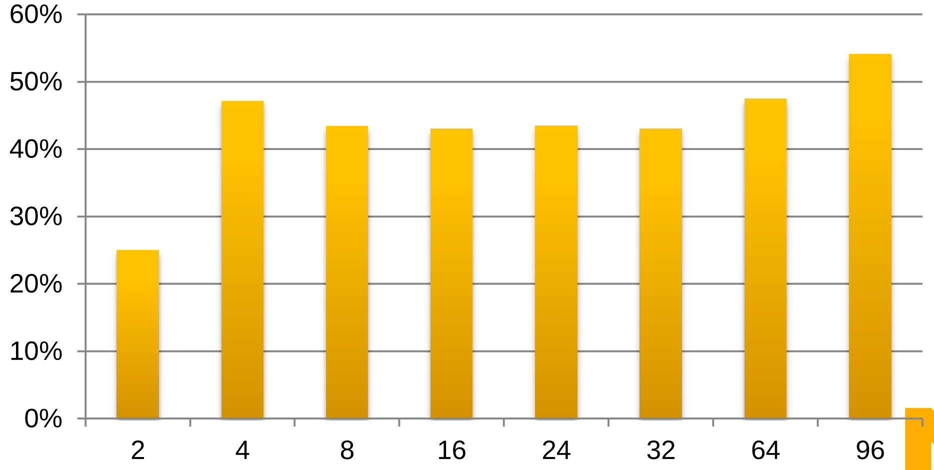
size = 0K (unique)



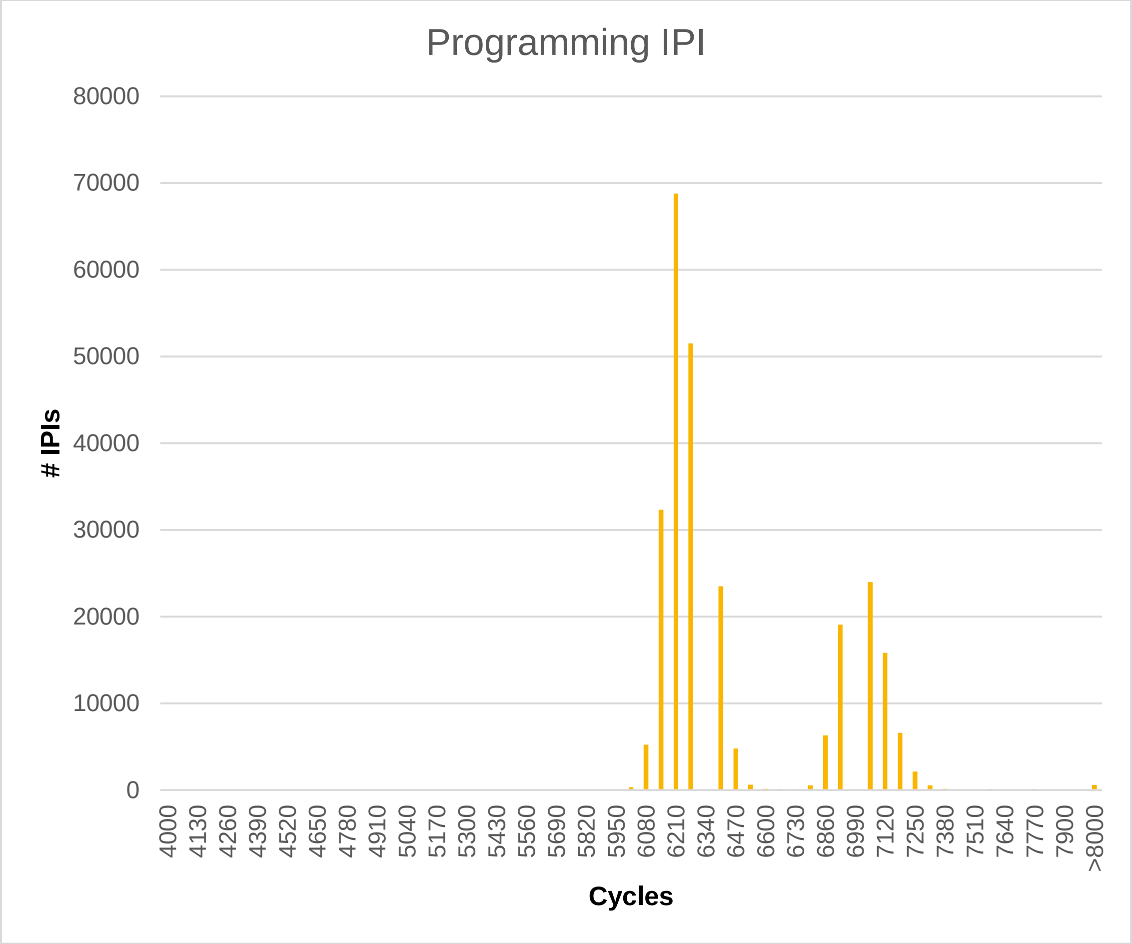
size = 4K (balanced)

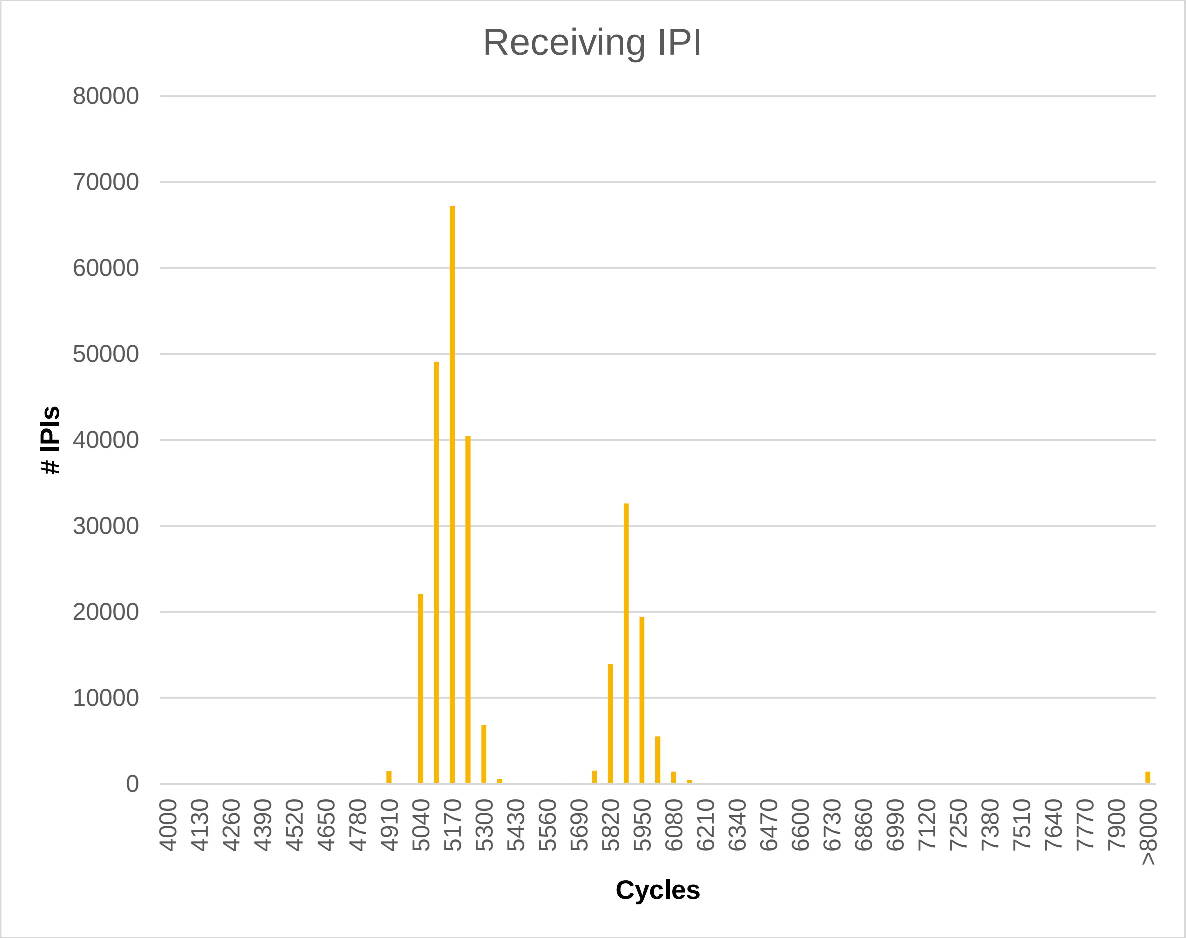


size = 4K (unique)

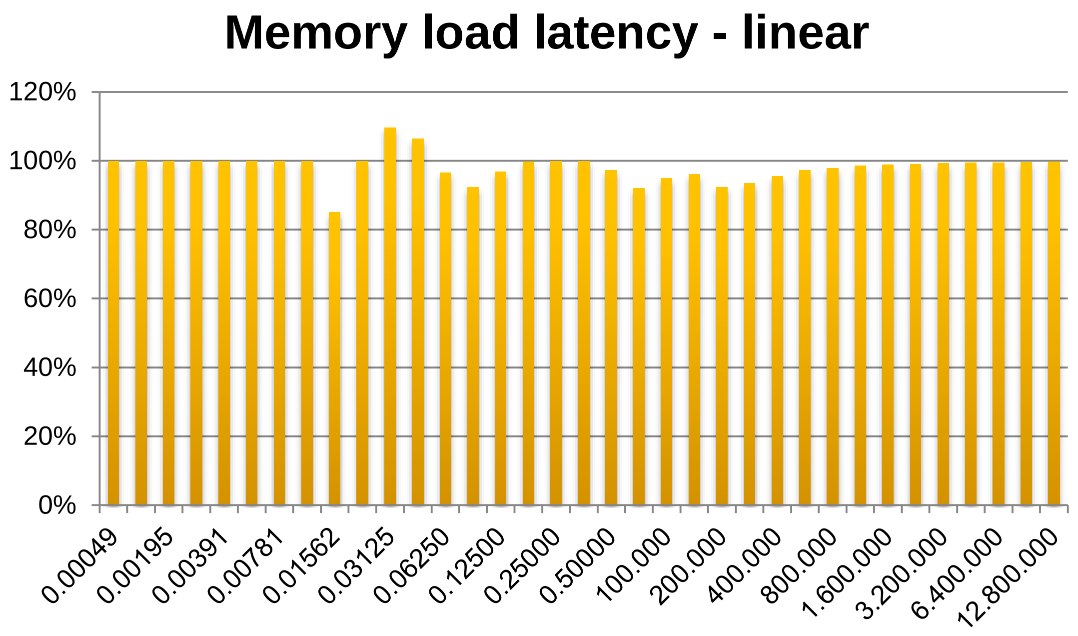


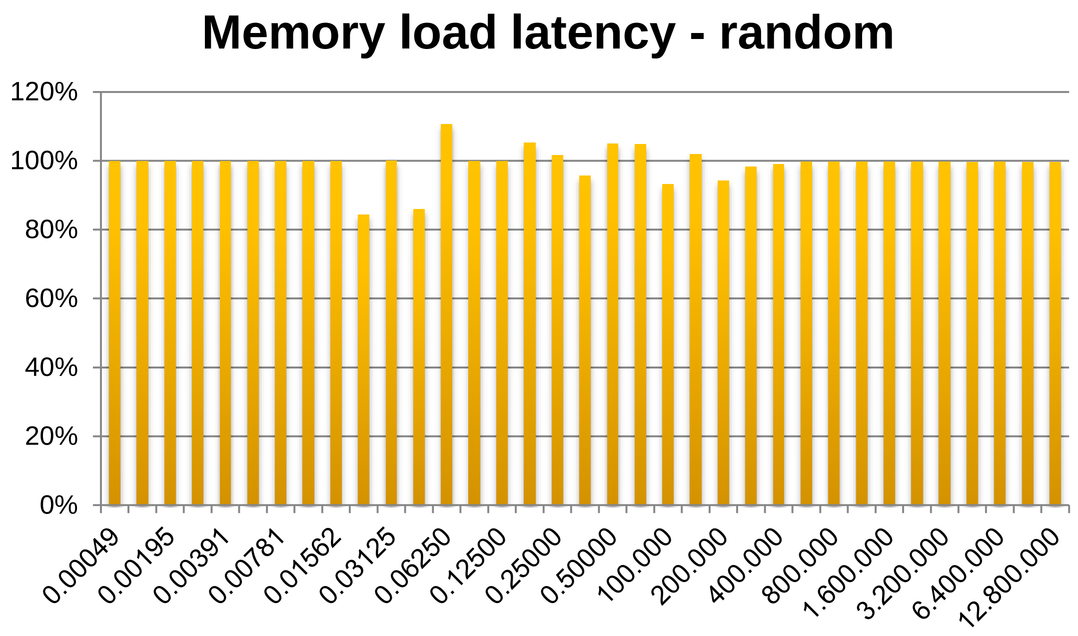
Context Switch Latency Distribution





Memory Load Latency – VM vs. Native





CONCLUSIONS

Conclusions

- For core related benchmarks the performance is good as there are very few exits
- The overhead sources are: guest exits caused especially by IPI and interrupt emulation
  - Performance improved by redesigning the GIC distributor emulation
- Memory related benchmark do not show important overhead in the virtualized environment
  - The number of page table levels does not have a significant impact

SECURE CONNECTIONS
FOR A SMARTER WORLD

ATTRIBUTION STATEMENT