

Intel Technology Goodies

some of which you may find useful for your research...

For discussion at Cloud ISTC Retreat

8 November 2013

Rich Uhlig

Context

- Discussion at last Cloud ISTC Retreat:
 - Can Intel provide access to pre-release hardware?
 - Research prototypes? (hardware or software)
- Today: An overview of possible candidates
- Qualifications
 - We may or may not be able to make these available
 - Some of this may never actually ship as products
 - Discussion useful in any event to identify areas of interest
- Future: Depending on response(s), we'll work a plan...

Scope

- This presentation is not...
 - An Intel hardware roadmap overview
- This presentation is...
 - A look new capabilities that may be “interesting”
 - Where “interesting” depends on your perspective and research goals
- Please interrupt with questions as we go...

Overview

- Research Prototypes

- NVMe Emulator
- Persistent Memory Emulator
- Memory-Race Recorder

- New Technologies

- Transactional Memory (TSX)
- New VT Features
- Shared Virtual Memory (SVM)
- Security Guard Extensions (SGX)

- New Platforms

- Xeon Phi
- μ Cluster
- Quark

- Software

- Persistent Mem Filesystem
- Data Plane Devel Kit (DPDK)

Emerging Memory Technologies

- PCM, STT-RAM, Memristors
- Non-volatile like NAND, but with much higher performance and endurance properties
- Approach DRAM performance, but with higher storage density
- Asymmetric Read/Write Perf
 - Reads within ~2-3X of DRAM
 - Writes within ~5-10X of DRAM
 - Fine-grained, random access possible

Table 1: Comparison of Memory Technologies

Parameter	DRAM	NAND Flash	NOR Flash	PCM
Density	1X	4X	0.25X	2X-4X
Read Latency	60ns	25 us	300 ns	200-300 ns
Write Speed	≈1 Gbps	2.4 MB/s	0.5 MB/s	≈100 MB/s
Endurance	N/A	10 ⁴	10 ⁴	10 ⁶ to 10 ⁸
Retention	Refresh	10yrs	10yrs	10 yrs

Qureshi, et al. [ISCA'09]

Memory	Real read latency	Real write latency	Performance simulation
DRAM	10ns	10ns	10ns
PCRAM	20ns	100ns	100ns
STTRAM	10ns	20ns	20ns
MRAM	12ns	12ns	12ns

Table IV: Memory access latencies of different memory systems

Li, et al. [IPDPS'12]

“Persistent Memory” Defined

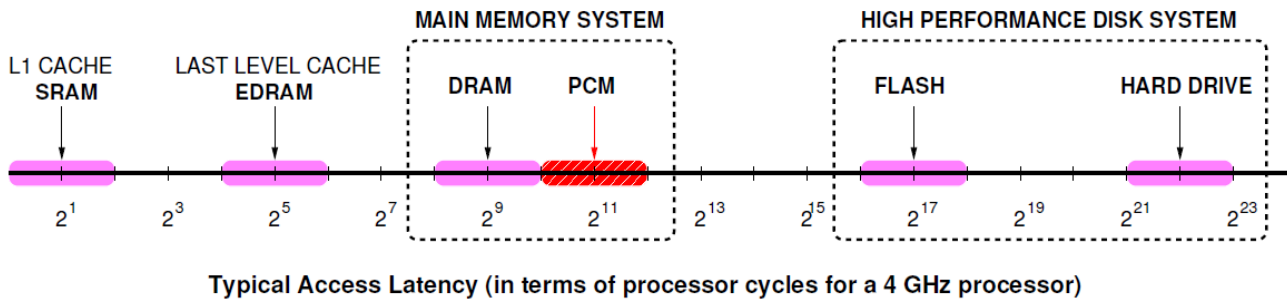


Figure from:
Qureshi, et al.
[ISCA'09]

- New NVMs are so close to DRAM, that it's tempting to:
 - Integrate them into the *cache-memory* hierarchy
 - Access them directly via CPU load/store instructions
 - Expose their properties of *persistence* to software
 - Then wait for new apps and general goodness to follow...

But are we getting ahead of ourselves?

An Alternative Approach...

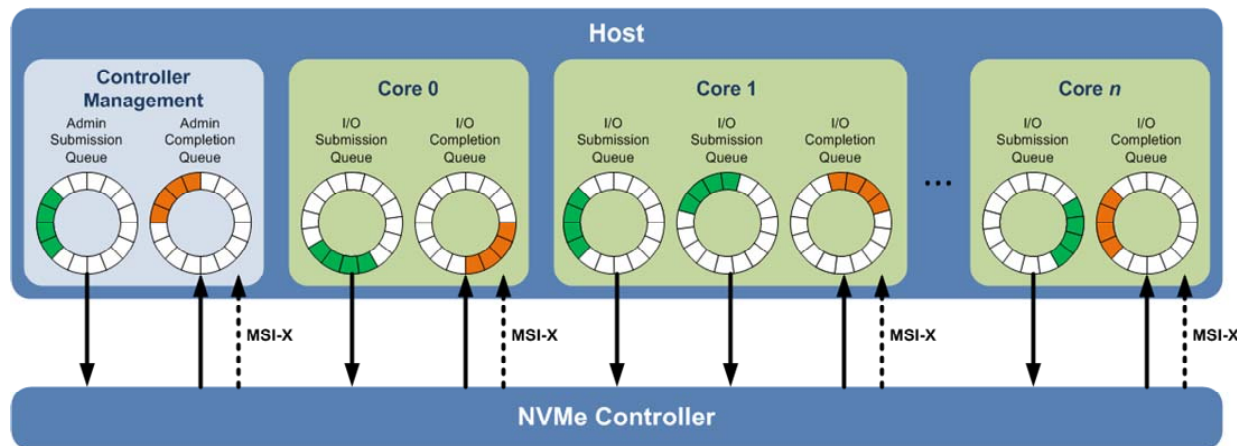
- Evolve traditional block-oriented storage interfaces
 - Transition from SAS/SATA to PCIe attach
 - Upgrade the storage controller to support high IOPS
 - Tune storage-driver stack to take out latency
- Leverage other platform advances as we go...
 - Interrupt-architecture improvements (directed MSIs, etc.)
 - Direct Cache Access (a.k.a., Direct I/O)
 - Multi-core CPUs
- Example: NVM Express (NVMe)



From Intel Developers Forum...*

NVMe Technical Basics

- The focus of the effort is efficiency, scalability and performance
 - All parameters for 4KB command in single 64B DMA fetch
 - Supports deep queues (64K commands per Q, up to 64K queues)
 - Supports MSI-X and interrupt steering
 - Streamlined command set optimized for NVM (6 I/O commands)
 - NVM technology agnostic



* Foil courtesy
Amber Huffman

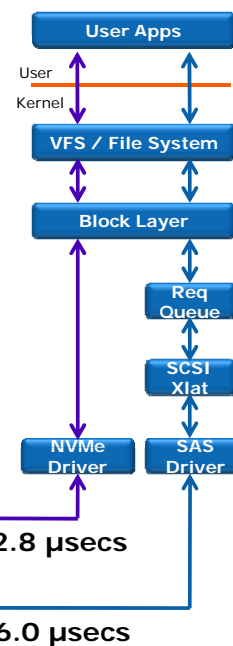
From Intel Developers Forum (2)*

Proof Point: NVMe Latency

- NVMe reduces latency overhead by **more than 50%**
 - SCSI/SAS: 6.0 μ s 19,500 cycles
 - NVMe: 2.8 μ s 9,100 cycles**
- NVMe is designed to scale over the next decade
 - NVMe supports future NVM technology developments that will drive latency overhead below one microsecond
- Example of latency impact: Amazon* loses 1% of sales for every 100 ms it takes for the site to load



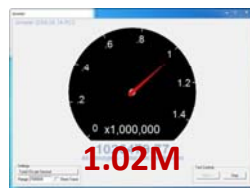
Linux* Storage Stack



Chatham NVMe Prototype



Prototype Measured IOPS



Cores Used for 1M IOPS

2.8 μ secs
6.0 μ secs



* Foil courtesy Amber Huffman

8 Measurement taken on Intel® Core™ i5-2500K 3.3GHz 6MB L3 Cache Quad-Core Desktop Processor using Linux RedHat® EL6.0 2.6.32-71 Kernel.

Summary: NVMe as a Baseline

- Latency
 - Expect to scale SW latency to below 1.0 μ s over time
 - Targeting end-to-end latency of < 5.0 μ s
- Throughput
 - Expect to scale up >> 1M IOPS with multi-core
 - Demonstrated to 2M IOPS with NVMe prototype

Any proposals for Persistent Memory should be compared against this baseline

Persistent-Memory Advantages

- Integration into the CPU cache-memory hierarchy
 - Benefit from prefetching, out-of-order execution, write buffering, etc.
- Fine-grained Accesses
 - Through standard ISA load/store instructions from user or kernel mode
 - Not “byte addressable”, but still finer grained than a block-storage interface
- Bandwidth
 - More bandwidth headroom via DDR channels into CPU socket
 - Example: ~9 GB/s per channel, up to 4 channels per socket (in Ivytown)
- Latency
 - Read latency can get into the low nsec range (depending on cache residency)
 - Can decouple “write” into memory from durability requirements (w/ SW help)

Software Research Questions

Applications

- Which particular apps does PM benefit?
- Are any entirely new usages enabled by PM?

Language / Runtime

- Durability model and recoverability?
- Concurrency model and sharing?
- Debugging: Pointer safety, memory leaks, etc.?

Filesystem / Object Store

- Can we collapse filesystems and virtual mem?
- What's the data model for persistent objects?

OS / VMM

- Should we rethink the process model?
- How should virtual memory change?
- How can PM be virtualized to guest OSES?

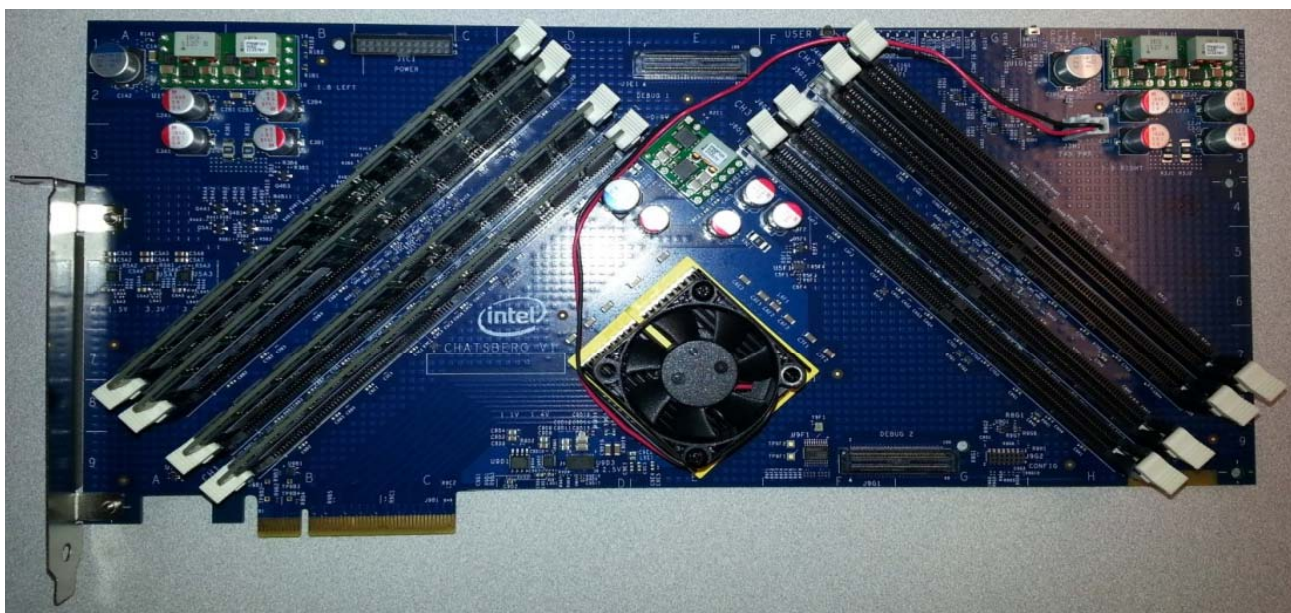
Hardware / ISA Interface

- Mechanisms for expressing write durability?
- Protection against stray writes?

But how to explore the software-design space?

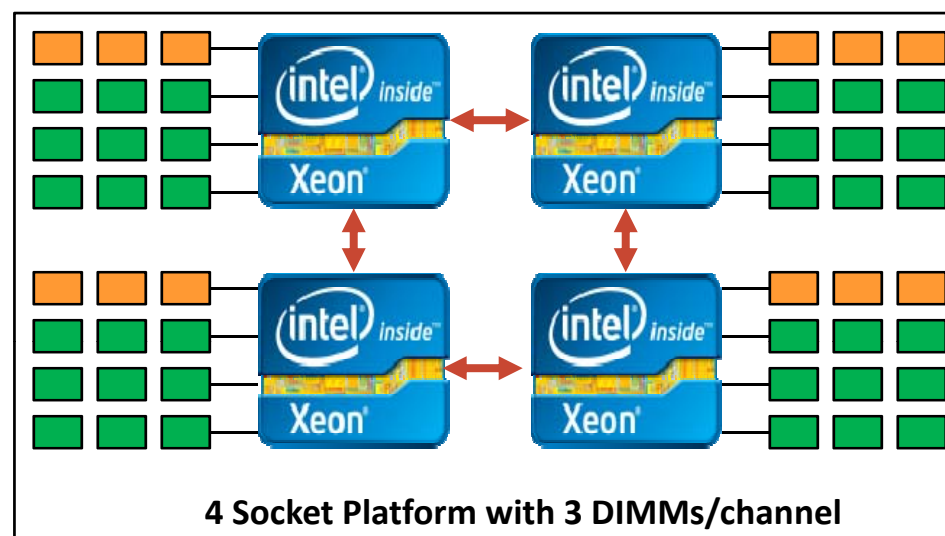
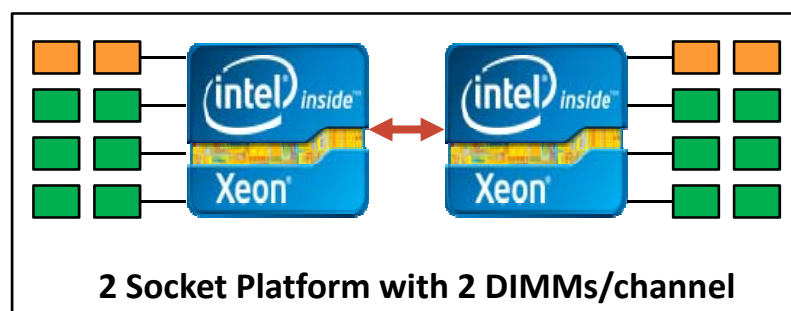
And how to compare block-oriented versus memory load-store interfaces?

① NVMe Storage Prototype (Chatham)



- PCI-Express Gen2 x8 add-in card with on-board DRAM
 - Supports NVMe controller interface
 - Up to 32GB capacity per card (multiple cards/platform possible)
 - NVM-like performance tunable (in FPGA) up to DDR3 speeds
 - Open-source Linux NVMe driver supporting 512B blocks

② Persistent-Memory Emulator



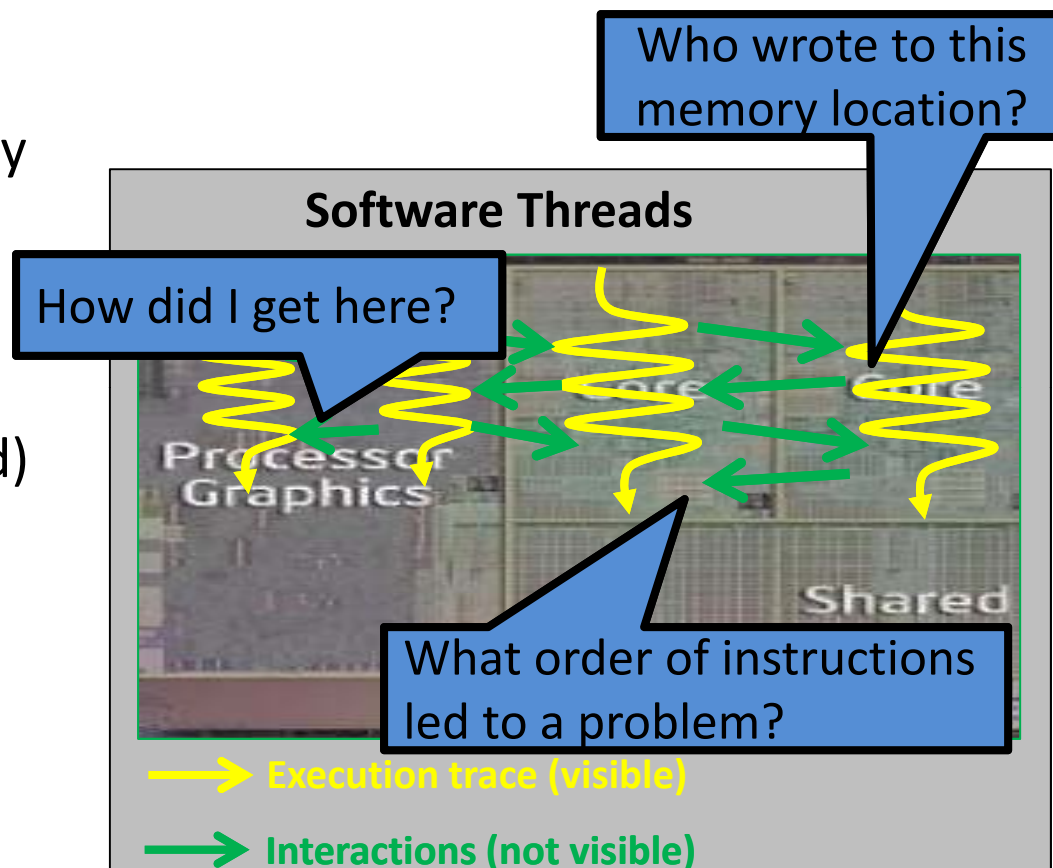
- Standard 2S/4S Xeon server with memory modifications
 - BIOS partitions DIMMs into volatile vs. (emulated) persistent memory
 - Configurable latency/BW for CPU accesses to persistent memory
 - Up to 384GB persistent memory capacity for 2S (1TB for 4S)
 - Works with a prototype Persistent Memory Filesystem (PMFS)

Parallel Program Debug

- Multicore software ecosystem is growing:
 - Multi-core in all segments (server, client, mobile)
 - SoCs bringing CPU, GPU and IP cores together
- But software debug tools remain primitive:
 - Code/Data Watch-points
 - Branch Trace Store (expect ~20x slowdown)
 - Last Branch Record (limited to last 16 branches)

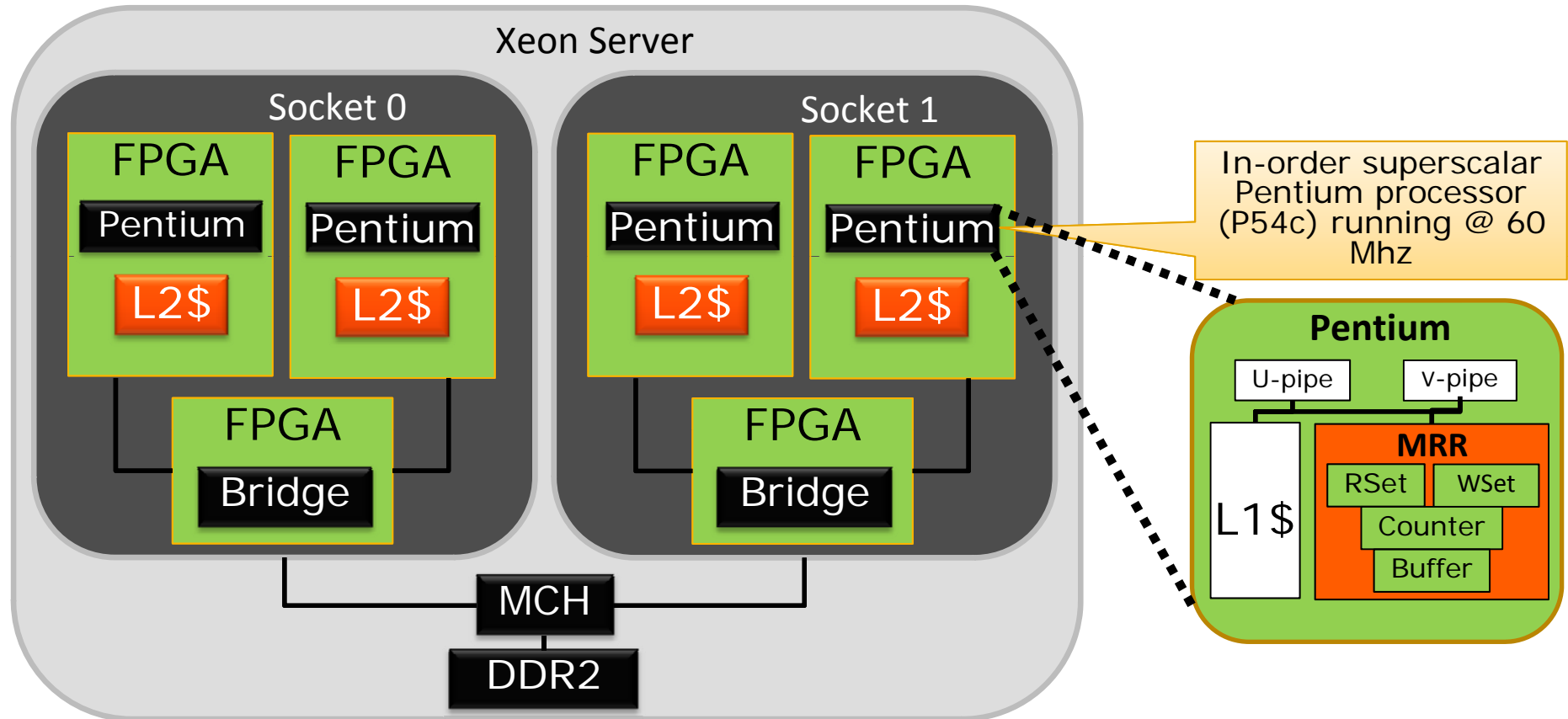
③ Memory-Race Recording

- A HW feature that captures the order of shared-memory accesses across threads or processes running on a multicore IA
- Allows micro-scale (detailed) analysis of parallel SW on multi-core systems
- Supports multi-threaded execution replay for debugging

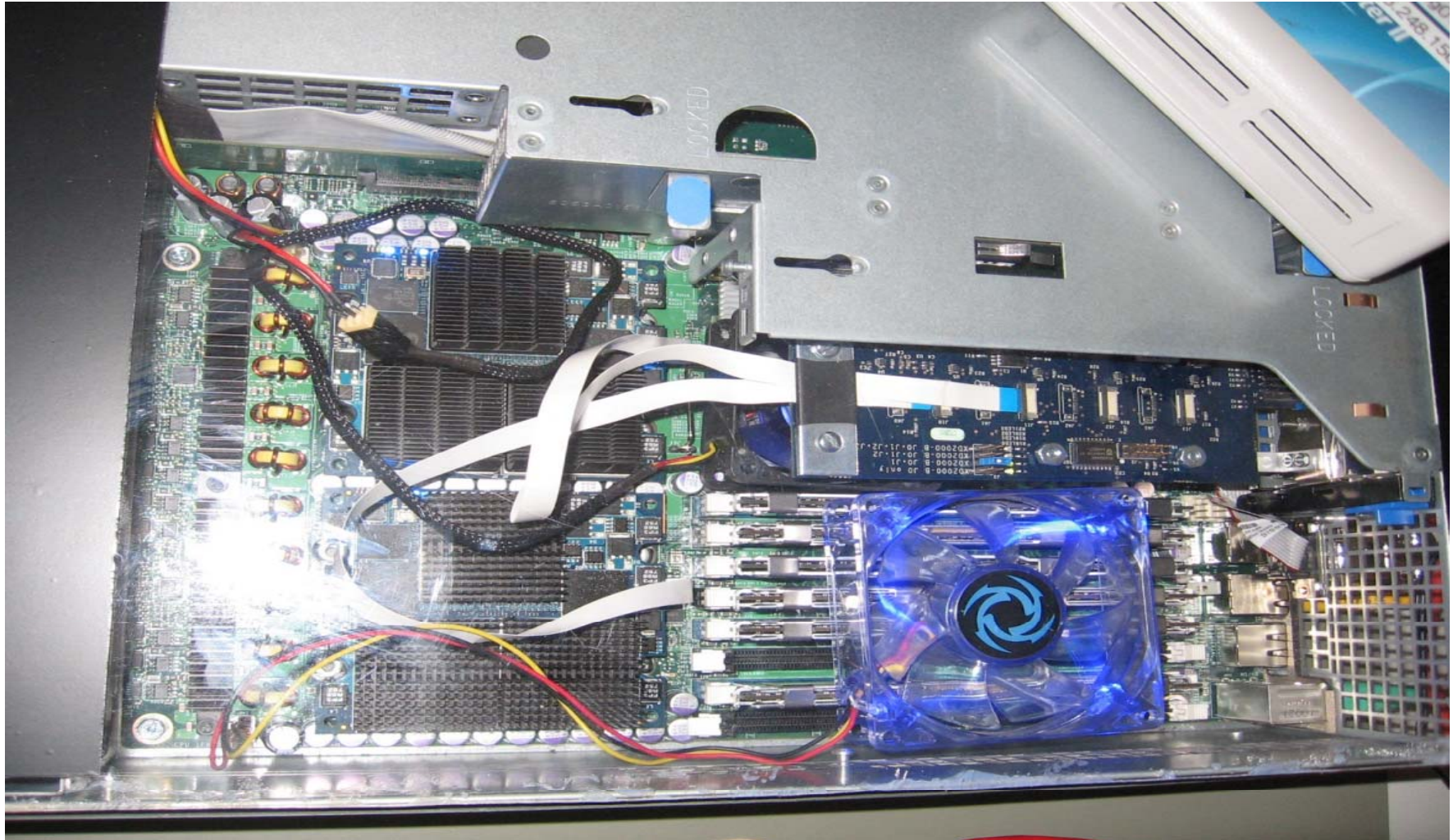


Intel Contact: Gilles Pokem

QuickRec MRR prototype



QuickRec MRR prototype



Research Prototypes: Status

- Both NVMe and PM emulator available in vLab
 - Hosted at Intel facility with remote login possible
 - Useful for “Specialization” & NVM research pillar?
- QuickRec MRR prototype functional
 - Tools for replaying/reconstructing traces working
 - Not yet externally accessible
 - Useful for “Problem Diagnosis” research?

Overview

- Research Prototypes

- NVMe Emulator
- Persistent Memory Emulator
- Memory-Race Recorder

- New Technologies

- Transactional Memory (TSX)
- New VT Features
- Shared Virtual Memory (SVM)
- Security Guard Extensions (SGX)

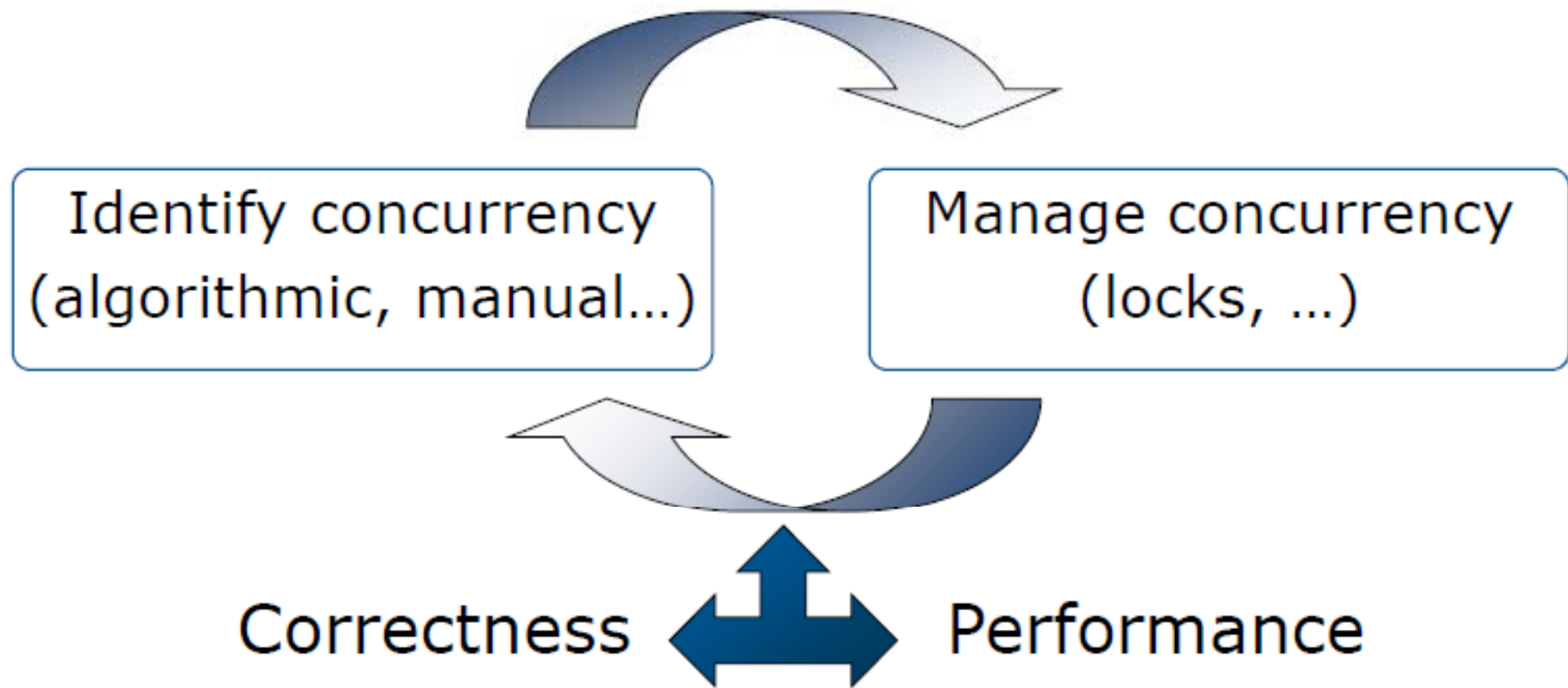
- New Platforms

- Xeon Phi
- Micro-cluster
- Quark

- Software

- Persistent Mem Filesystem
- Data Plane Devel Kit (DPDK)

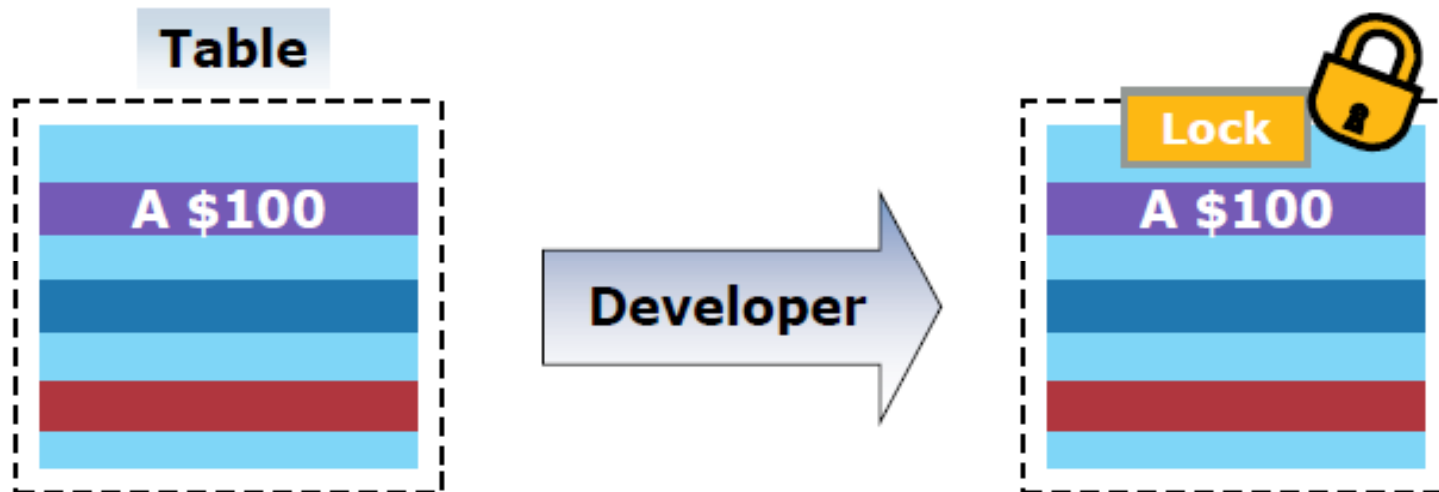
Difficulty of Software Development



Hard to Write Fast and Correct Multi-Threaded Code

From IDF 2012: Rajwar and Dixon

The Need for Synchronization



Alice wants \$50 from A

- A was \$100, A is now \$50

Bob wants \$60 from A

- A was \$100, A is now \$40

A should be -10

Alice wants \$50 from A

- **Alice locks table**

• A was \$100, A is now \$50

Bob wants \$60 from A

- **Bob waits till lock release**

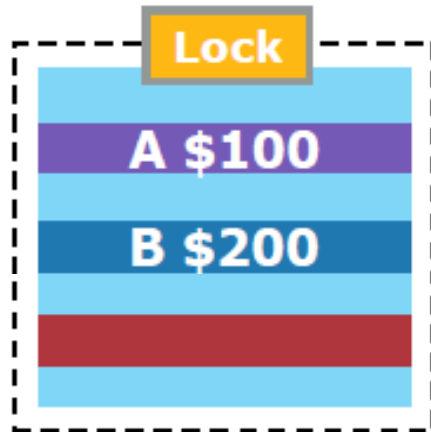
- A was \$50, Insufficient funds

Bob and Alice saw A as \$100. Locks prevent such data races

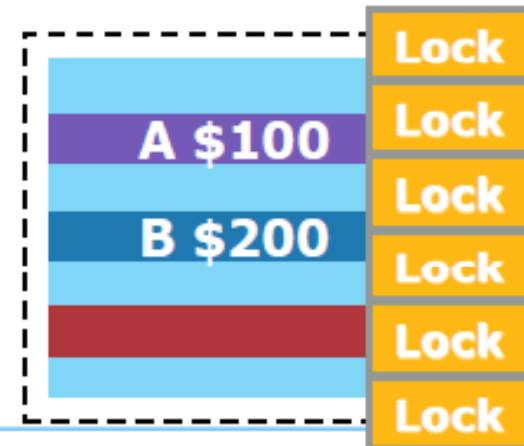
From IDF 2012: Rajwar and Dixon

Lock Granularity Optimization

Coarse Grain Locking (lock per table)



Fine Grain Locking (lock per entry)



Alice withdraws \$20 from A

- Alice locks table

Bob wants \$30 from B

- Waits for Alice to free table

Alice withdraws \$20 from A

- Alice locks A

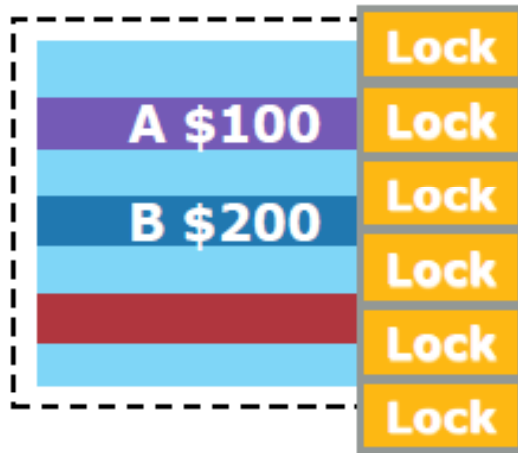
Bob wants \$30 from B

- Bob locks B

Such Tuning is Time Consuming and Error Prone

From IDF 2012: Rajwar and Dixon

Complexity of Fine Grain Locking



Alice transfers \$20 from A to B

- Alice locks A and locks B
- Performs transfer
- Alice unlocks A and unlocks B

Alice transfers \$20 from A to B

Locks A

Cannot lock B

Bob transfers \$50 from B to A

Locks B

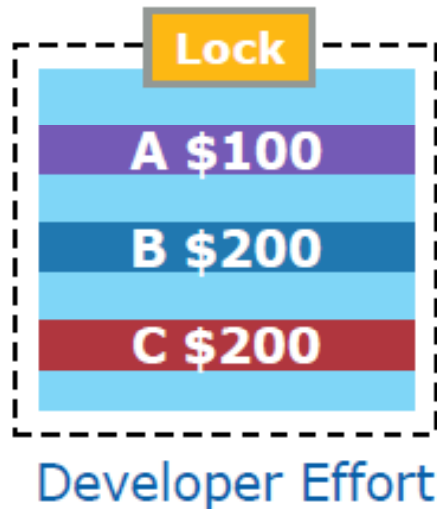
Cannot lock A

Expensive and Difficult to Debug Millions of Lines of Code

From IDF 2012: Rajwar and Dixon

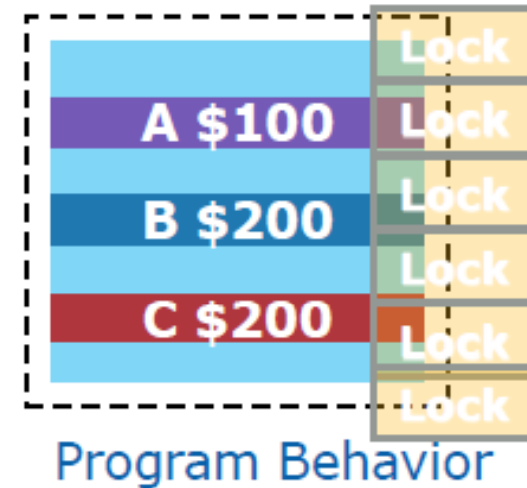
What We Really Want...

Coarse grain locking effort



Hardware

Fine grain locking behavior

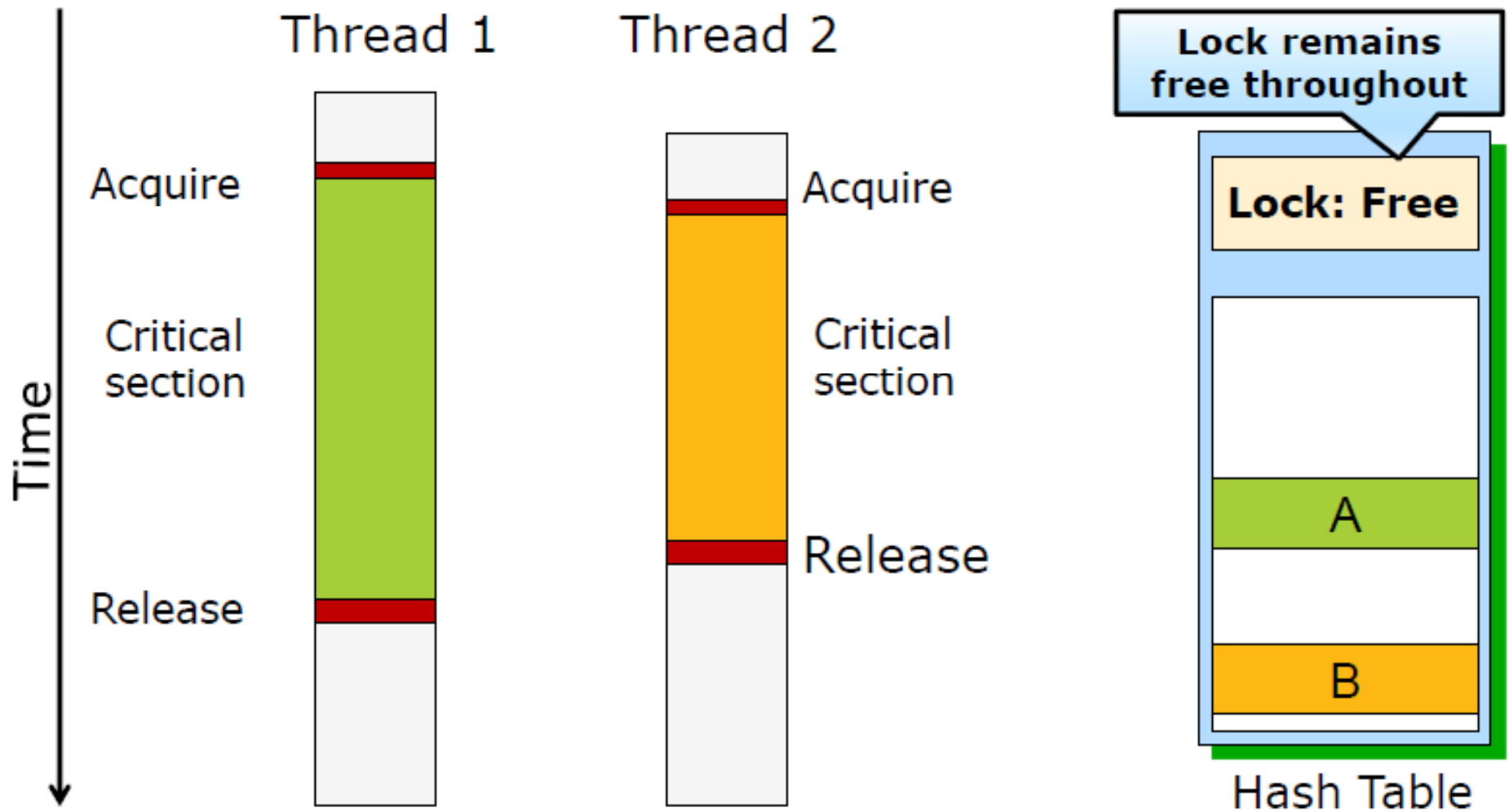


- Developer uses coarse grain lock
- Hardware elides the lock to expose concurrency
 - Alice and Bob don't serialize on the lock
 - Hardware automatically detects real data conflicts

Lock Elision: Fine Grain Behavior at Coarse Grain Effort

From IDF 2012: Rajwar and Dixon

A Canonical Intel® TSX Execution



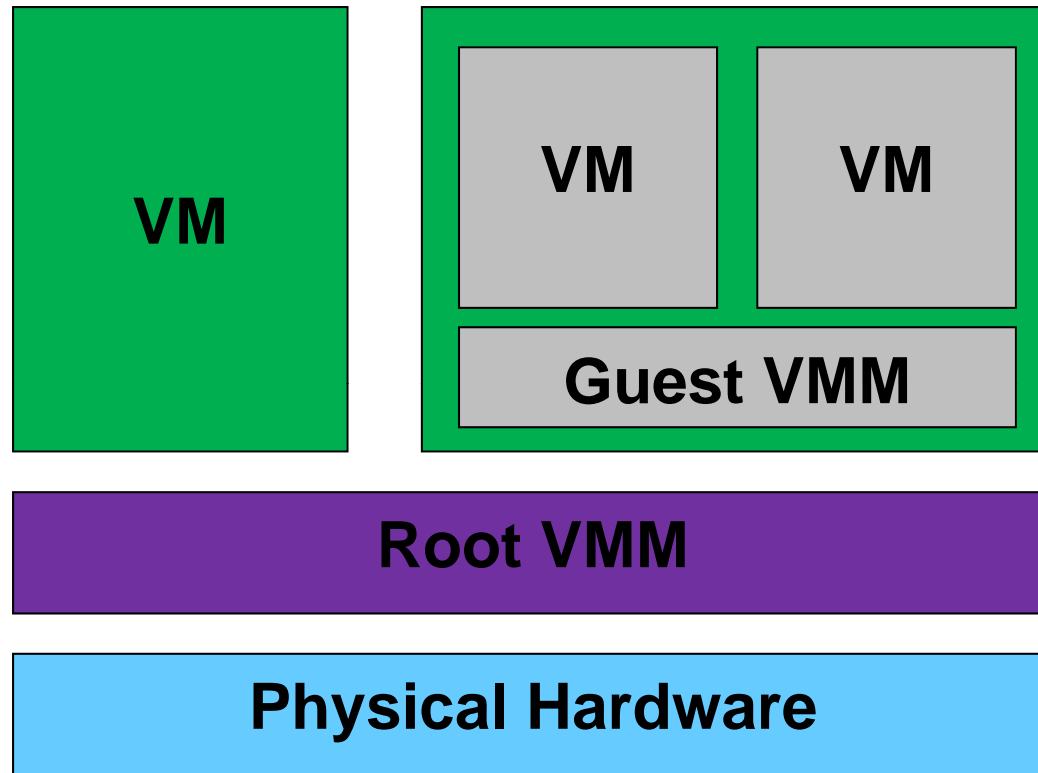
No Serialization and No Communication if No Data Conflicts

From IDF 2012: Rajwar and Dixon

④ Transactional Memory (TSX)

- Hardware Lock Elision (HLE)
 - SW uses XACQUIRE/XRELEASE to mark critical sections
 - HW executes transactionally w/o acquiring lock
 - Abort causes a re-execution without elision
- Restricted Transactional Memory (RTM)
 - SW uses XBEGIN/XEND to specify critical sections
 - Abort transfers control to target specified by XBEGIN
 - Software implements fix-up code in handler
- Nice results with Cuckoo hashing. Other apps?

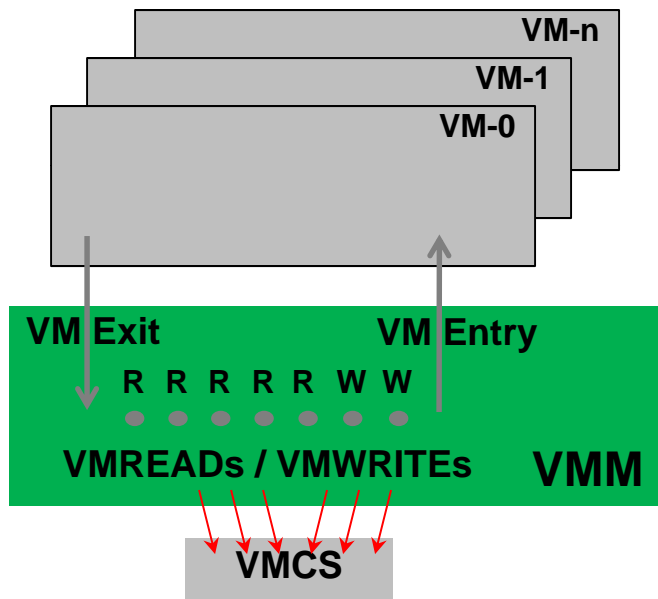
Nested Virtualization: Motivation



- Running a VMM as a guest to another VMM
- Many applications as uses for VT proliferate

Implementing Nesting...

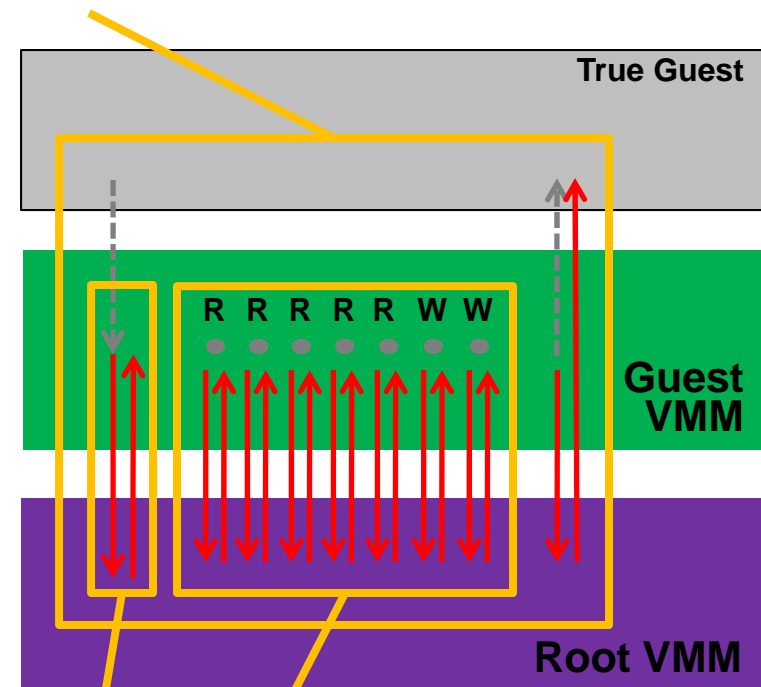
“Standard” Virtualization



- VMCS = VM Control Structure
- Holds guest and host CPU register state

Opportunity #2: Reduce “virtual” VM exits entirely (e.g., via EPT, vAPIC)

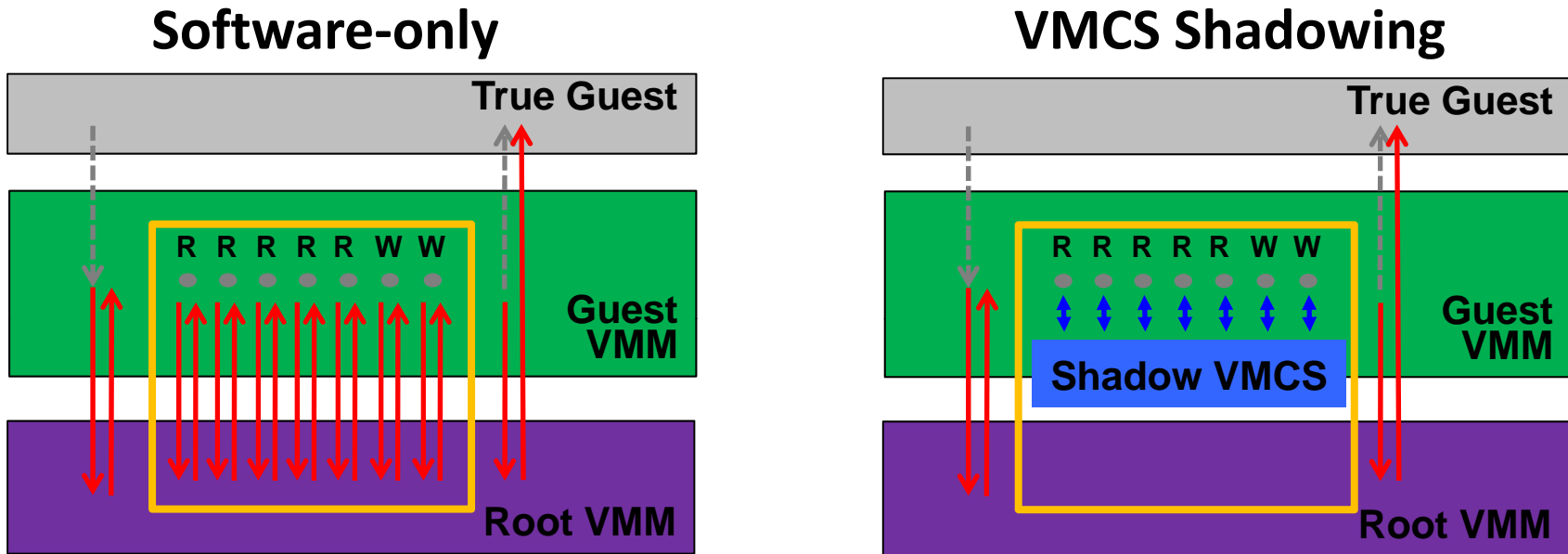
“Nested” Virtualization



Opportunity #3: Eliminate VM exits on guest VMCS Accesses

Opportunity #1: Reduce transition latencies

VMCS Shadowing



- Direct Guest VMM VMREAD/VMWRITE to a Shadow VMCS
 - Accesses to Shadow VMCS done by hardware
 - Eliminates majority of nesting-induced VM exits
 - Improves performance of software stacks that support nesting

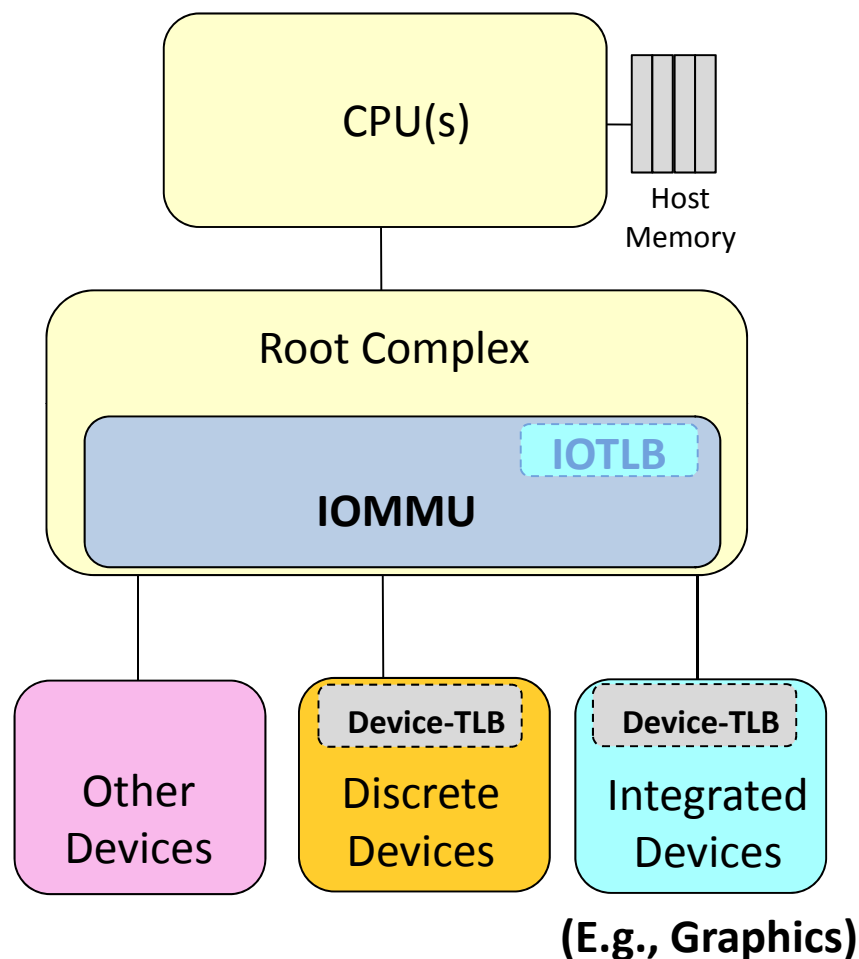
⑤ Other New VT Features

- Nested Virtualization Support
 - VMCS Shadowing
 - Applicability to “Automation” pillar? (e.g. Xen-Blanket)
- Support for faster VM migration
 - Extended Page Table (EPT) Accessed/Dirty bits
 - Applicability to “Cloudlet” & “To-the-Edge” research?
- Support for improved hypervisor-based Security
 - VMFUNC and #VE
 - May be of interest to Security ISTC researchers?

⑥ Shared Virtual Memory (SVM)

- Heterogeneous CPU/GPU Problems Today
 - CPU and GPU use separate address spaces...
 - ... expressed with different paging structures
 - Requires marshalling of data structures
- Application would ideally like to use:
 - Same virtual address space for CPU and GPU
 - Requires new hardware support
 - Natural extension VT-d (IOMMU) translation

SVM: System Architecture

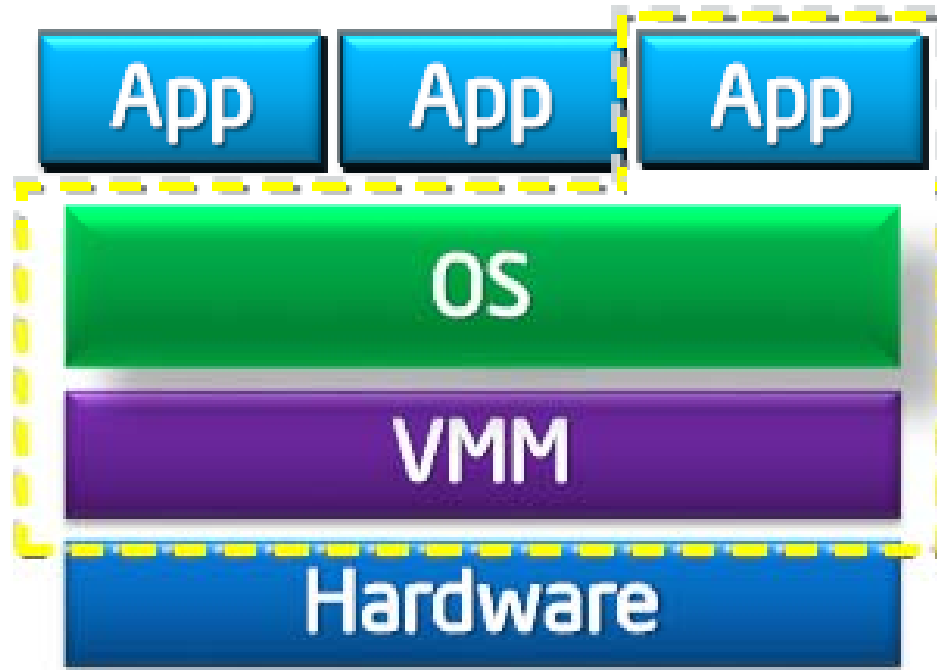


- Hardware Components
 - IOMMU support in platform
 - Device-TLB support at device (GPU)
- IOMMU
 - Provides a stable s/w interface
 - Service requests from device-TLB
 - IOTLB functions as secondary TLB
- Device-TLB
 - Functions as primary TLB
 - Detects page-faults at the source

Applicability to “Specialization” Research Pillar?

Security Guard Extensions (SGX)

Attack surface today

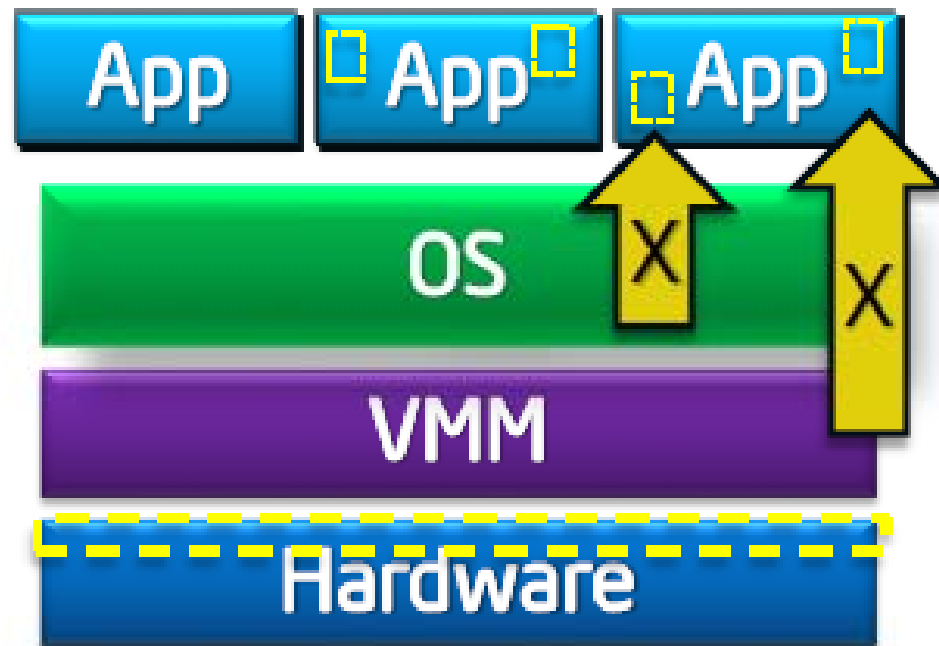



Attack Surface

From HASP 2013: McKeen, et al.

Security Guard Extensions (SGX)

Attack surface with Intel® SGX



Attack Surface 

From HASP 2013: McKeen, et al.

⑦ Security Guard Extensions (SGX)

- Application able to defend its own secrets
 - Smaller attack surface (app + processor)
 - Malware that subverts OS/VMM, BIOS, drivers can't steal secrets
- See Carlos Rozas talk from Wed
 - Demo of SGX emulator at SOSPP
 - Applicability to secure cloud services?

Overview

- Research Prototypes

- NVMe Emulator
- Persistent Memory Emulator
- Memory-Race Recorder

- New Technologies

- Transactional Memory (TSX)
- New VT Features
- Shared Virtual Memory (SVM)
- Security Guard Extensions (SGX)

- New Platforms

- Xeon Phi
- Micro-cluster
- Quark

- Software

- Persistent Mem Filesystem
- Data Plane Devel Kit (DPDK)

⑧ Xeon Phi Update

Next Intel® Xeon Phi™ Processor: **Knights Landing**

Designed using
Intel's cutting-edge
14nm process

The big news

Not bound by "offloading"
bottlenecks
***Standalone CPU or
PCIe coprocessor***

Leadership compute & memory
bandwidth

**Integrated
on-package memory**



All products, computer systems, dates and figures specified are preliminary based on current expectations, and are subject to change without notice.

Copyright © 2013 Intel Corporation. All rights reserved



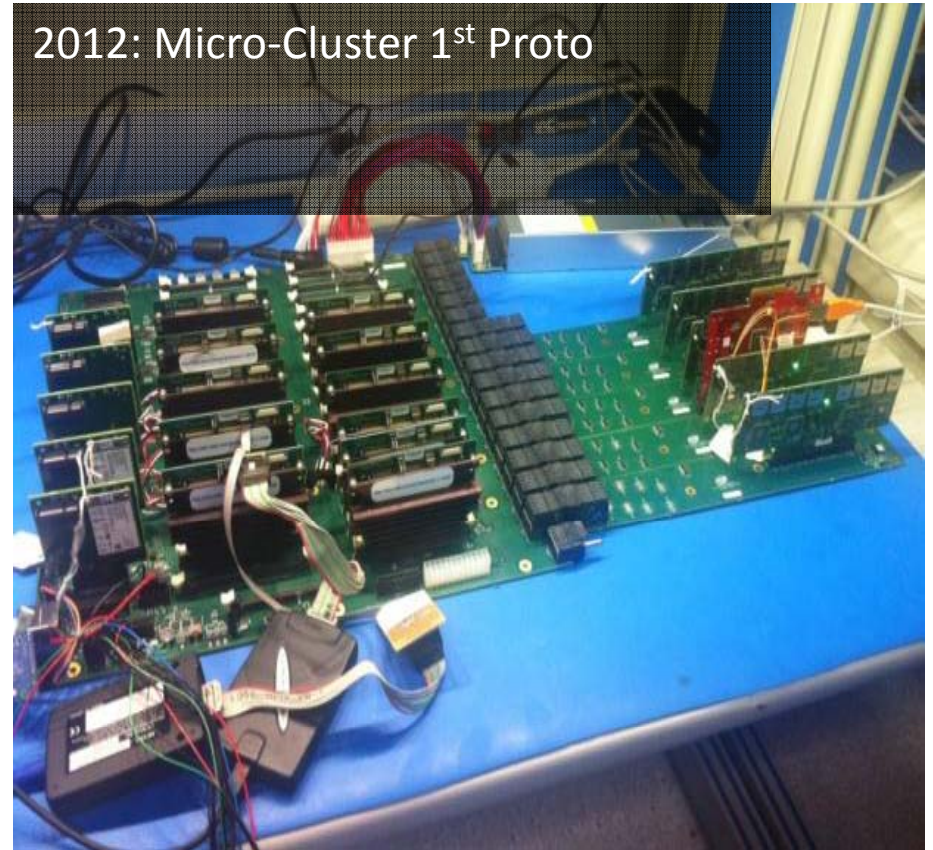
μ Cluster Evolution (inspired by FAWN)

2011: "Carpet Cluster"



"Carpet" Cluster

2012: Micro-Cluster 1st Proto



Lightpeak-based Fabric

⑨ μCluster 2.0



- 16 or 32 Avoton Nodes with Fulcrum Alta switch
- Basis for future software-router research

Quark and the Internet of Things

Forward pointer to Myles' talk...

Overview

- Research Prototypes

- NVMe Emulator
- Persistent Memory Emulator
- Memory-Race Recorder

- New Technologies

- Transactional Memory (TSX)
- New VT Features
- Shared Virtual Memory (SVM)
- Security Guard Extensions (SGX)

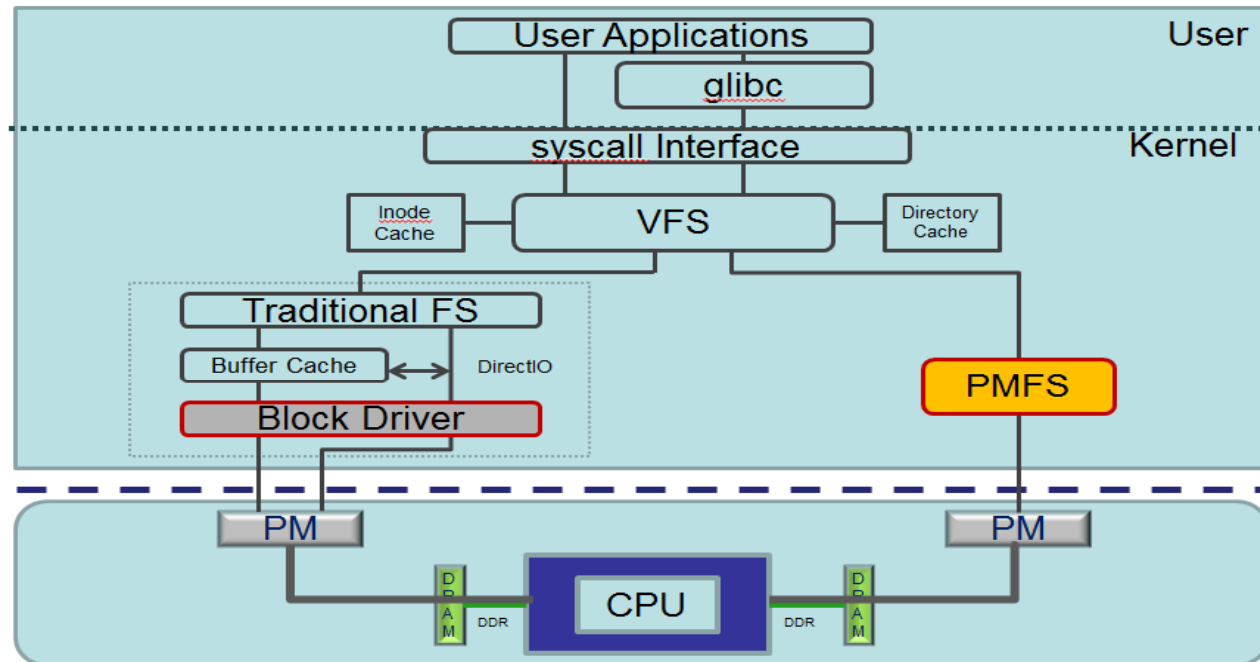
- New Platforms

- Xeon Phi
- Micro-cluster
- Quark

- Software

- Persistent Mem Filesystem
- Data Plane Devel Kit (DPDK)

Persistent Memory Filesystem (PMFS)

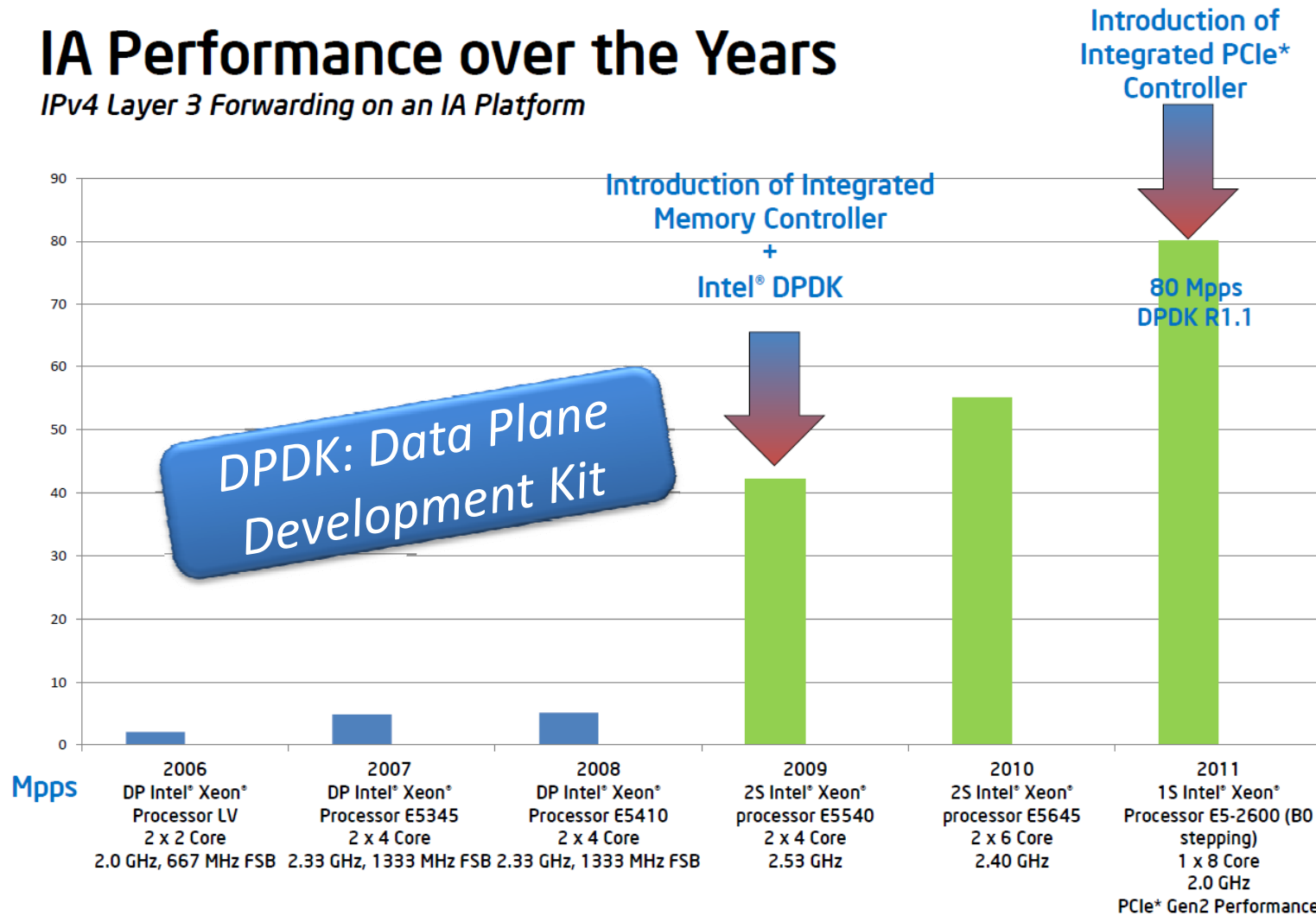


- Works with PM emulator described earlier
- Avoids traditional storage-stack components (page cache)
- Supports direct application access (via mmap) to PM
- Available at GitHub: <https://github.com/linux-pmfs/pmfs>

Advances in IA Packet Processing...

IA Performance over the Years

IPv4 Layer 3 Forwarding on an IA Platform



From: <http://www.intel.com/content/dam/www/public/us/en/documents/presentation/dpdk-packet-processing-ia-overview-presentation.pdf>

Data Plane Developer Kit (DPDK)

- Optimized packet processing for IA platforms
- Leverages recent server enhancements
 - Integrated memory and IO controllers
 - Direct I/O technology
- Available under BSD and GPL licenses
 - Nice work on soft switch by Andersen, et al.
 - Any interest for other research projects?

Discussion

- Some models for getting access to this stuff:
 - Internships on-site at Intel
 - Remote access through vLab
 - Access via lab at CMU
- Anything look interesting?
 - What would you do with it?
 - Priorities?
- Anything else you wish you had seen here?