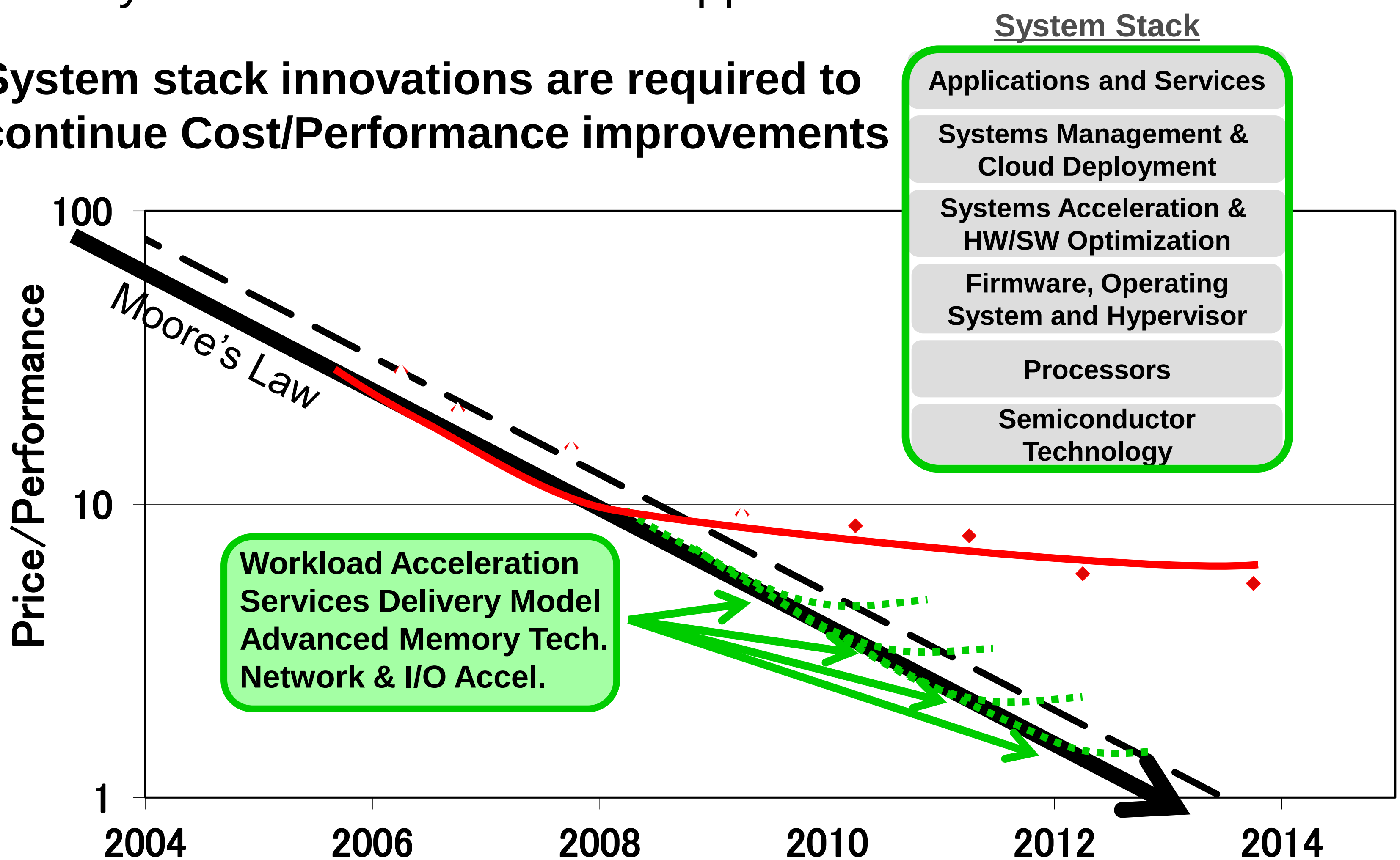
Power Systems

I/O Virtualization and System Acceleration in POWER8

Michael Gschwind
IBM Power Systems

Industry Trends Generate New Opportunities

System stack innovations are required to continue Cost/Performance improvements



System Design ca. 2015

- Consolidation in the Cloud
  - Virtual Machine Aggregation
  - Processor Virtualization
- I/O Aggregation
  - I/O Virtualization
  - I/O Isolation
- Maintain Performance growth for critical workloads
  - Under Power constraints
  - Under Cost constraints
  - Under Area constraints

Innovation Through Open Standards

OpenPOWER™

- Consolidation in the Cloud
  - Virtual Machine Aggregation
  - Processor Virtualization
- I/O Aggregation
  - I/O Virtualization
  - I/O Isolation
- Maintain Performance growth for critical workloads
  - Under Power constraints
  - Under Cost constraints
  - Under Area constraints

Power Instruction Set Architecture

I/O Design Architecture

Coherent Accelerator Interface Architecture

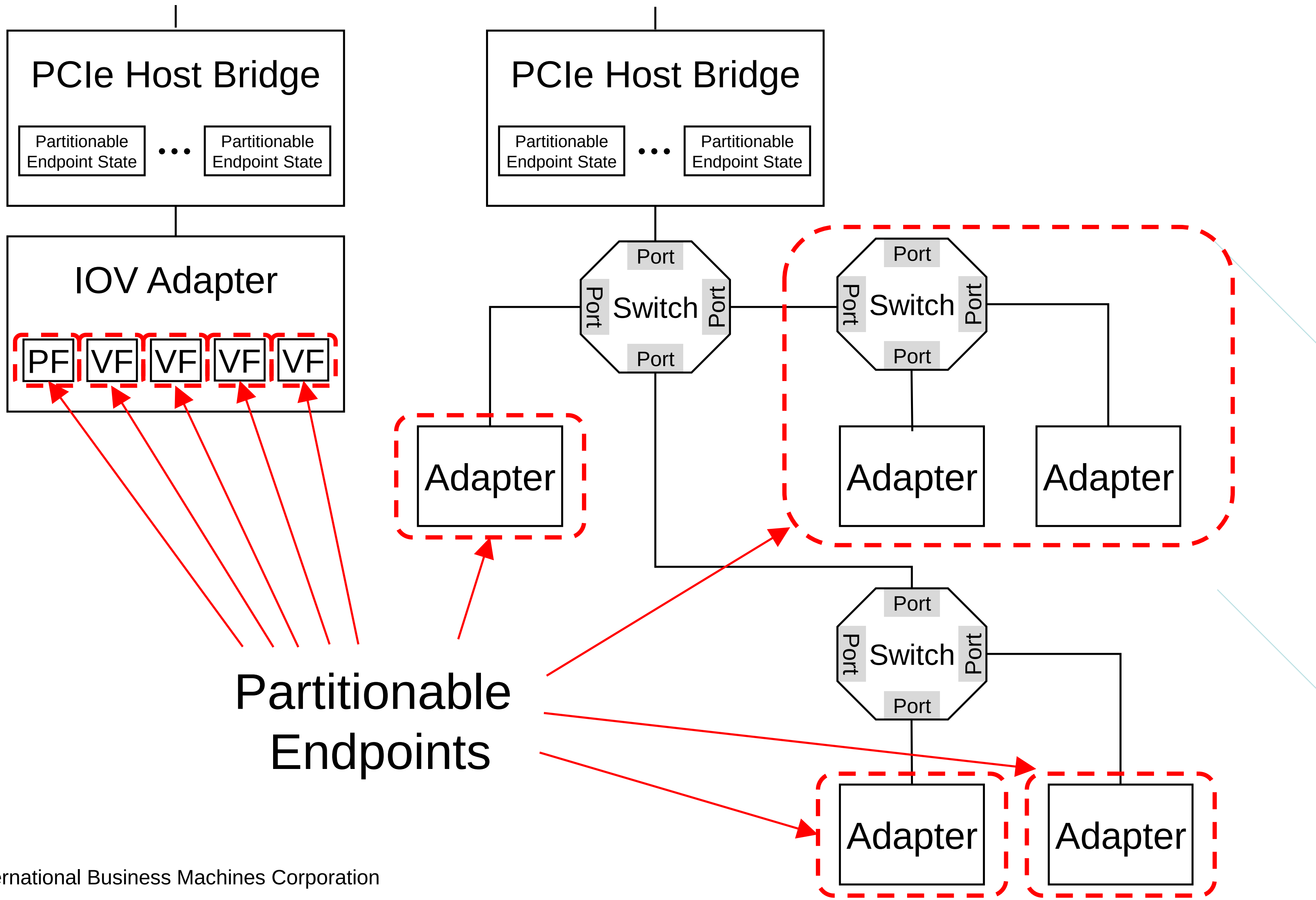
I/O Design Architecture

- I/O Virtualization
  - IO address translation
  - Single level translation → support contiguous PowerVM partitions
  - Hierarchical translation → support Linux/KVM partition memory

- Partition data isolation
  - Separate I/O address spaces for partitions

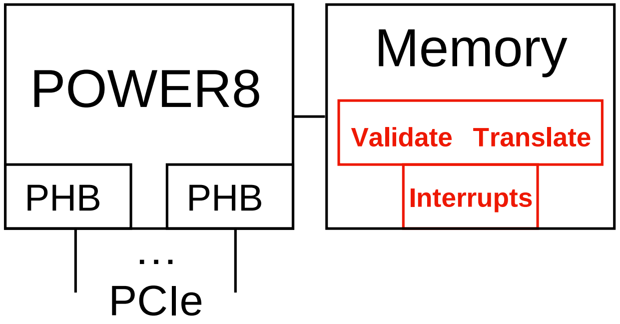
- Partition fault isolation
  - Fault management domains based on Partitionable Endpoints (PE)

Partitioning and Managing the I/O Space: Partitionable Endpoints



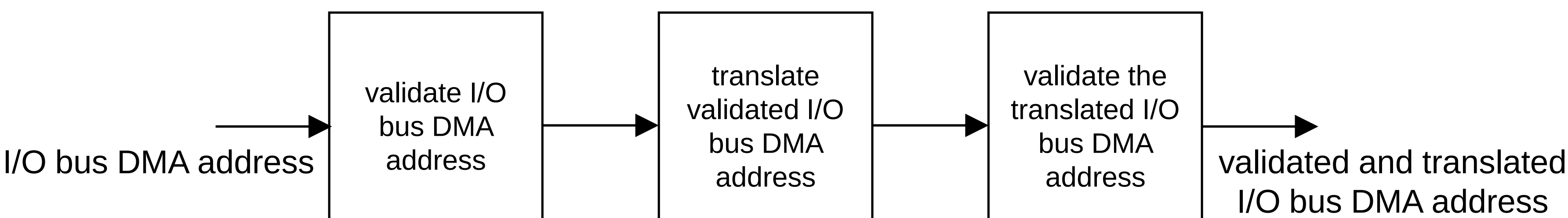
Managing Partitionable Endpoints

- Many I/O functions tracked using Partitionable Endpoints
  - MMIO Load/Store address domains
  - DMA I/O bus address domains and TCEs
  - Interrupts
  - Ordering of transactions per PE
  - PE Error and Reset domains
- Enhanced RAS capabilities
  - Enhanced I/O Error Handling (EEH)
  - Partition isolation



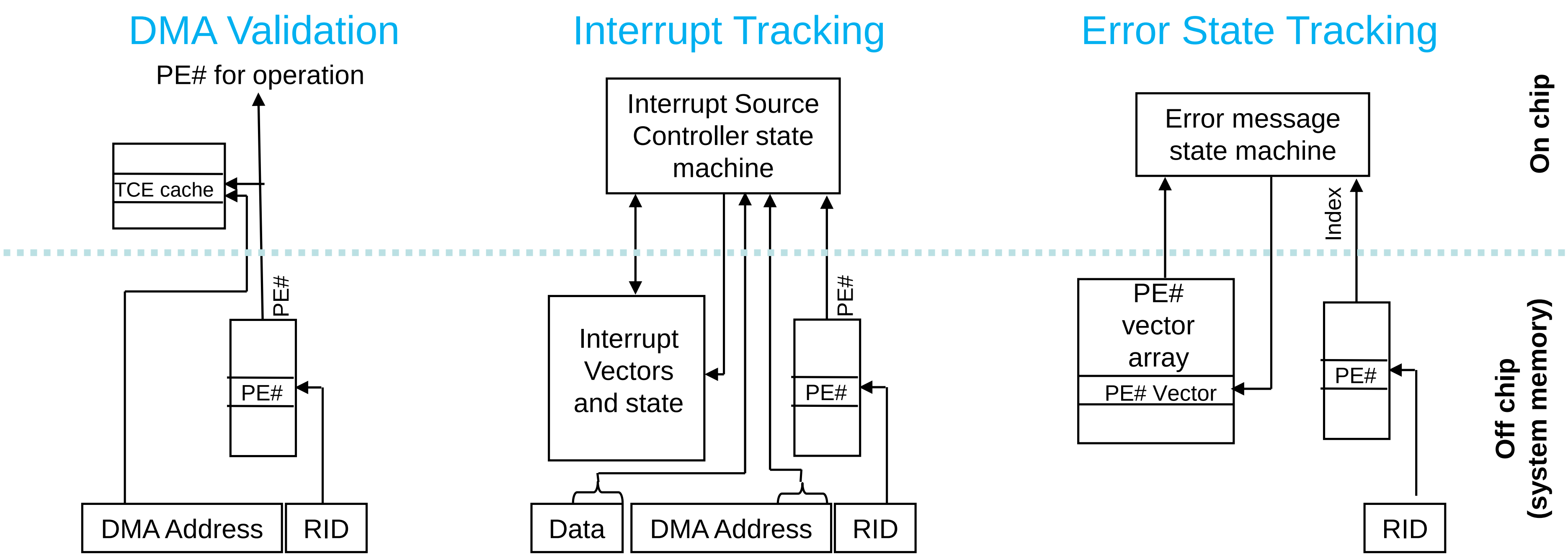
I/O DMA memory translation

- Translate I/O DMA memory addresses to system memory address
  - Isolation of partitions
  - Create contiguous view of non-contiguous data
  - Enable 32 bit devices for 64 bit systems
- Processor chip contains cache of recently accessed tables
  - Full tables stored in system memory



Partitionable Endpoint Management

- DMA validation
  - “Trusted” Requester ID (RID): 16-bit bus/device/function number
  - “Untrusted” 64-bit address – received from device driver
- Interrupts tracked in system memory
- PEs impacted by I/O error identified via RID
  - SRIOV Physical Function fault → PEs of Phys. & all Virt. Adapter Functions

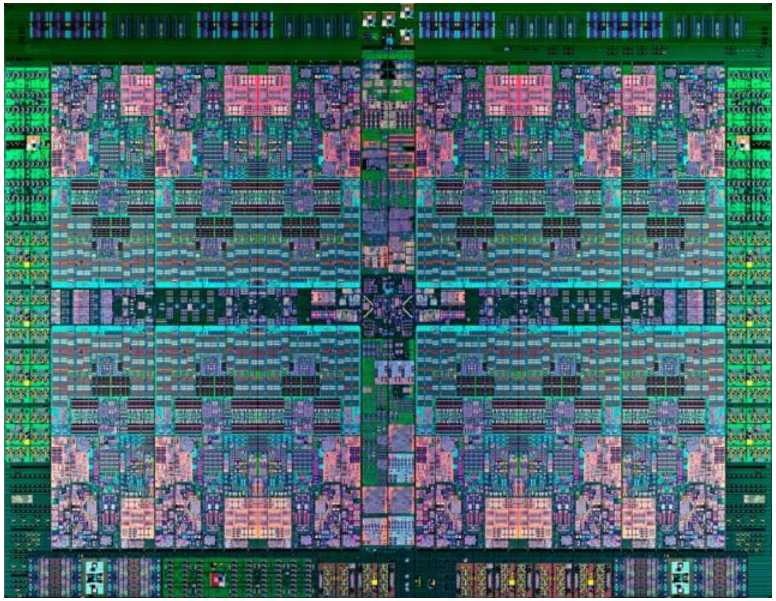


Power Systems

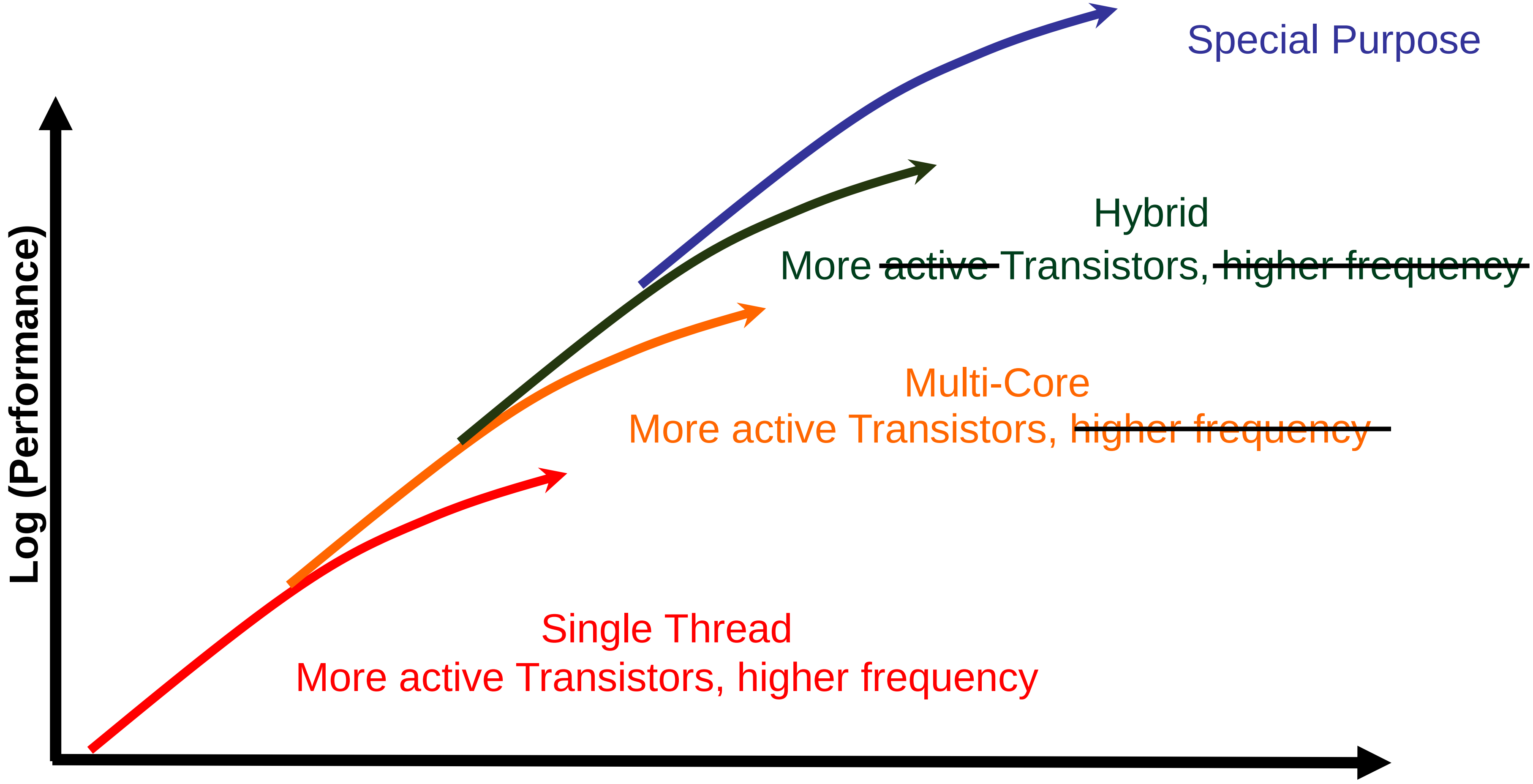
IBM

Coherent Accelerator Processor Interface (CAPI)

- Integrated in every Power8 system
- Builds on a long history of IBM workload acceleration
- Integrates with big-endian and little-endian accelerators



Microprocessor Trends



Heterogeneous System Challenges

- **The 4 ‘P’s of System Design**

- Programmer **Productivity**
- Realize accelerator **Performance** benefits
- **Portability**: Investment protection for applications
- **Partitioning** for multi-user systems: processes, partitions

Workload-optimized acceleration with coherent accelerators

- Attached accelerators
  - Accelerate functions that do not fit traditional CPU model
  - Heterogeneous System Architecture

- Coherent integration in system architecture
  - Data sharing
  - Programming
  - Performance

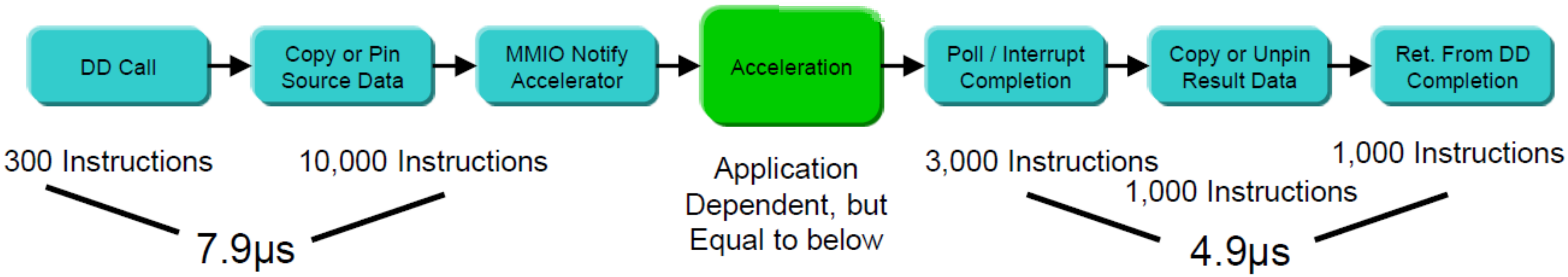
Application Acceleration

- Fine-grained data sharing
  → coherent, shared memory
- Accelerator-initiated data accesses/transfers
  → coherent, shared memory
- Pointer identity
  → shared addressing
- Flexible synchronization
  → symmetric, programmable interfaces

CAPI Acceleration overcomes Device Driver Deceleration

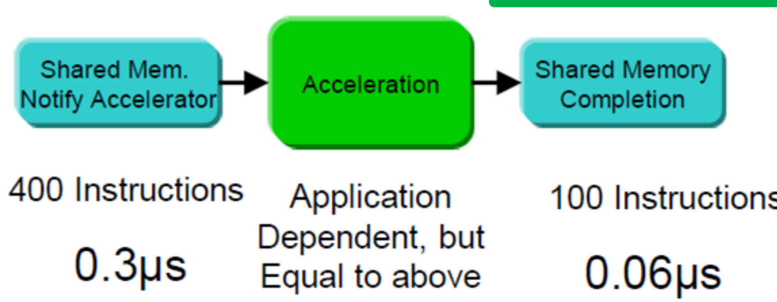
Typical I/O Model Flow:

Total ~13µs for data prep



Flow with Coherent Model:

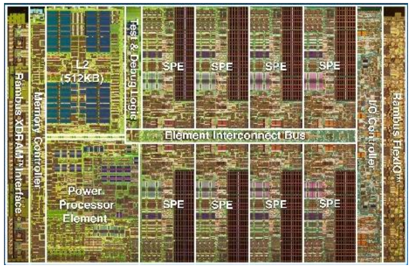
Total ~0.36µs



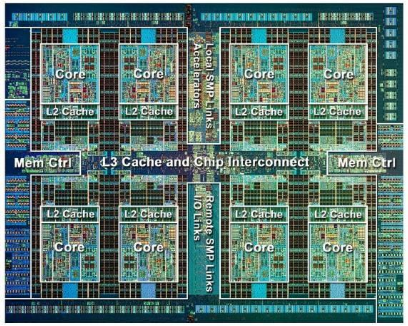
Workload-optimized acceleration

- On-chip integrated accelerators (SoC design)
  - Compute accelerator (Cell BE)
  - Compression (P7+)
  - Encryption (P7+)
  - Random number generation (P7+)
  - …

- SoC design offers highest integration, but…
  - Requires new chip design for accelerator
  - Long time to market
  - Requires very high volumes



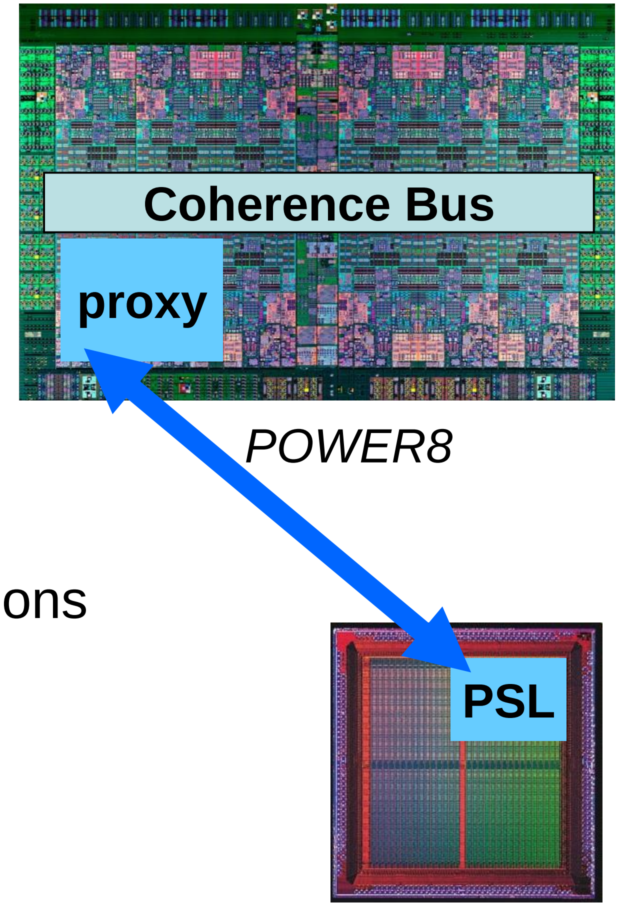
Cell BE



POWER7+

CAPI: Coherent Accelerator Processor Interface

- Integrate accelerators into system arch.
  - Modular interface
  - Third-party high value-add components
- Standardized, layered protocol
  - architectural interface
  - functional protocol
  - PCIe signaling protocol
- Create workload-optimized innovative solutions
  - Faster time to market
  - Lower bar to entry
  - Variety of implementation options
    FPGAs, ASICs

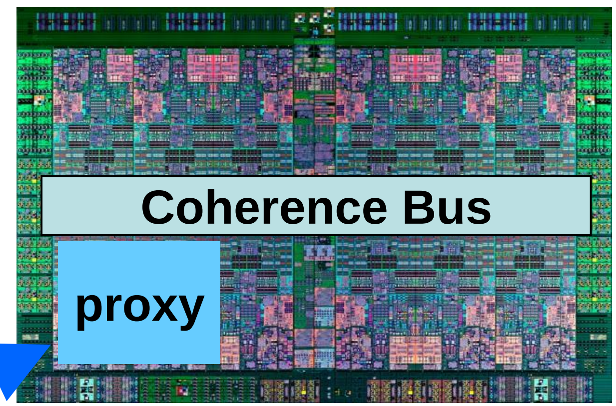


* Power Service Layer

CAPI accelerator programming

Virtual Addressing

- **Pointer identity:** Pointers reference same object as the host application
- CAPI accelerators work with same virtual memory addresses as CPU
- CAPI shares page tables and provides address translation of host application
- Peer-to-peer programming between CPU and accelerator with in-memory data sharing



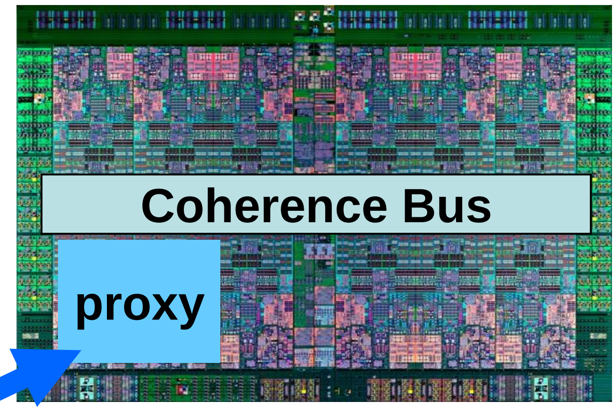
Virtualization and Partitioning

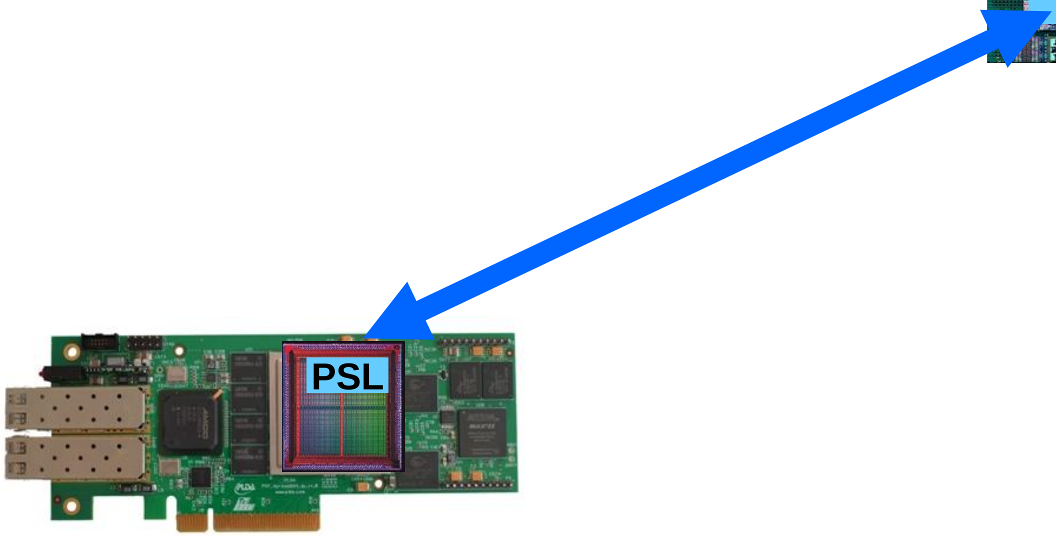
- Address translation supports process isolation ➔ Accelerator has access to application context (only)
- Address translation supports partition isolation ➔ Accelerator has access to partition data (only)

CAPI accelerator programming

Hardware Managed Cache Coherence

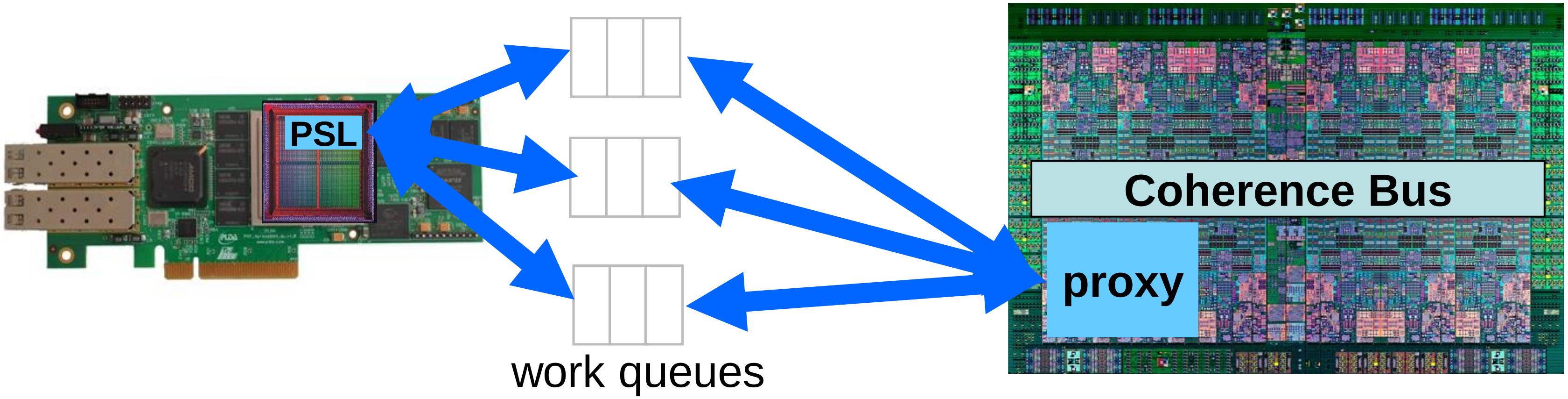
- No need for memory pinning
- Data fetched by accelerator based on accelerator application flow
- Accelerator participates in locks
- Low latency communication





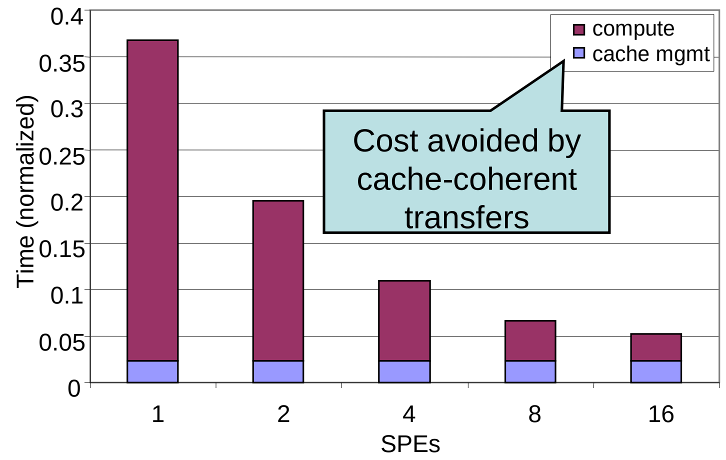
CAPI Accelerator Virtualization

- Dedicated Model
  - Accelerator assigned to single process in a partition
  - Binding via Operating System (process) and Hypervisor (partition)
- Time-quantum shared programming model
  - Protocol-controlled model
- Accelerator-directed shared programming model
  - networking model (select context based on incoming data)

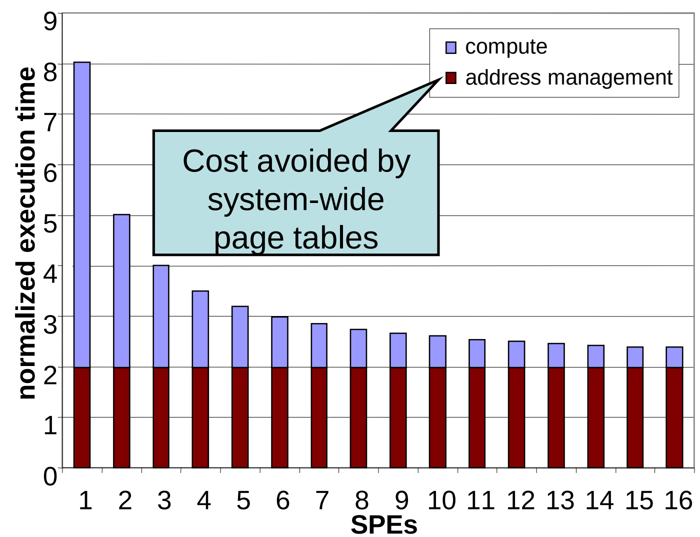


Coherent acceleration of data sharing

- Offload data synchronization and data transfers
  - No need to invalidate data before initiating I/O transfers
  - Accelerator feeds itself ➔ avoid use of high-function CPU as data mover



- Available translation, transfer and synchronization bandwidth scales with parallelism
  - As more accelerators are used, available resources scale up
  - Avoid CPU becoming serial bottleneck

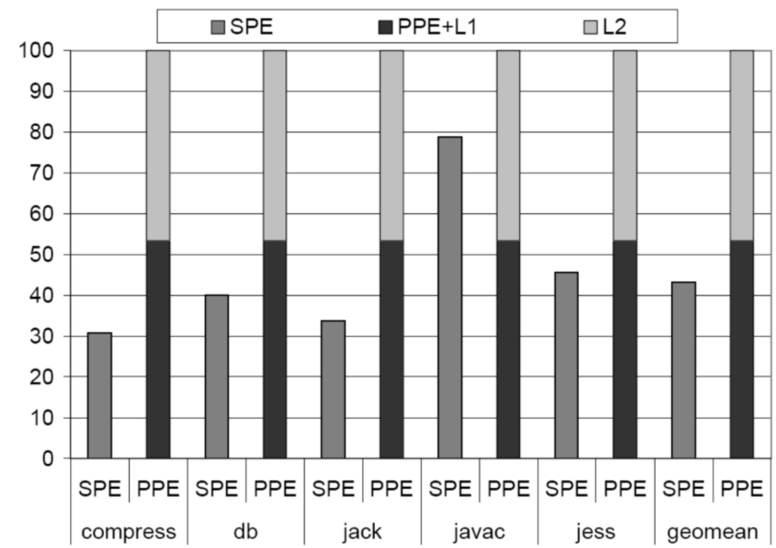


[Gschwind, ICCD 2008]

Coherent accelerator programming flexibility

- Garbage collection accelerator
  - Explore boundaries of acceleration
  - Study accelerator programmability
- Pointer identity: advanced data structures
  - Autonomous traversal of data
- Self-paced memory access
  - Simplifies data management
  - No callbacks to request data
  - Zero-copy data access
- Handle complex access patterns

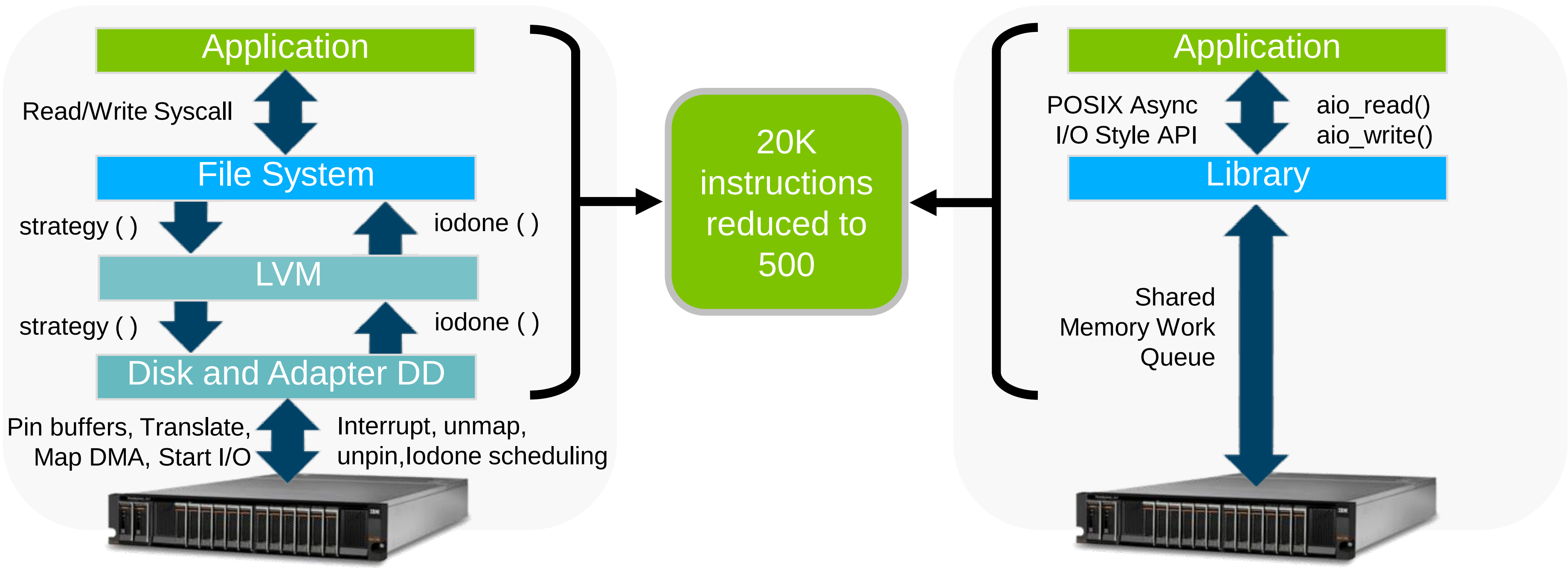
Normalized area×delay product



[Cher & Gschwind, VEE 2008]

CAPI Attached Flash Optimization

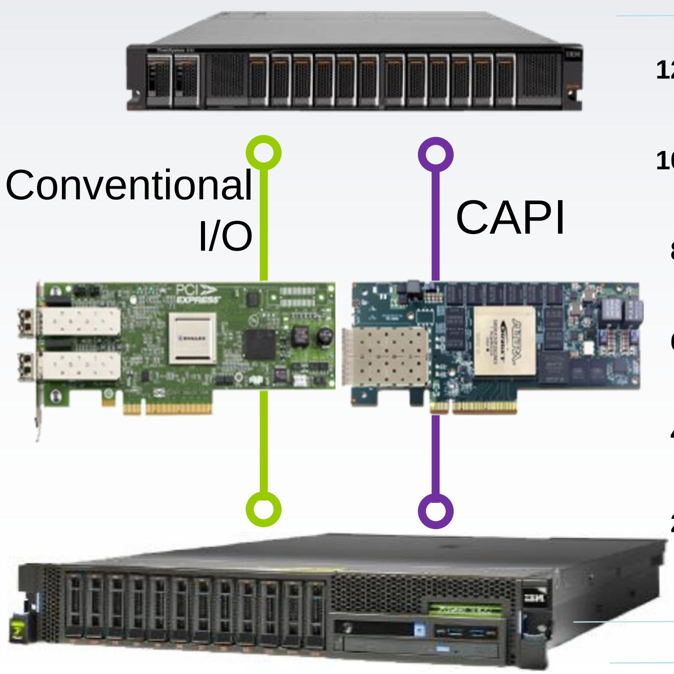
- Attach IBM FlashSystem to POWER8 via CAPI
- Read/write commands issued via APIs to eliminate 97% of path length
- Saves 20-30 cores per 1M IOPS



CAPI Unlocks the Next Level of Performance for Flash

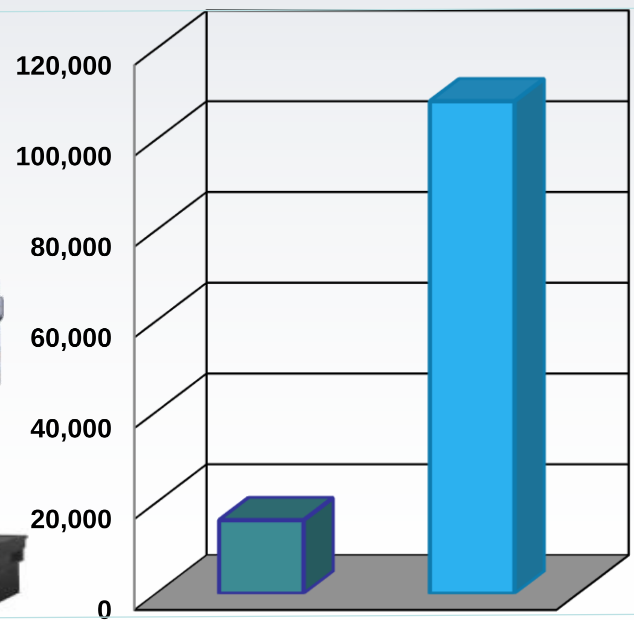
Identical hardware with 2 different paths to data

FlashSystem

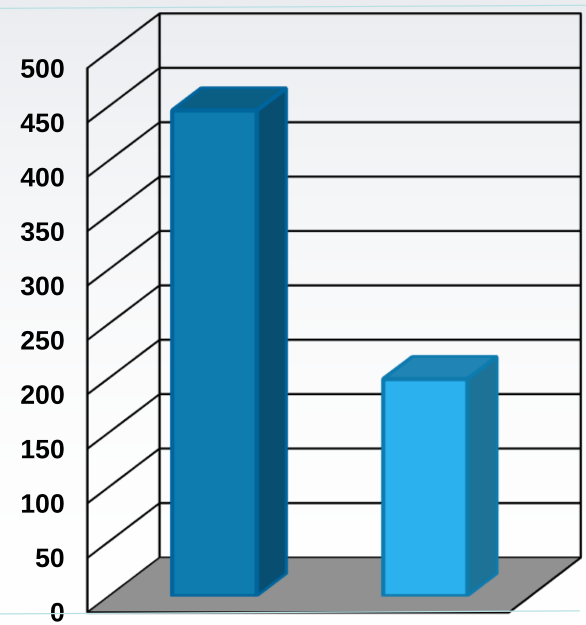


POWER S822L

IOPS per Hardware Thread



Latency (µs)

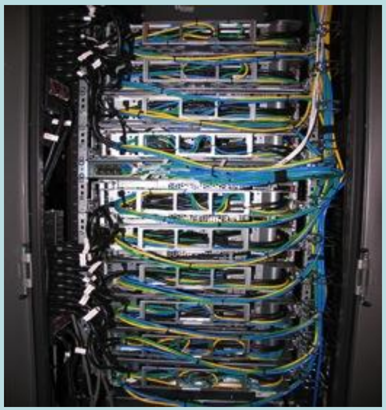


>5x better IOPS per HW thread

>2x lower latency

What CAPI Means for NoSQL Solutions

Today's NoSQL in memory (x86)



www

10Gb Uplink

Load Balancer

500GB Cache Node

Backup Nodes

Infrastructure Requirements

- Large distributed (Scale out)
- Large memory per node
- Networking bandwidth needs
- Load balancing

Differentiated NoSQL (POWER8 + CAPI + FlashSystem)

www

10Gb Uplink

POWER8 Server

Flash Array w/ up to 56TB

Infrastructure Attributes

- 192 threads in 4U server drawer
- 56 TB of flash per 2U drawer
- Shared Memory & cache for dynamic tuning
- Elimination of I/O and network overhead
- Cluster solution in a box

Power CAPI-attached FlashSystem for NoSQL regains infrastructure control and reigns in the cost to deliver services.

Summary

- POWER8 delivers advanced virtualization for CPU and I/O
  - High-performance I/O virtualization
  - Data isolation
  - Fault isolation
- POWER8 takes the next step in exploiting system accelerators
  - Eliminate overheads inherent in I/O model
  - Reduced latency
  - Increased throughput
- Collaborative Innovation based on Open Standards
  - Both specifications donated to OpenPOWER Foundation by IBM
  - Available royalty-free to all OpenPOWER members