

# **Omnix: an accelerator-centric OS for omni-programmable systems**

**Rethinking the role of CPUs in modern computers**

**Mark Silberstein**  
EE, Technion

# Oratorio for CPUs, accelerators and OS in 4 parts


Part 1: Accelerando con brio

We  accelerators

Part 2: Amore SOSPenuto appassionato

OS  CPU  accelerators

Part 3: Subito non-CPUtto

CPU 

Part 4: Tutti accelerando a capella

OmniX = OS  accelerators

# 2004: The free lunch is over

On the  
blog



November 4: [Other Concurrency Sessions at PDC](#)  
November 3: [PDC'09: Tutorial & Panel](#)

## The Free Lunch Is Over

A Fundamental Turn Toward Concurrency in Software

By Herb Sutter

The biggest sea change in software development since the OO re

*This article appeared in [Dr. Dobb's Journal](#), 30(3), March 2005.*

**Update note: The CPU trends graph last updated August 2009  
first posted here in December 2004.**

# 2015: no more lunch as we know it

The **last** International Technology Roadmap for Semiconductors (ITRS)

IEEE Rebooting Computing Initiative & International Roadmap of Devices and Systems

Tom Conte, 2015 IEEE Computer Society President,  
Co-Chair, IEEE Rebooting Computing Initiative,  
Schools of CS & ECE, Georgia Institute of Technology

## What the problem is

- Transistors are getting smaller *but not faster*
  - From a microarchitect's perspective:  
10nm isn't any better than 14nm, which was only marginally better than 22nm
- Moore's Law for 2D *really* ends in 2021
- Single thread exponential performance scaling ended in 2005
  - *Multicore didn't continue scaling*

2

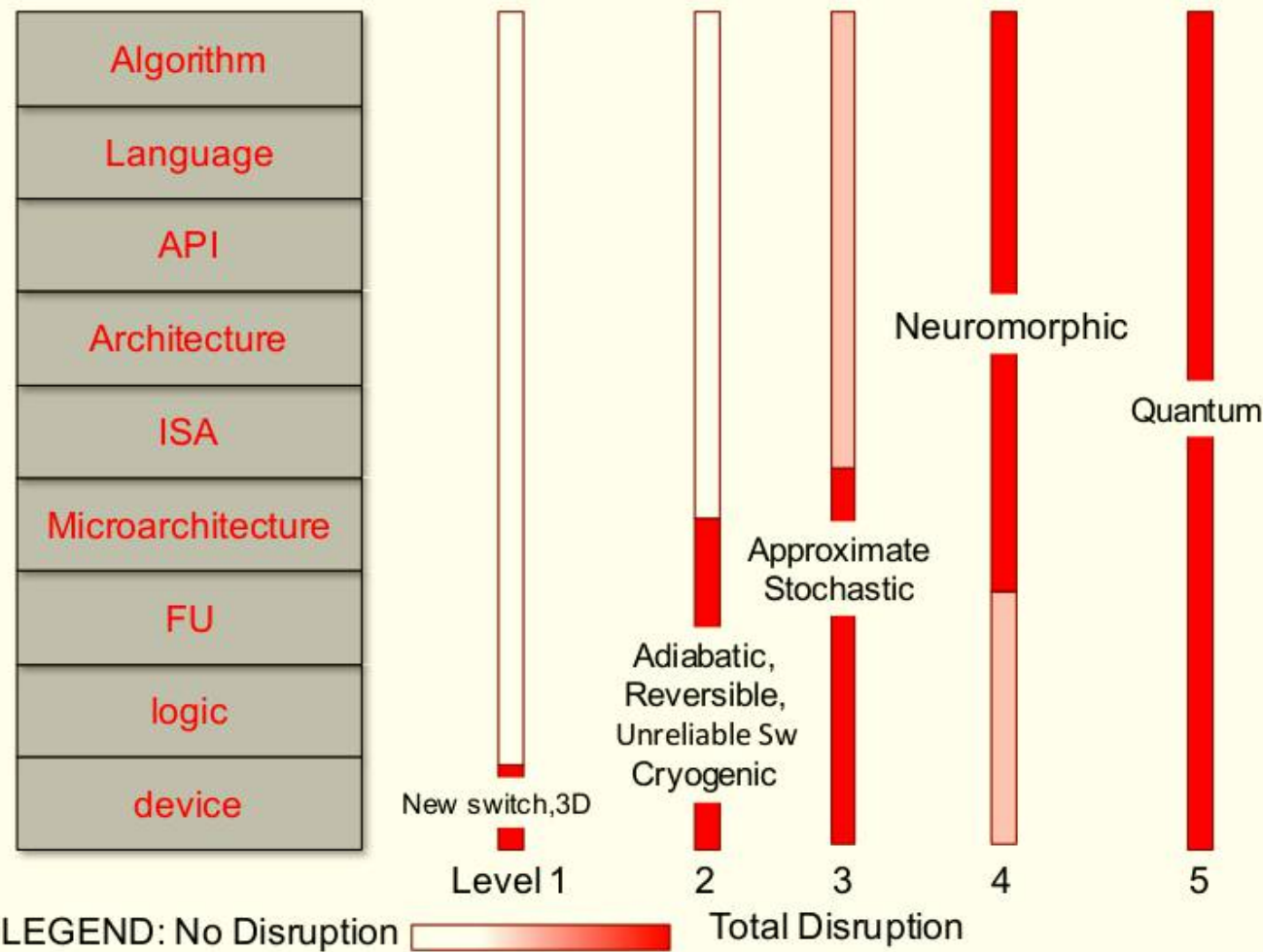


# Looking beyond CMOS

- Cryogenic computing
- Approximate/stochastic computing
- Neuromorphic computing
- Biological computing/storage
- Quantum computing

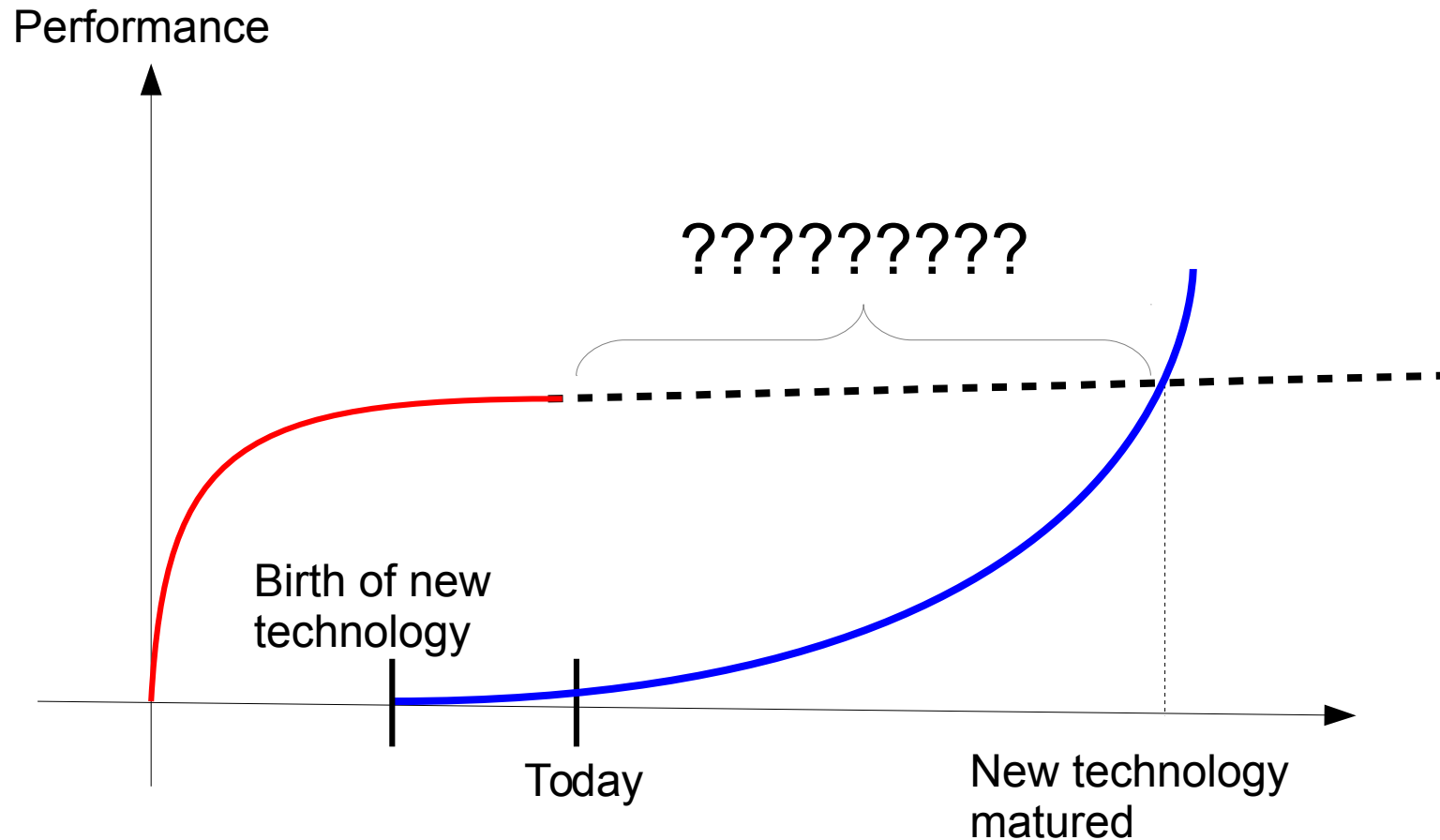
# Looking beyond CMOS

## Differing Levels of Disruption in Computing Stack

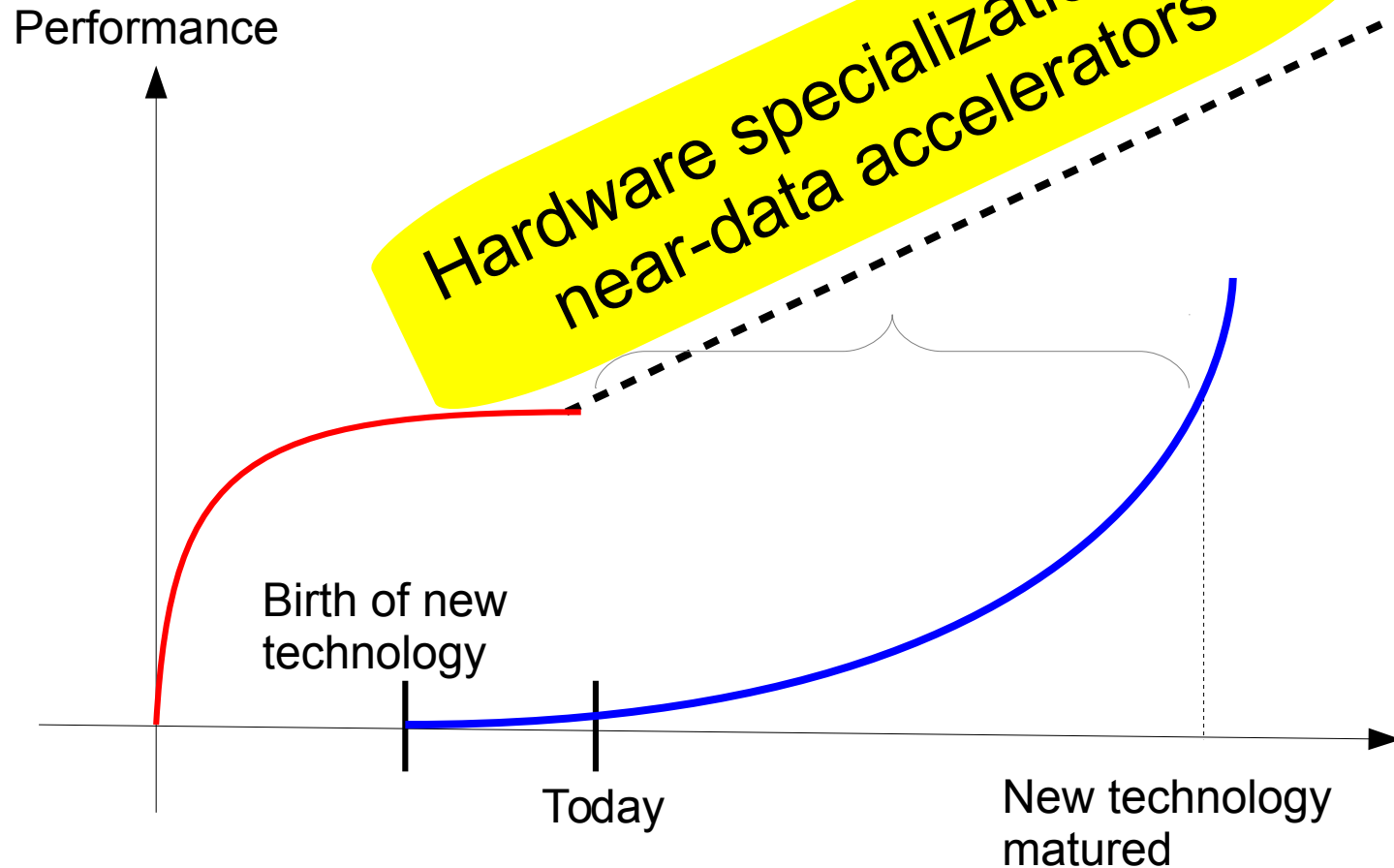


From «IEEE rebooting computing»

# What to do until the next revolution?



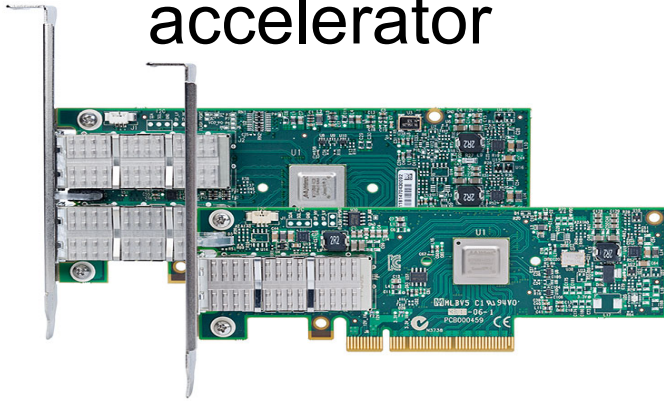
# What to do until the next revolution?



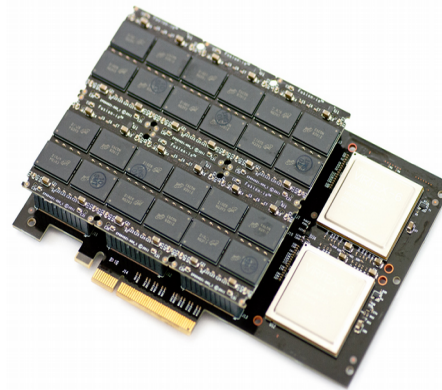


# Computer hardware: circa ~2017

Network I/O  
accelerator



GPU parallel  
accelerator

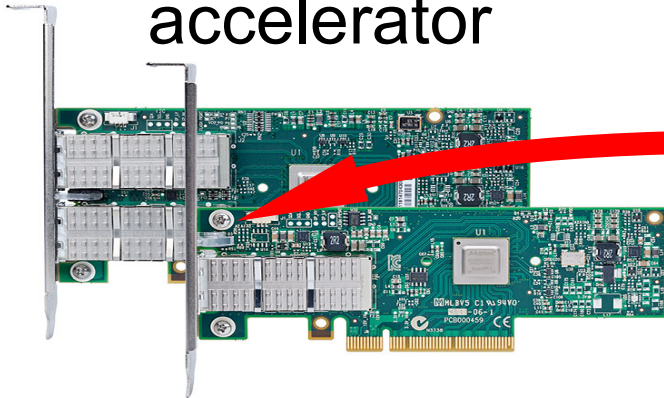


[size ~ transistor count]

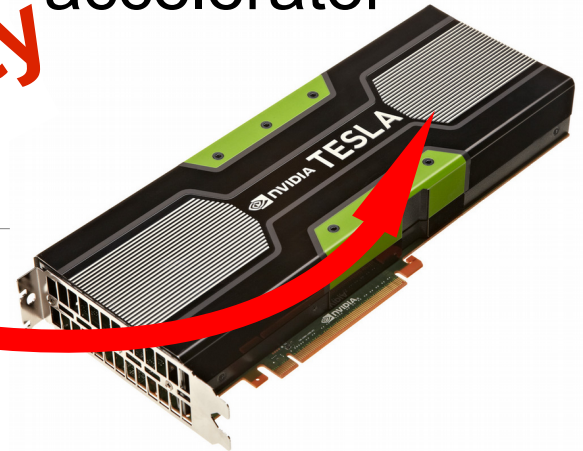
Storage I/O accelerator

# Central Processing Units (CPUs) are no longer **Central**

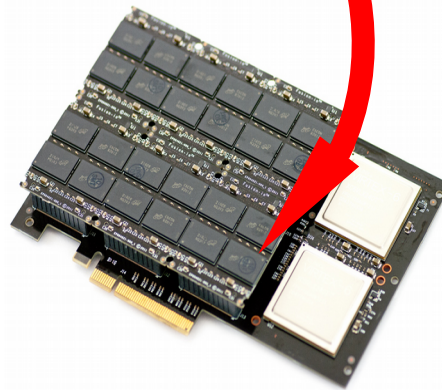
Network I/O  
accelerator



GPU parallel  
accelerator



**Programmability**



Storage I/O accelerator

# Omni-programmable system

## Near-X-execution Units: NXUs

Network I/O  
accelerator

GPU parallel  
accelerator

**Programmability**

**Near-Data  
Processing**

**Accelerated  
Processing**

**Near-Data  
Processing**

Storage I/O accelerator

Part 2: Amore SOSPenuto appassionato

OS  CPU  accelerators

Challenges of programming  
omni-programmable systems

# Truisms

Programming accelerators is hard

but

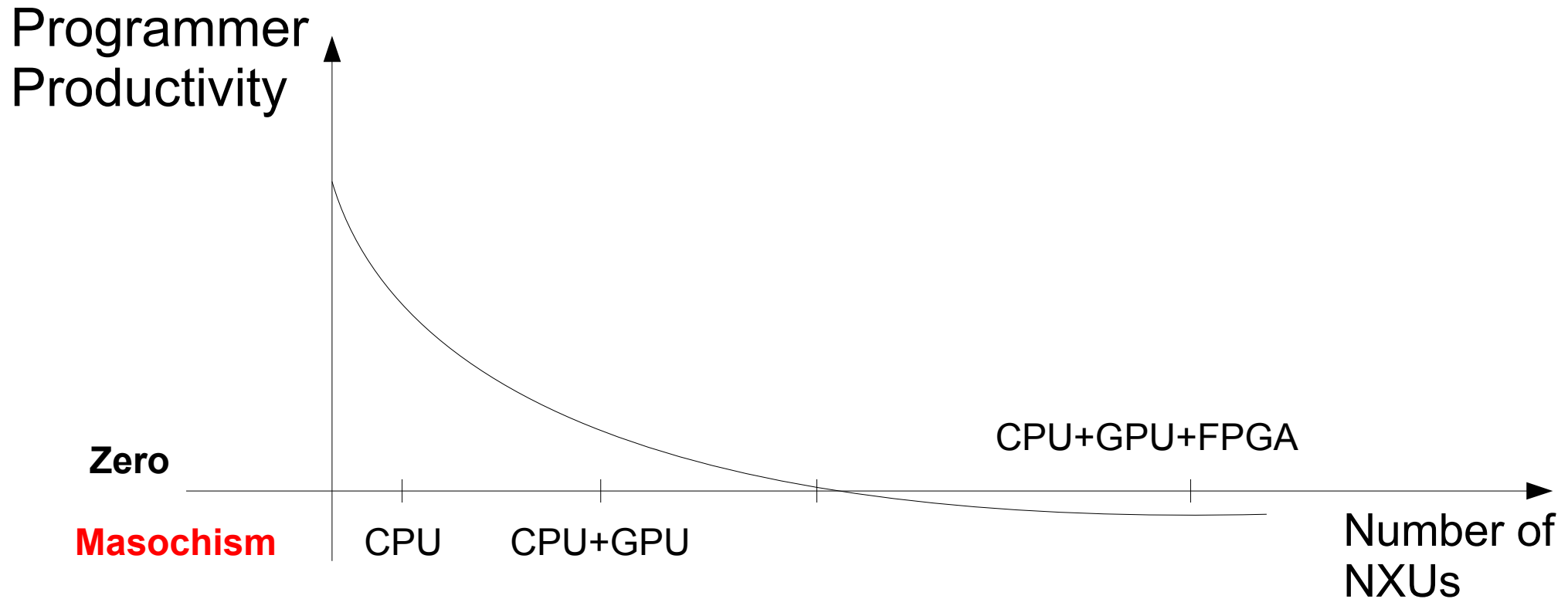
Programming is hard

Writing efficient programs is hard

Multi-threaded programming is hard

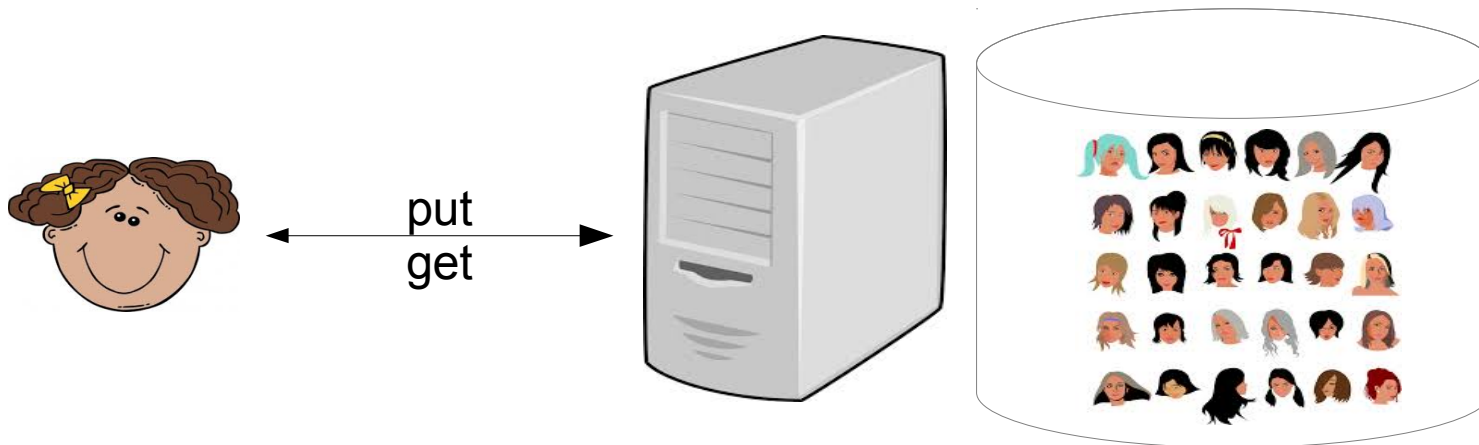
so...?

# Maintaining whole-application efficiency will be hard



# Example: image server

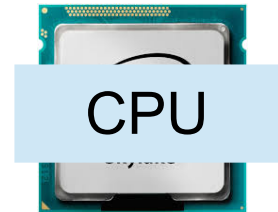
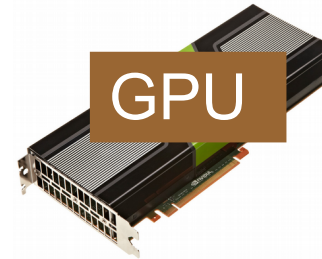
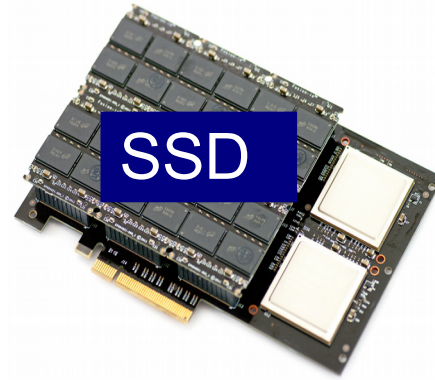
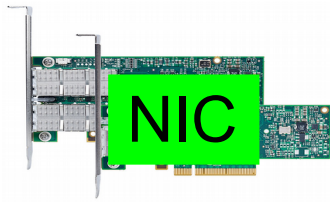
1. put: parse  $\rightarrow$  contrast-enhance  $\rightarrow$  store
2. get: parse  $\rightarrow$  resize  $\rightarrow$  store  $\rightarrow$  marshal





# Accelerating with NXUs

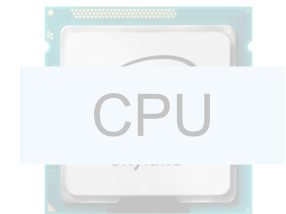
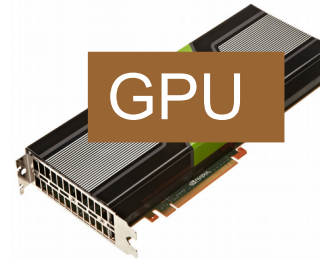
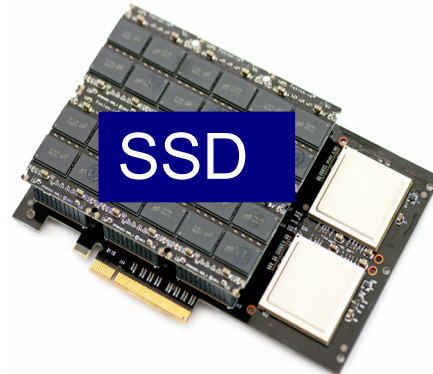
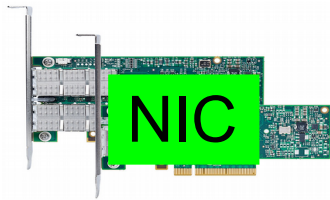
1. put: **parse** → **contrast-enhance** → **store**
2. get: **parse** → **resize** → **store** → **marshal**





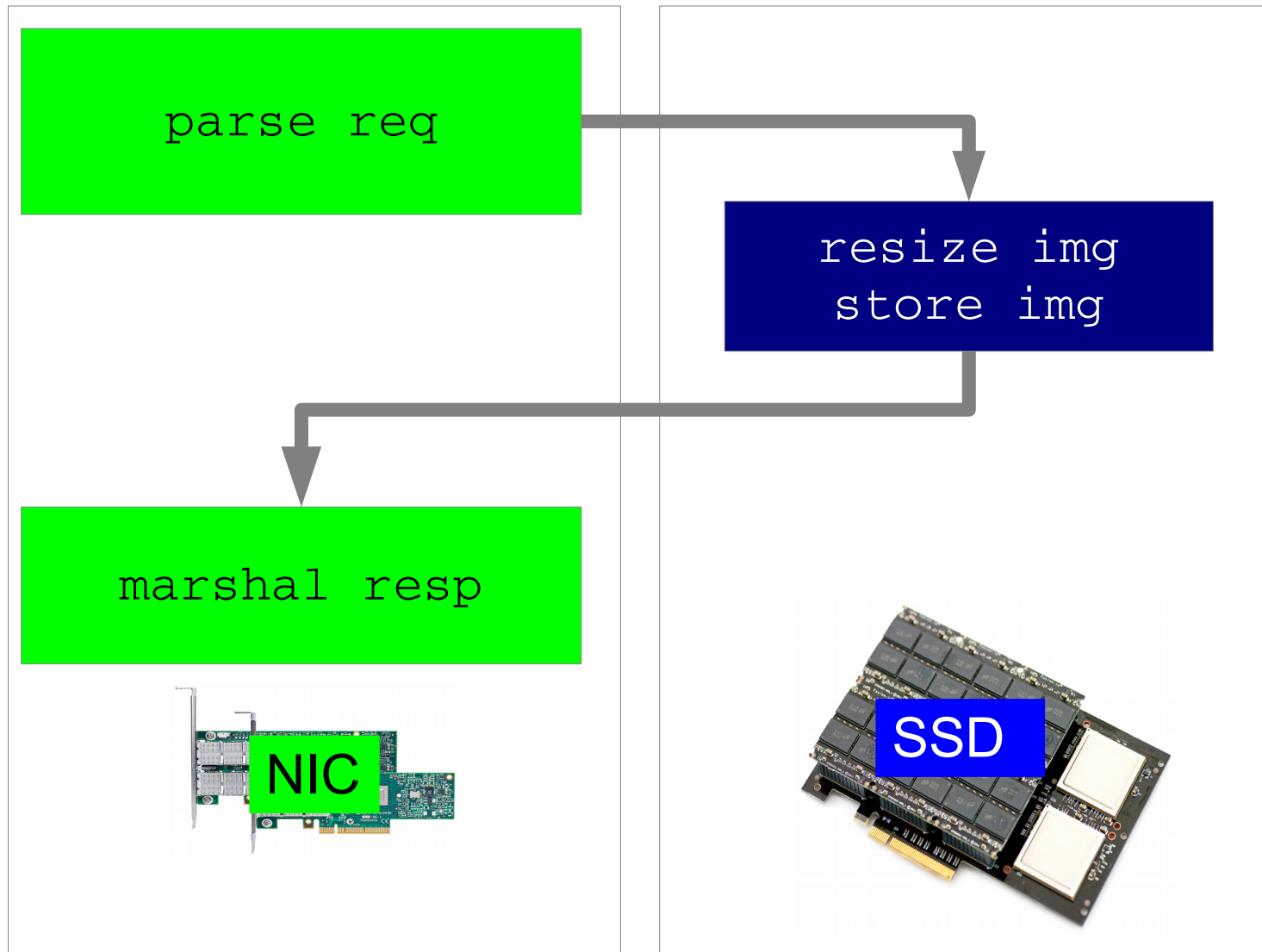
# Accelerating with NXUs

1. put: **parse** → **contrast-enhance** → **store**
2. get: **parse** → **resize** → **store** → **marshal**



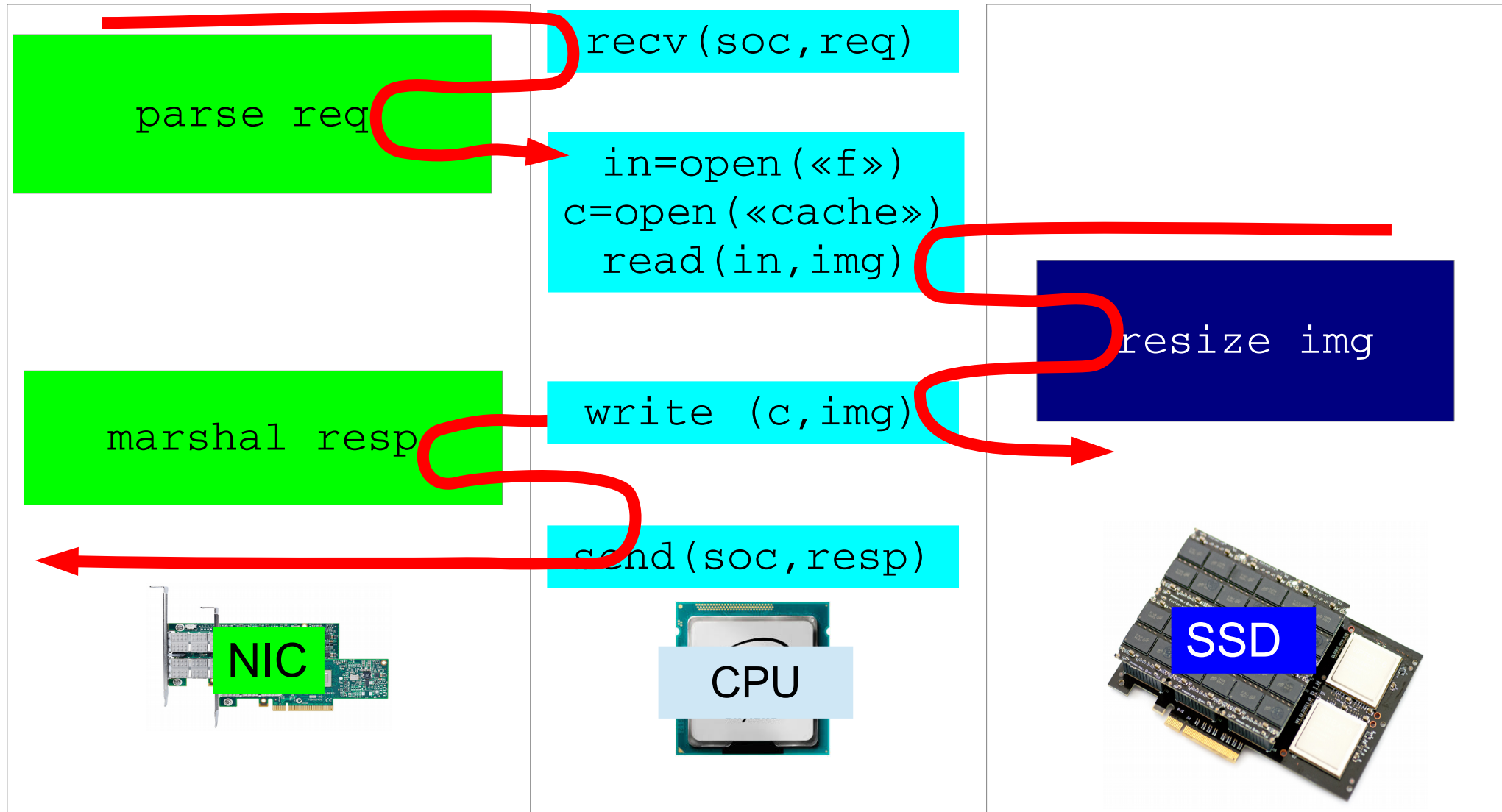
# Closer look at *get*

parse → **resize** → **store** → marshal



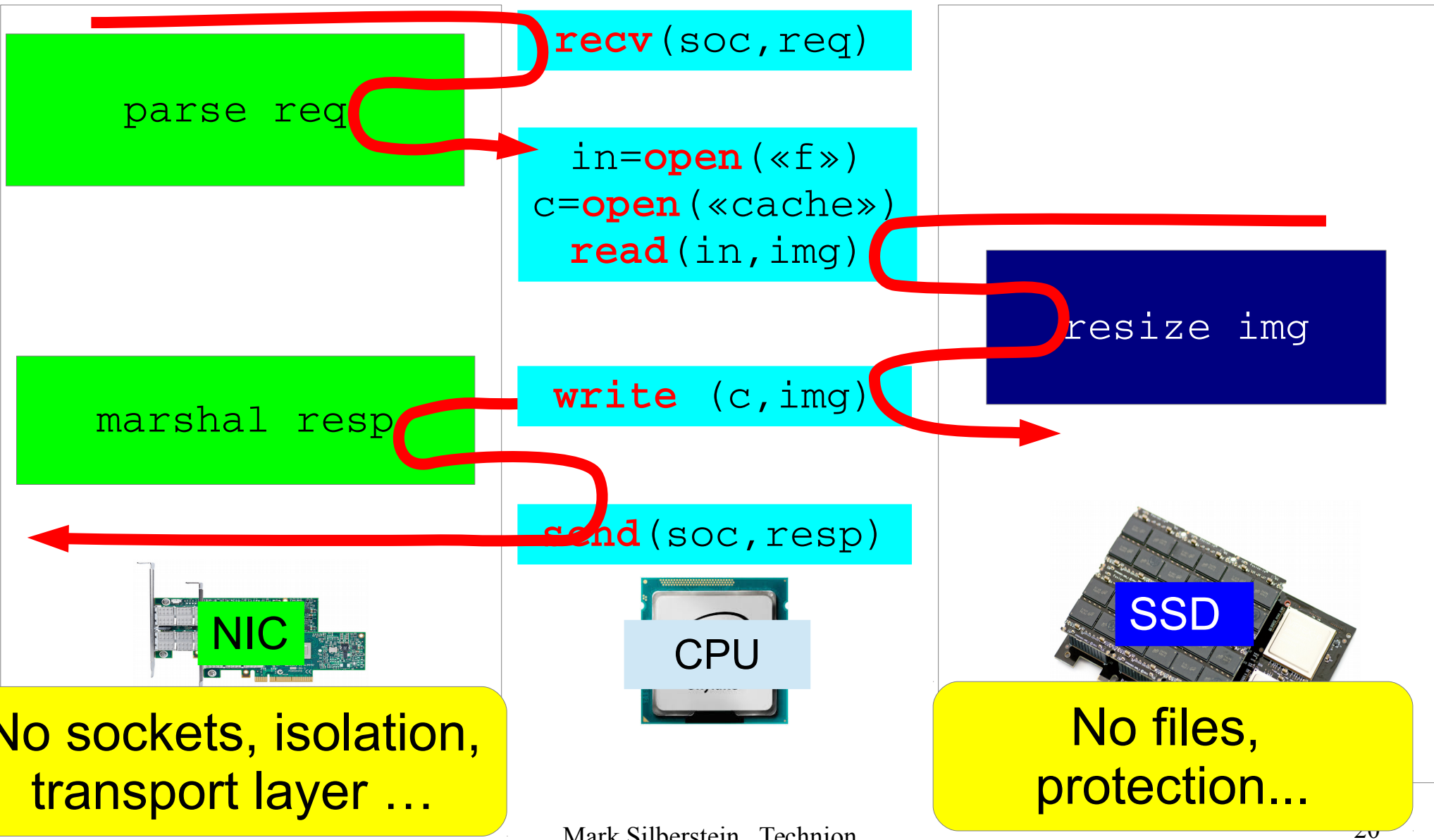
# Reality: offloading overheads dominate

get: **parse** → **resize** → **store** → **marshal**

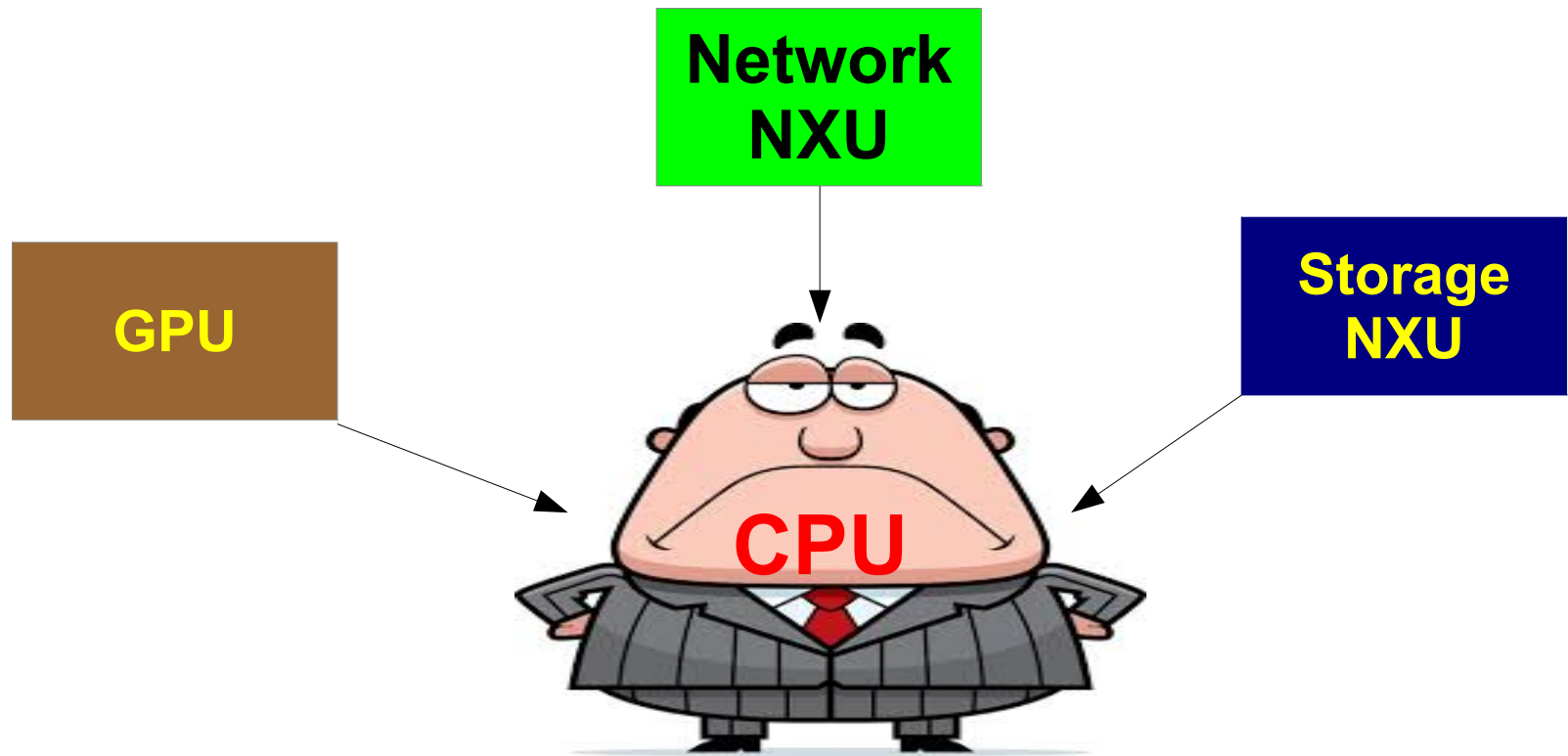


# NXUs use CPU to access I/O abstractions!

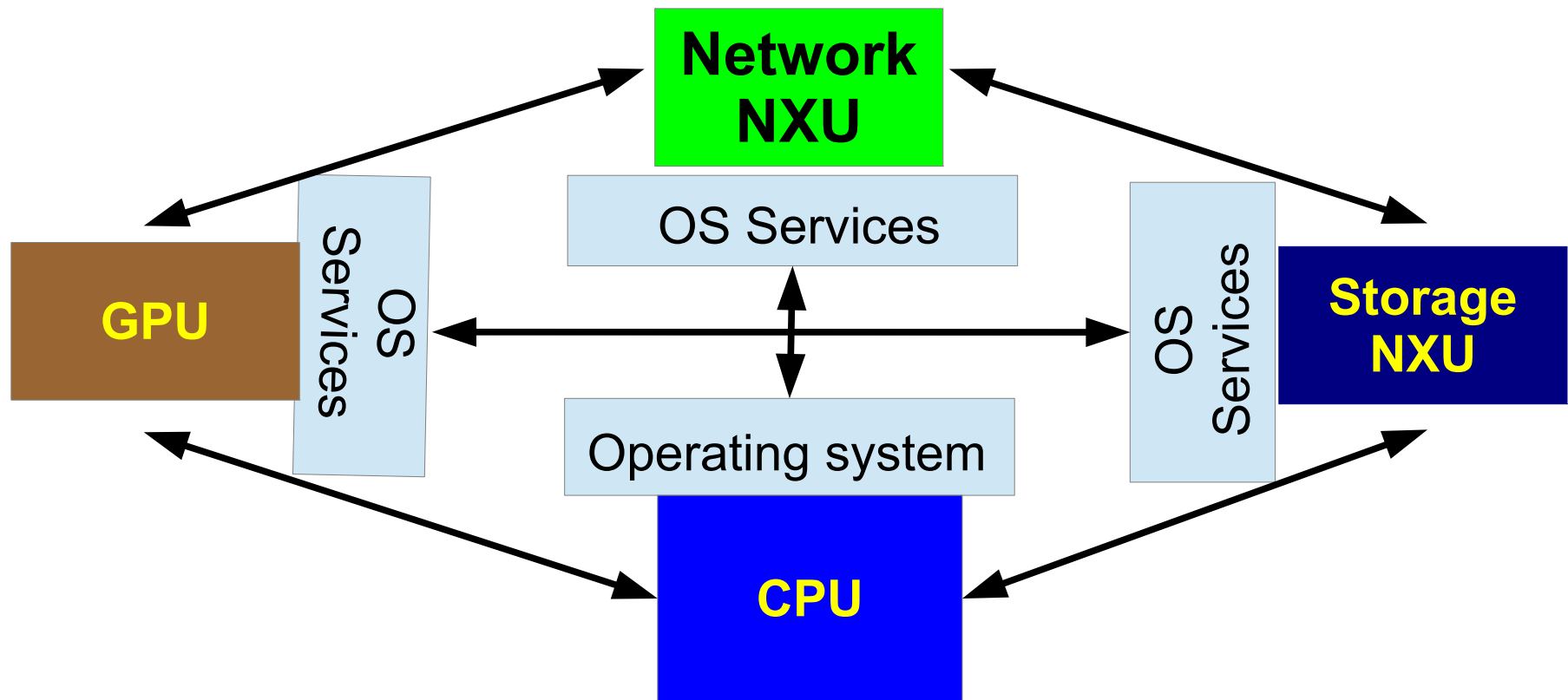
get: **parse** → **resize** → **store** → **marshal**



# **THE** problem: OS architecture is CPU - centric



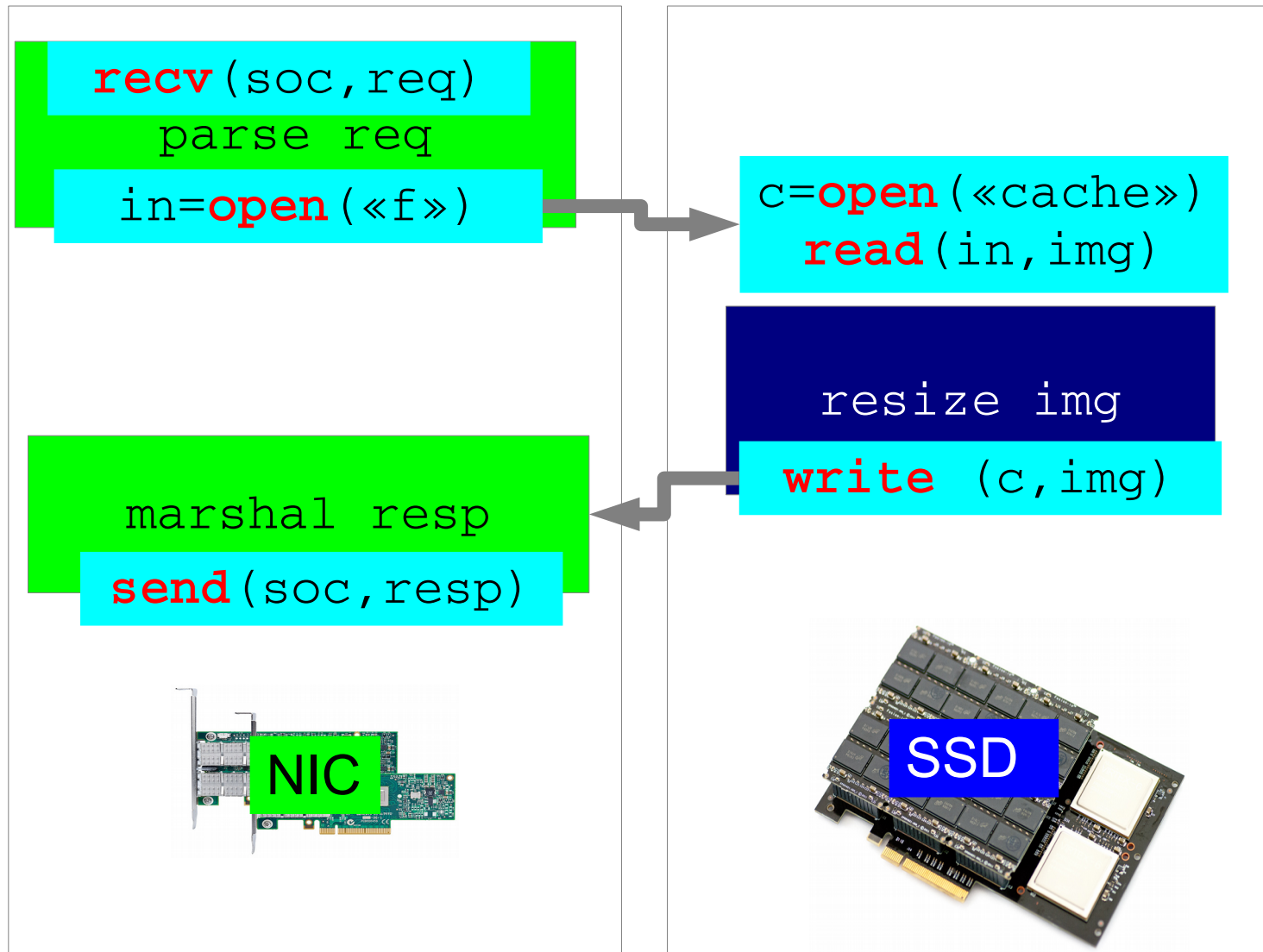
# *OmniX*: accelerator-centric OS architecture



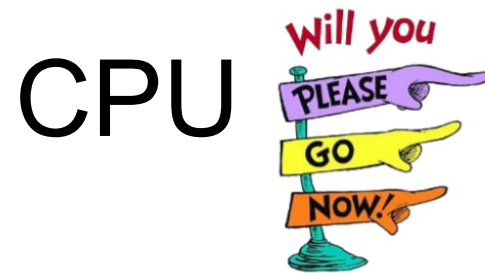
Hardware is already here (almost)

# Wouldn't it be lovely?

get: **parse** → **resize** → **store** → **marshal**



## Part 3: Subito non-CPUtto



# OmniX design choices



# NXU hardware:

## What does the future hold?

- Q:General purpose computations?
- Q:Support for *self-management*: Interrupt handling, VM management, privileged execution?
- Q:Discrete or integrated?
- Q:Memory organization?
- Q:System memory model?

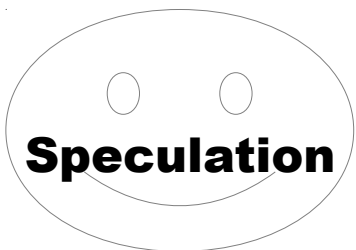
# NXU hardware:

## What does the future hold?

- Q:General purpose computations?
- Q:Support for *self-management*: Interrupt handling, VM management, privileged execution?
- Q:Discrete or integrated?
- Q:Memory organization?
- Q:System memory model?

# Will NXUs support self-management?

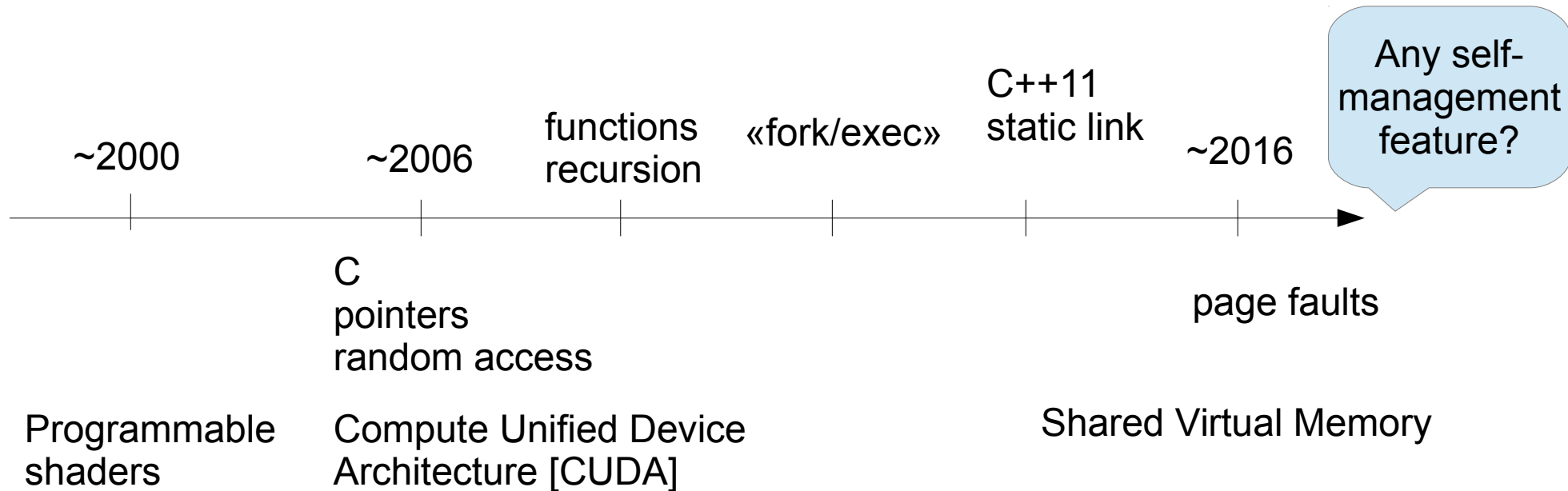
- Essential for running an OS:
  - Interrupt handling, in-device VM and address space management, privileged execution



Self-management is unlikely in the next generations of NXUs

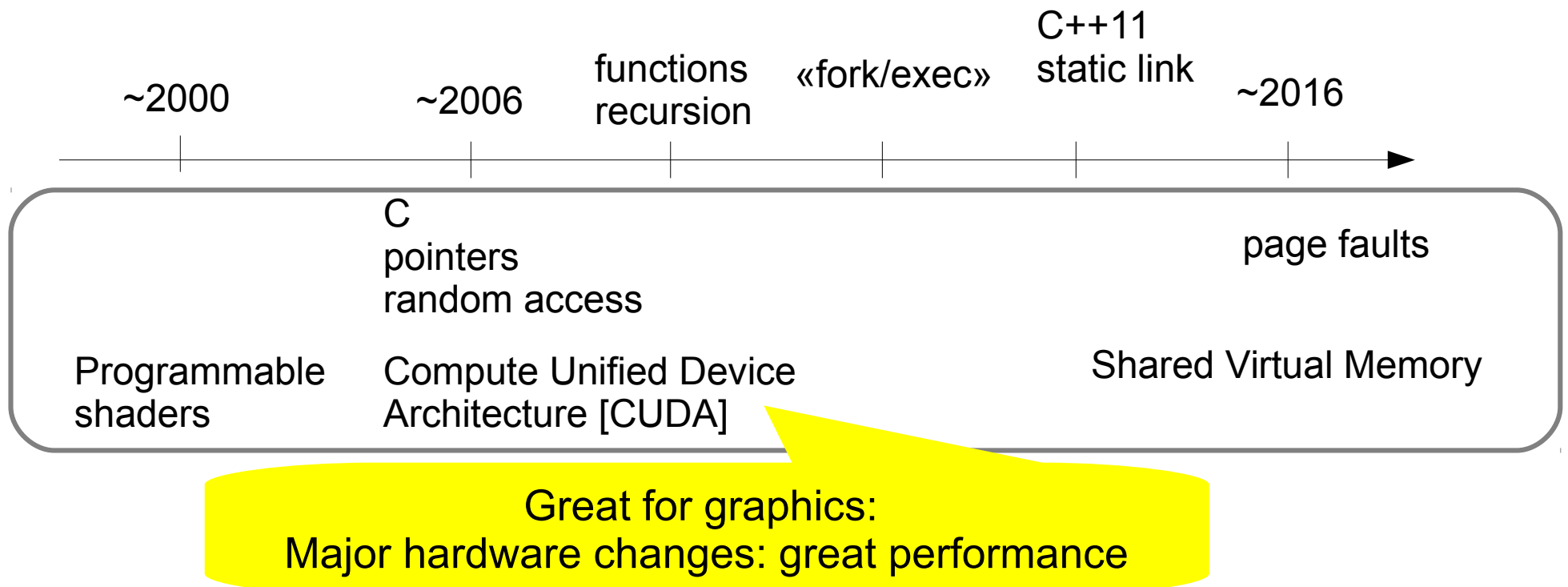
# Lets learn from GPGPUs

- Emerged as a *hack*, then endorsed by NVIDIA
- Dramatic programmability improvements



# Lets learn from GPGPUs

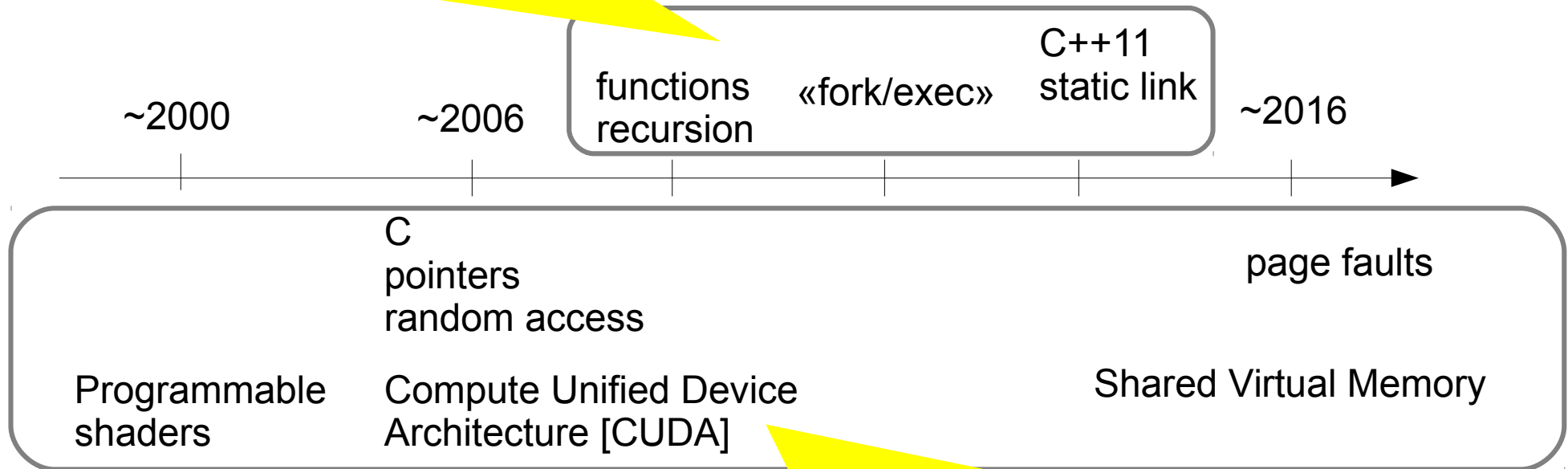
- Emerged as a *hack*, then endorsed by NVIDIA
- Dramatic programmability improvements



# Lets learn from GPGPUs

- Emerged as a *hack*, then endorsed by NVIDIA

- Drawbacks
  - Not needed for graphics:
  - Minor hardware changes, Performance so-so!



Great for graphics:  
Major hardware changes: great performance

Exceptions: double precision, precise exceptions

# Support for self-management in GPUs?

- Improves graphics performance — NO!
- Requires major hardware changes — YES!
- Guess what the answer is (and probably will be)

# Emerging NXUs: similar piggyback on high-end I/O hardware

- Mellanox Innova: an FPGA glued into Connect-X4 HCA as a bump-in-the-wire
  - No self-management support
- Smart SSDs from Samsung: re-use existing ARM cores.
  - Possibly can run an OS, but normally do not



# Emerging NXUs: similar piggyback on high-end I/O hardware

- Mellanox Innova: an FPGA glued into Connect-X4 HCA as a bump-in-the-wire
  - No self-management support
- Smart SSDs from Samsung: re-use existing ARM cores.
  - Possibly can run an OS, but normally do not

Speculation

**Performance first, Programmability last**

Self-management will be added if it contributes to performance + has low hardware cost

# Emerging NXUs: similar piggyback on high-end I/O hardware

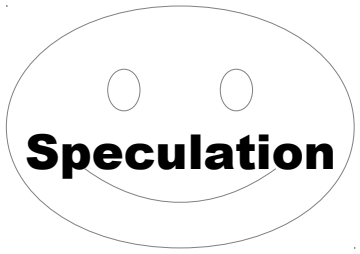
- Mellanox Innova: an FPGA glued into Connect-X4 HCA as a bump-in-the-wire
  - No self-management support
- Smart SSDs from Samsung: re-use existing ARM cores.
  - Possibly can run an OS, but normally do not

Speculation

**Performance first, Programmability last**

Self-management will be added if it contributes to performance + has low hardware cost

**OmniX does not rely on self-management in NXUs**

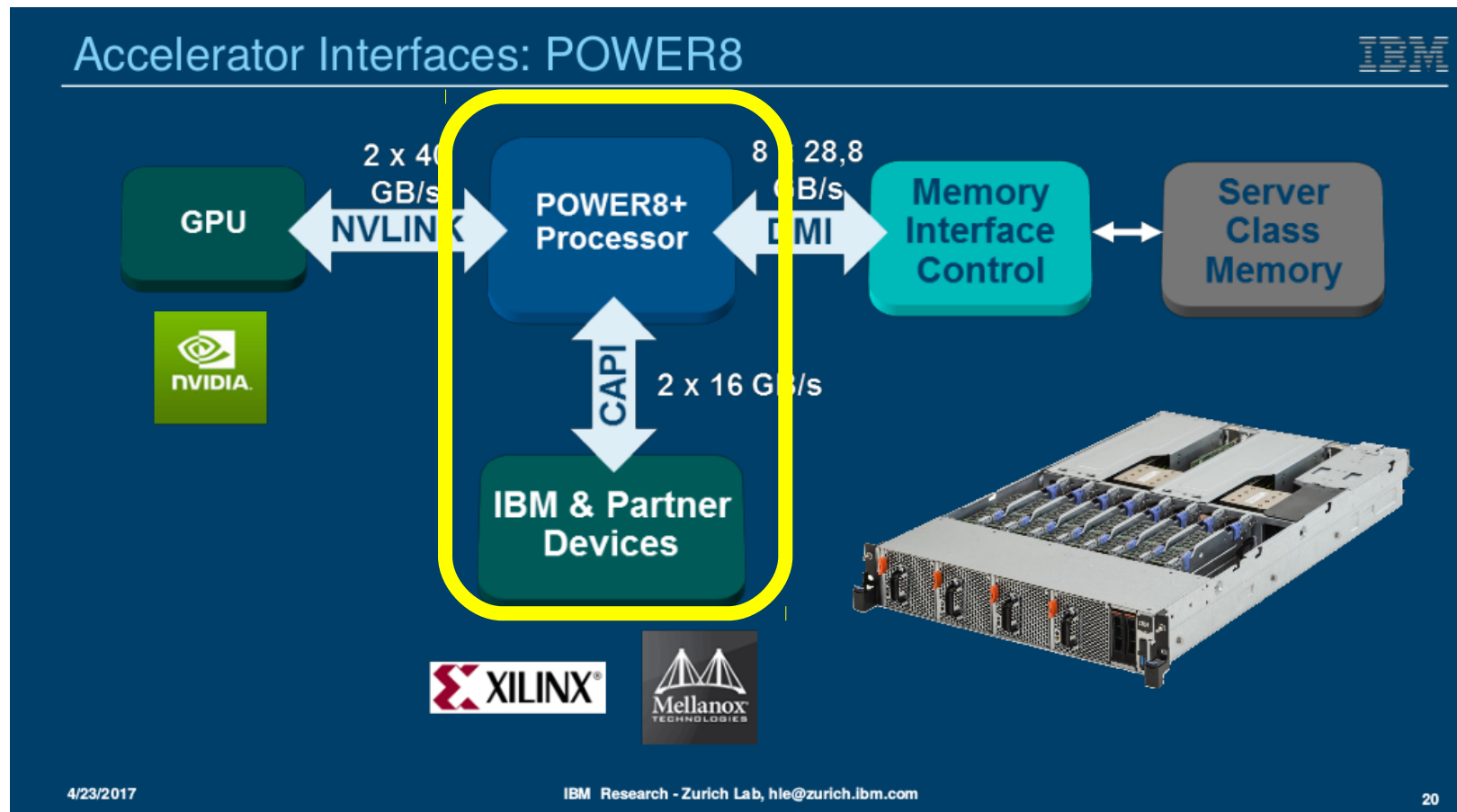


# System memory model

- Shared *virtual memory* with the host
- Coherence + remote atomics
- Extreme NUMA

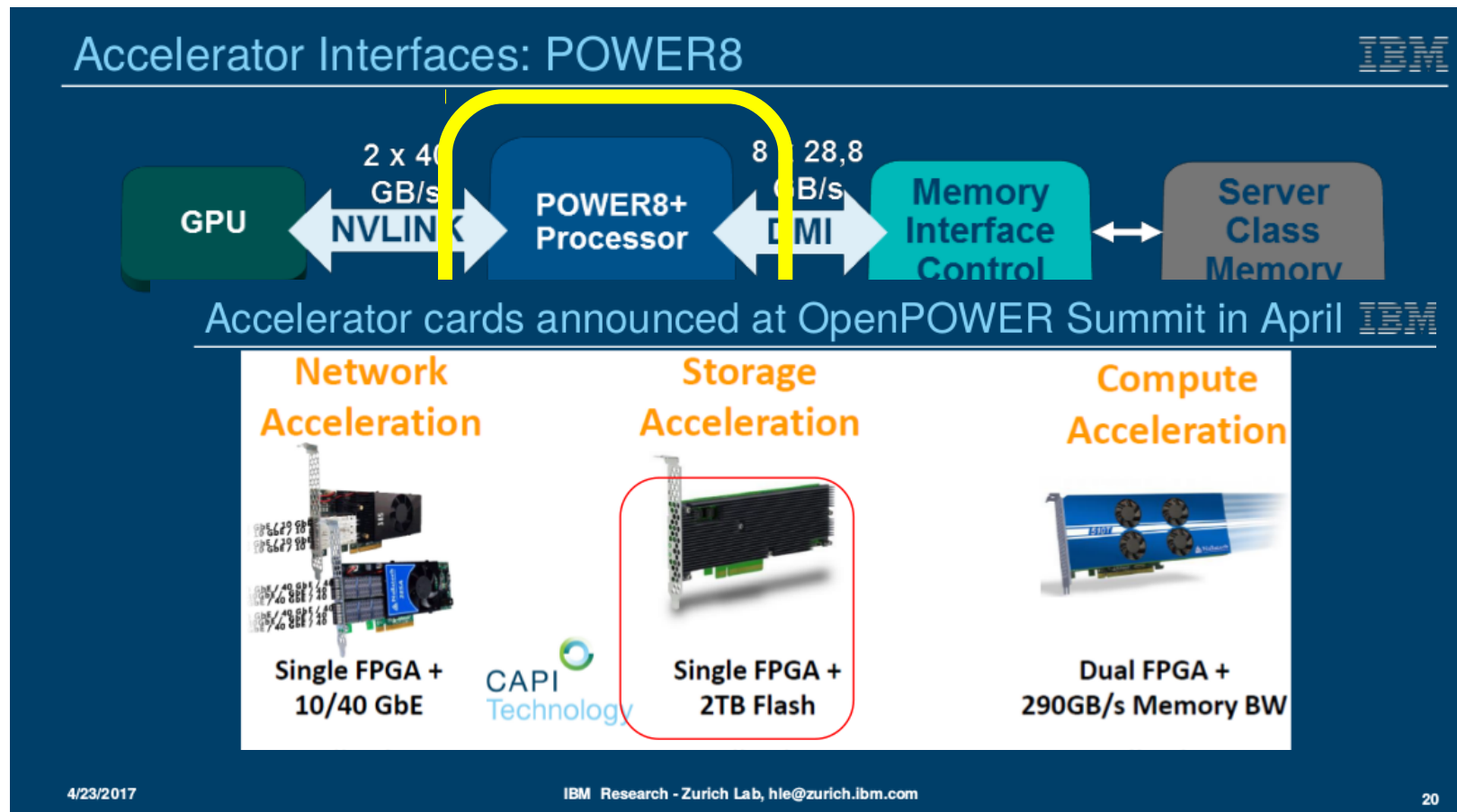
# CAPI/OpenCAPI/CCIX...

- Emerging chip-to-chip interconnects add support to VM and coherence



# CAPI/OpenCAPI/CCIX...

- Emerging chip-to-chip interconnects add support to VM and coherence



Part 4: Tutti accelerando a capella

OmniX = OS  accelerators

# OmniX design

- Each NXU runs an optimized library OS
  - Shared socket/FD namespace
  - Shared virtual address space
  - Single application OS (unikernel)
  - Protection via SRIOV
- 
- NXUs invoke tasks and perform I/O directly on their peers, without CPU mediation
  - CPU used for setup and management

# OmniX design

- Each NXU runs an optimized library OS
  - Shared socket/FD namespace
  - Shared virtual address space
  - Single application OS (unikernel)
- NXUs invoke tasks and perform I/O directly on their peers, without CPU mediation
- CPU used for setup and management
  - Protection via SRIOV



# Why CPU mediation is bad?

- Higher latency
- CPU is the bottleneck
- Poor scalability
- Poor performance isolation

# 1. Increased I/O latency

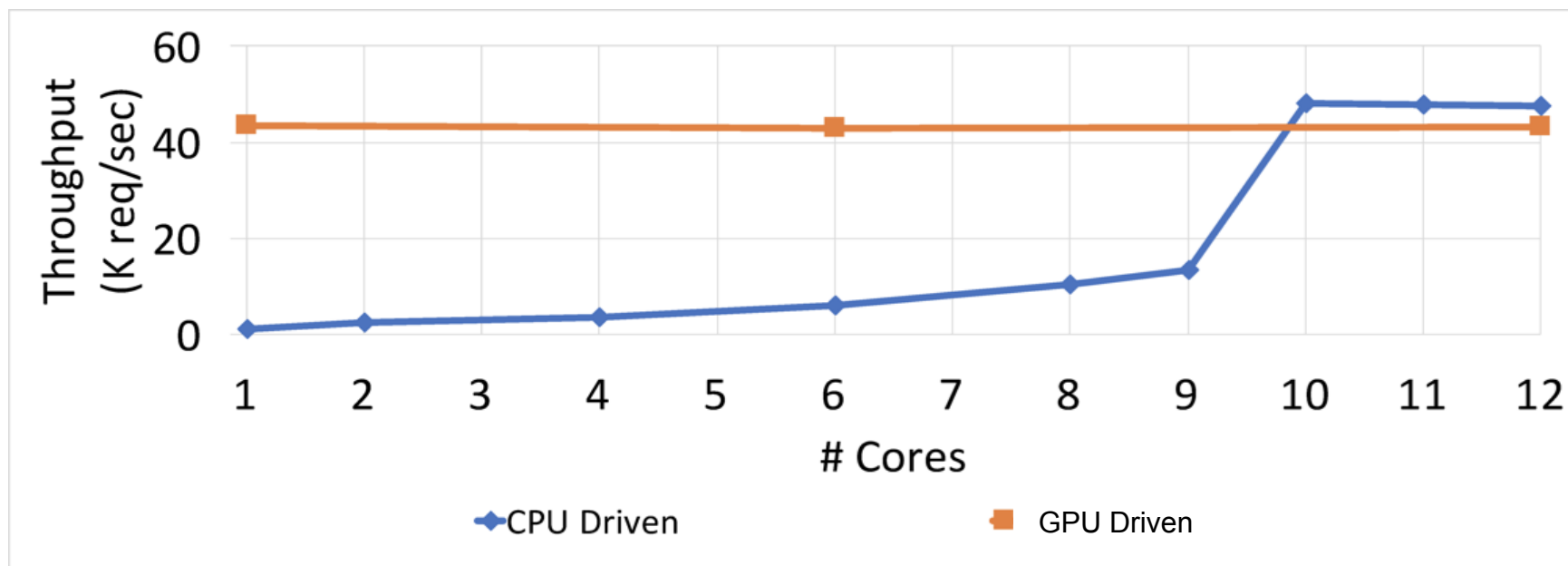
GPU-to-GPU roundtrip latency via Infiniband

GPUnet (CPU-mediated): 50 usec

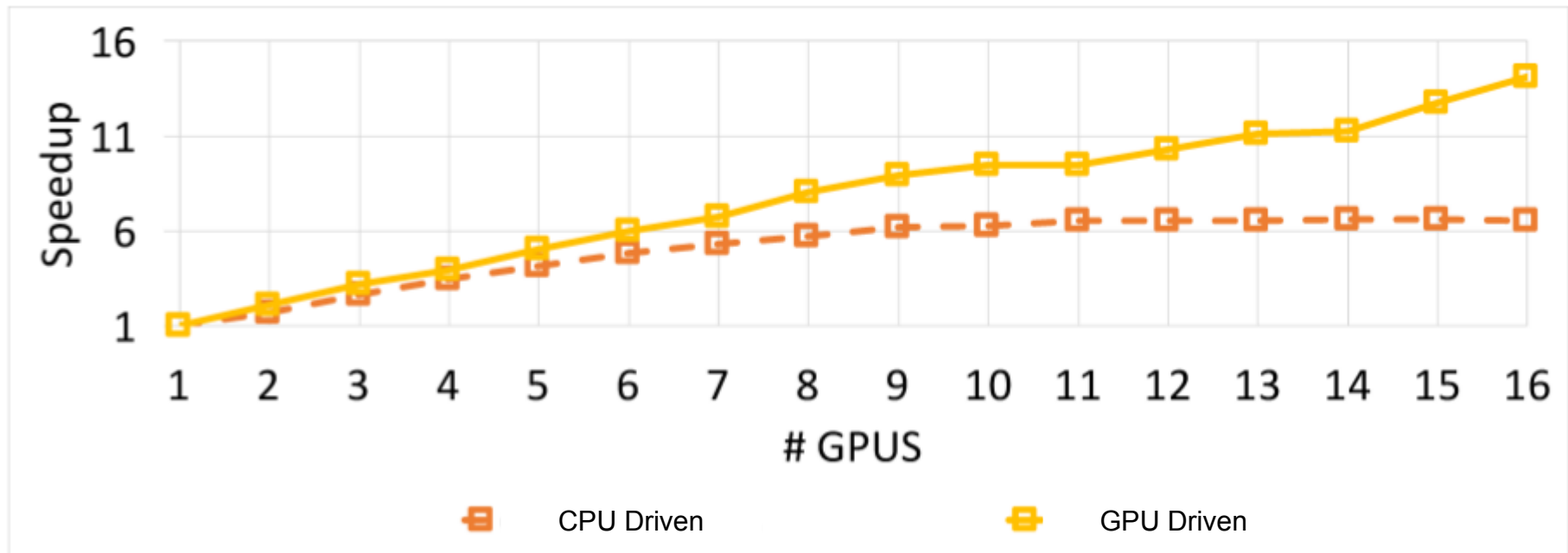
GPUrdma (NIC controlled by GPU): 10 usec

## 2. CPU is the bottleneck

- Example: Image Similarity Search, 6 GPUs
- Dataset statically partitioned, random data access, 2ms latency per request
- GPU-driven: GPU invocations without CPU involvement

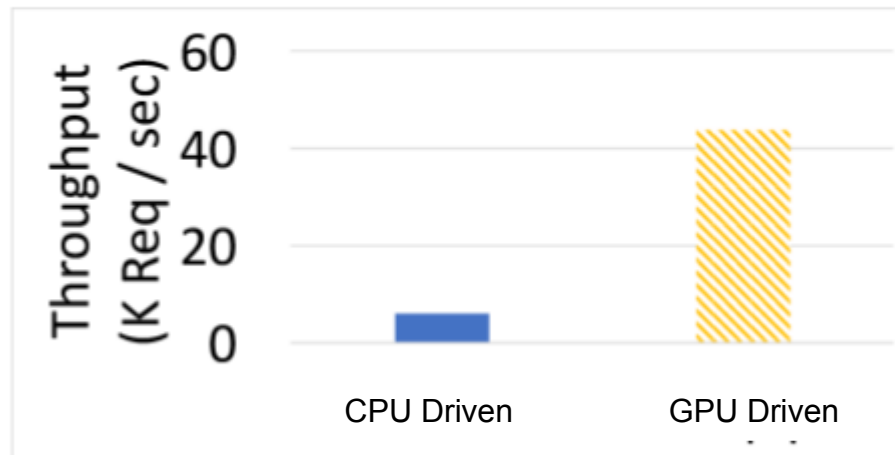


### 3. Poor scaling with #NXUs



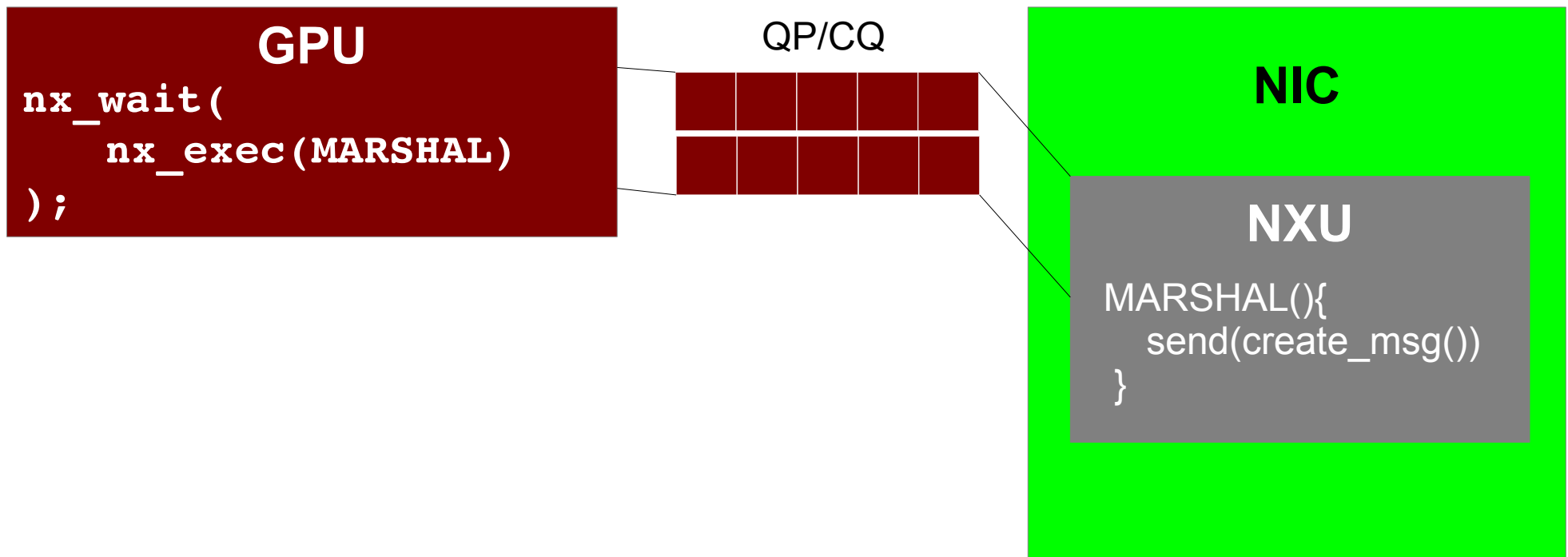
## 4. Poor performance isolation

- CPU hog running with multi-GPU server



# OmniX removes the CPU from both data and control planes

- NXUs invoke tasks and I/O operations on each other and on themselves



# Virtual memory as a capability

- NXU task/IO queues are mapped into the shared virtual address space
- Without access to the queue the NXU/device cannot be used
- Mapping/unmapping corresponds to grant/revoke: a privileged operation

# Coherent Virtual Shared Memory

- Location and transfer type agnostic
  - SSD performs send to the NIC
  - NIC passes GPU buffer to code running on SSD
- Coherence essential with local caches



# CPU's role

Do the setup  
Then leave



- Not really.
- Handle exceptions, first access to resources (files, sockets), cleanup, any privileged operations
- Runs the main program

# Open questions

- Does it require reimplementing the FS/Network stack on NXUs?

Observation: the problem is similar to that of RDMA access to file server

- Support for near-memory computations
- How do we do scheduling?
- Support for asynchronous execution on non-premptive devices

# First steps:

## OS services for GPUs/NICs

- GPUfs: file system access from GPUs  
(ASPLOS13, TOCS14, CACM15)
- GPUnet: network abstractions for GPUs  
(OSDI14, TOCS16)
- GPUrdma: native RDMA for GPUs  
(ROSS16)
- ActivePointers: In-GPU VM Management  
(ISCA16)
- GPUPipe: CPU-less network servers (under submission)
- NICA: network application accelerators  
(ongoing)

# OmniX is an ongoing work in Accelerator Computer Systems Lab



- Haggai Eran, Amir Watad, Shai Bergman, Tanya Brokhman, Vasilis Dimistas, Lior Zeno, Maroun Tork, Meni Orenbach, Shai Vakhnin, Lev Rosenblit, Marina Minkin, Pavel Lifshits and Gabi Malka

**OmniX: Accelerator-centric OS**  
is essential for  
**efficiency and programmability of**  
omni-programmable systems



**mark@ee.technion.ac.il**

backup

# NXU hardware:

## What does the future hold?

- Q: General purpose computations?  
A: Slow
- Q: Support for *self-management*:  
A: Limited or absent
- Q: Discrete or integrated?  
A: Both, but discrete will be faster
- Q: Memory organization?  
A: NUMA (NUM-B)
- Q: System memory model?  
A: Coherent Shared Virtual Memory

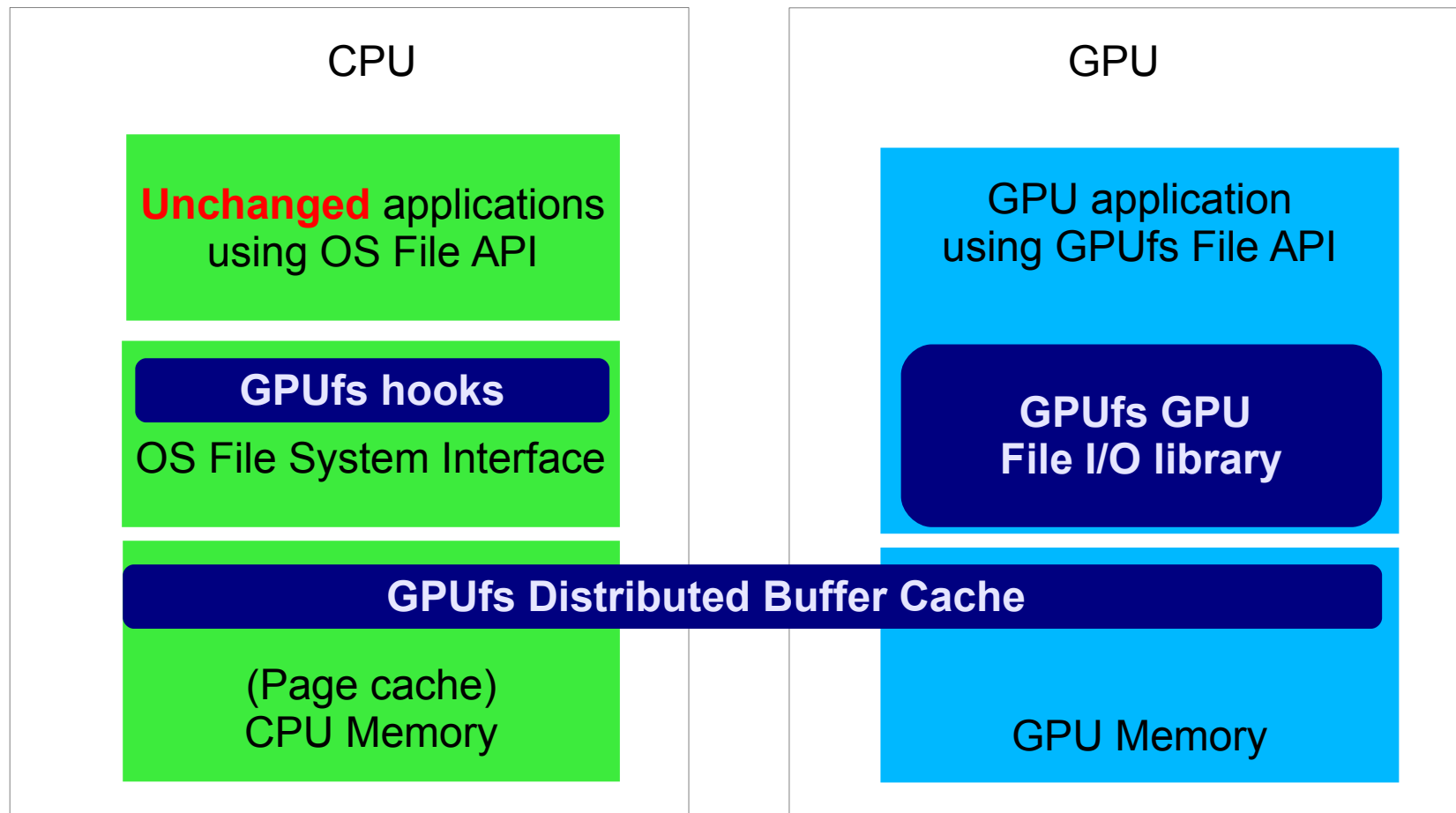
# OmniX design implications

- Q: General purpose computations?  
A: Slow → Run hardware-optimized tasks
- Q: Support for *self-management*?  
A: Limited or absent → User-space code only  
Need CPU for privileged ops
- Q: Discrete or integrated?  
A: Both, but discrete will be faster → Locality is critical
- Q: Memory organization?  
A: NUMA (NUM-B)
- Q: System memory model?  
A: Coherent Shared Virtual Memory → Use VM for protection



# On-NXU I/O services

- Internal services
  - I/O abstractions for NXU programs

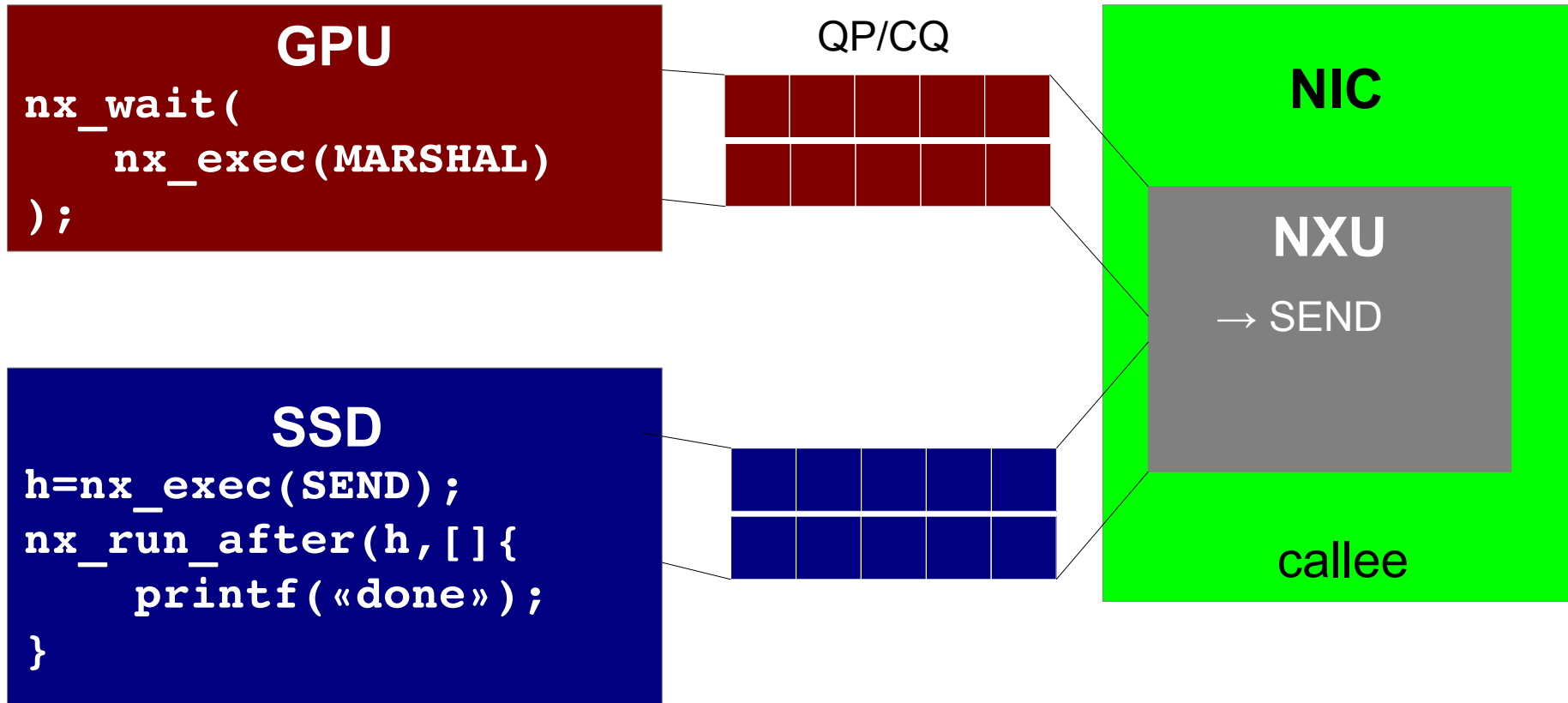


# On-NXU I/O services

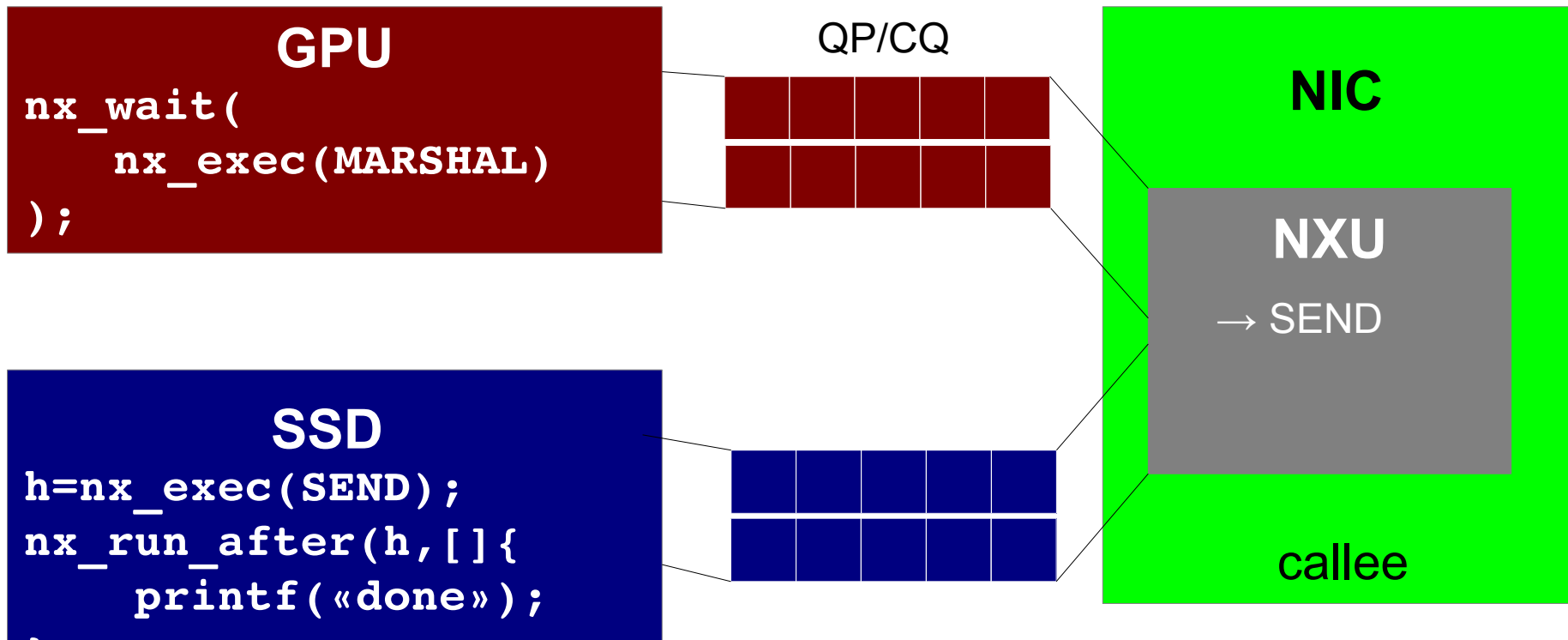
- I/O services for the hosting device
  - Networking on NICs, files on SSDs
- Interfaces
  - CQ/QP: for networking
  - mmap: for storage
- Requirements from hardware
  - Isolation and QoS
  - Security, e.g., avoid spoofing or listen on arbitrary port

# Direct local/remote task invocation

# Direct local/remote task invocation



# Direct local/remote task invocation



We implement (almost) that today with GPUs

GPUrdma: direct control of the NIC from the GPU

GPUpipe: direct task invocation among GPUs

# Case in point: CAPI improves performance!

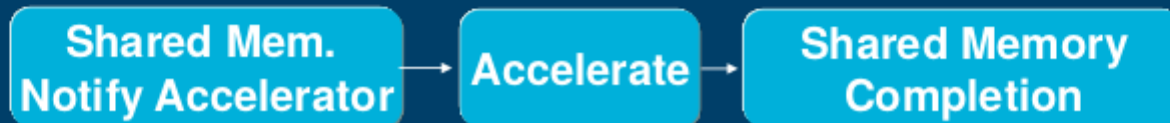
## CAPI ... Coherent Accelerator Processor Interface



### Standard I/O Model Flow



### Flow with a Coherent Model



... for a single CAPI FFT call is

- 10% higher than CPU (can be improved as the AFU is bandwidth optimized)
- 4x better compared to a PCIe version using OpenCL

