ANDREW AND ERNA VITERBI FACULTY OF ELECTRICAL ENGINEERING

ACSL Technion Accelerated Computing Systems Lab

TECHNION Israel Institute of Technology

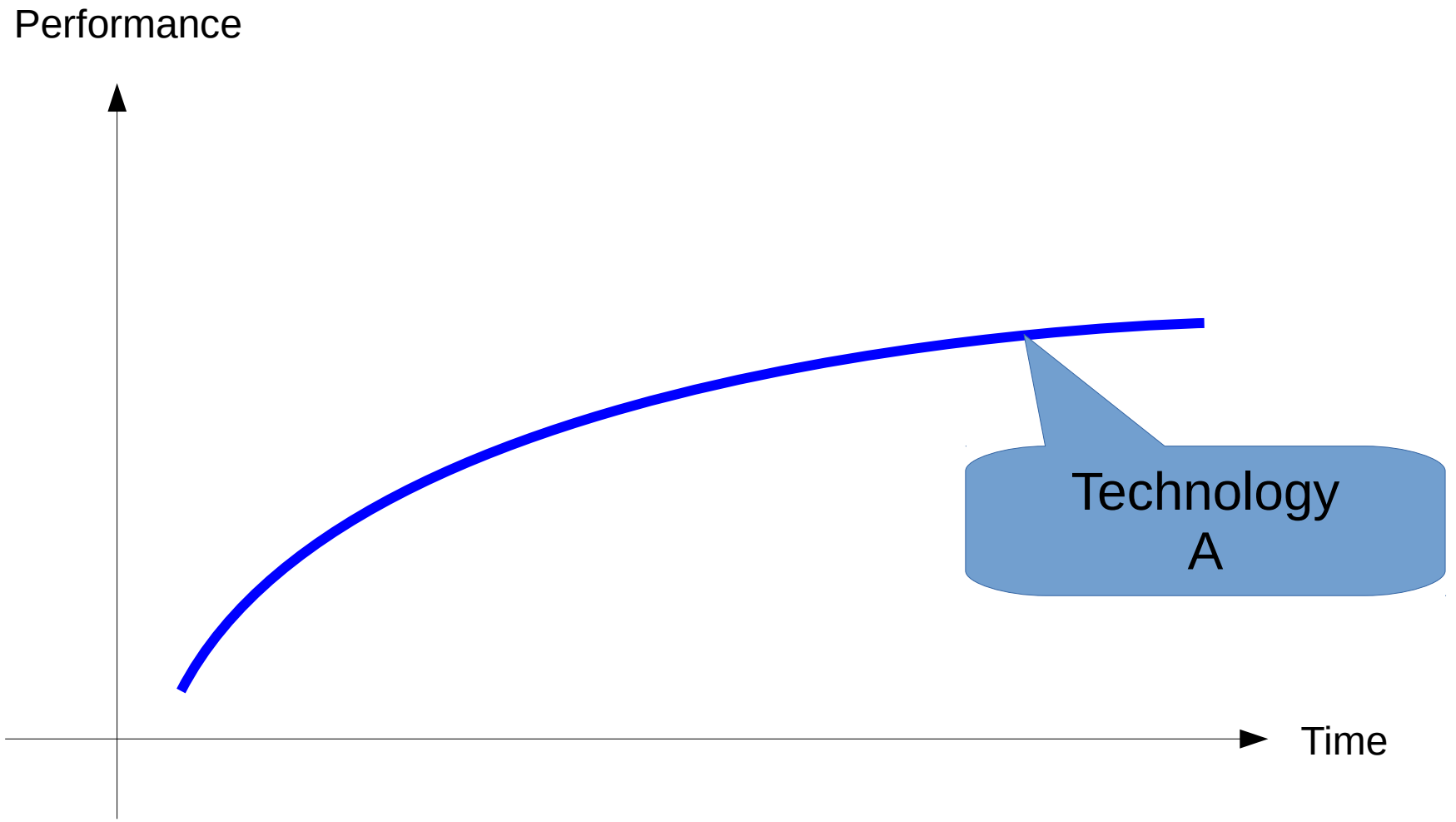
Accelerating Network Application on SmartNICs with NICA

Haggai Eran, Gabi Malka, Lior Zeno, Maroun Tork

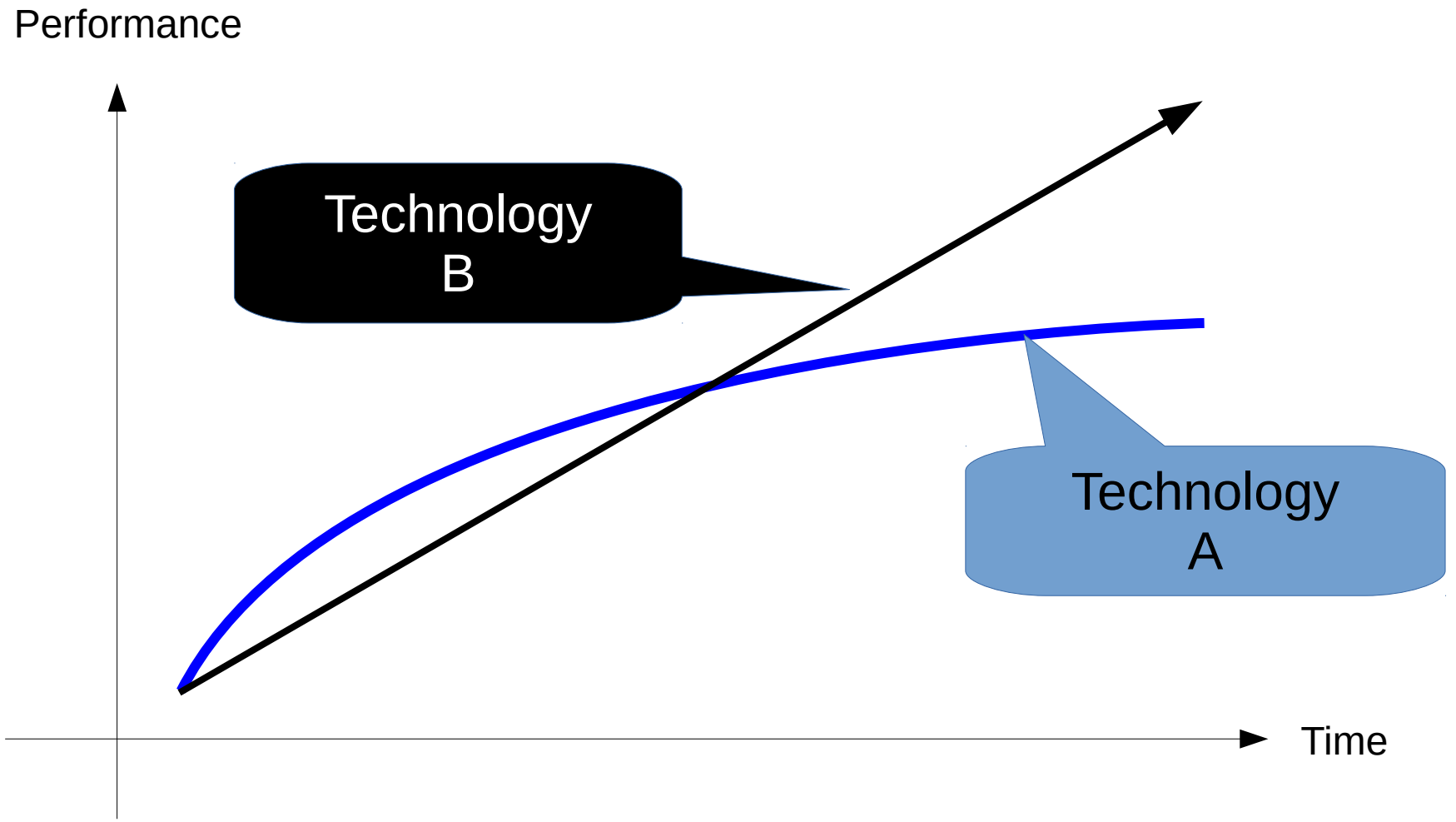
Mark Silberstein

Why do we keep building new computer systems?

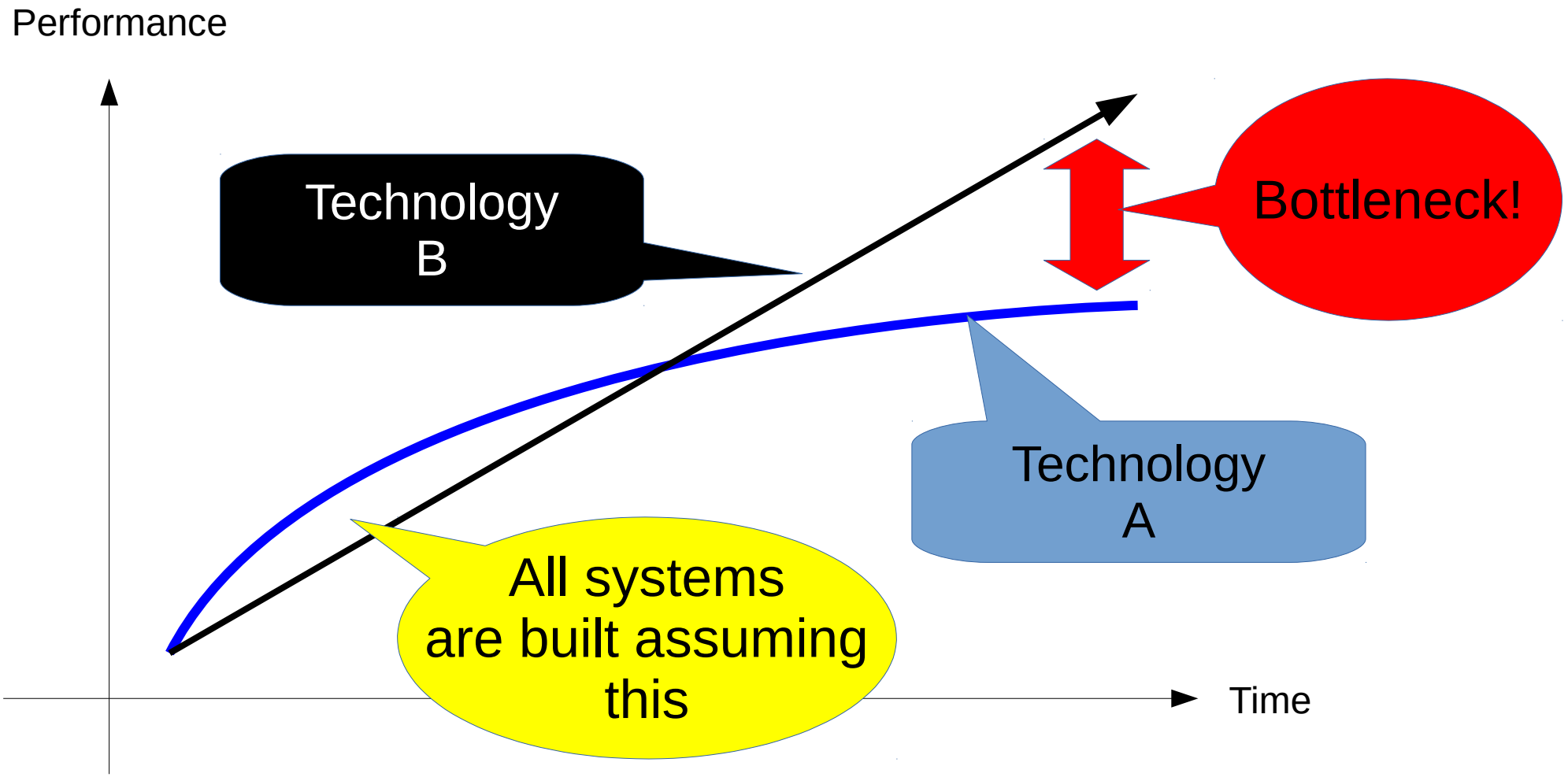
Incommensurate scaling



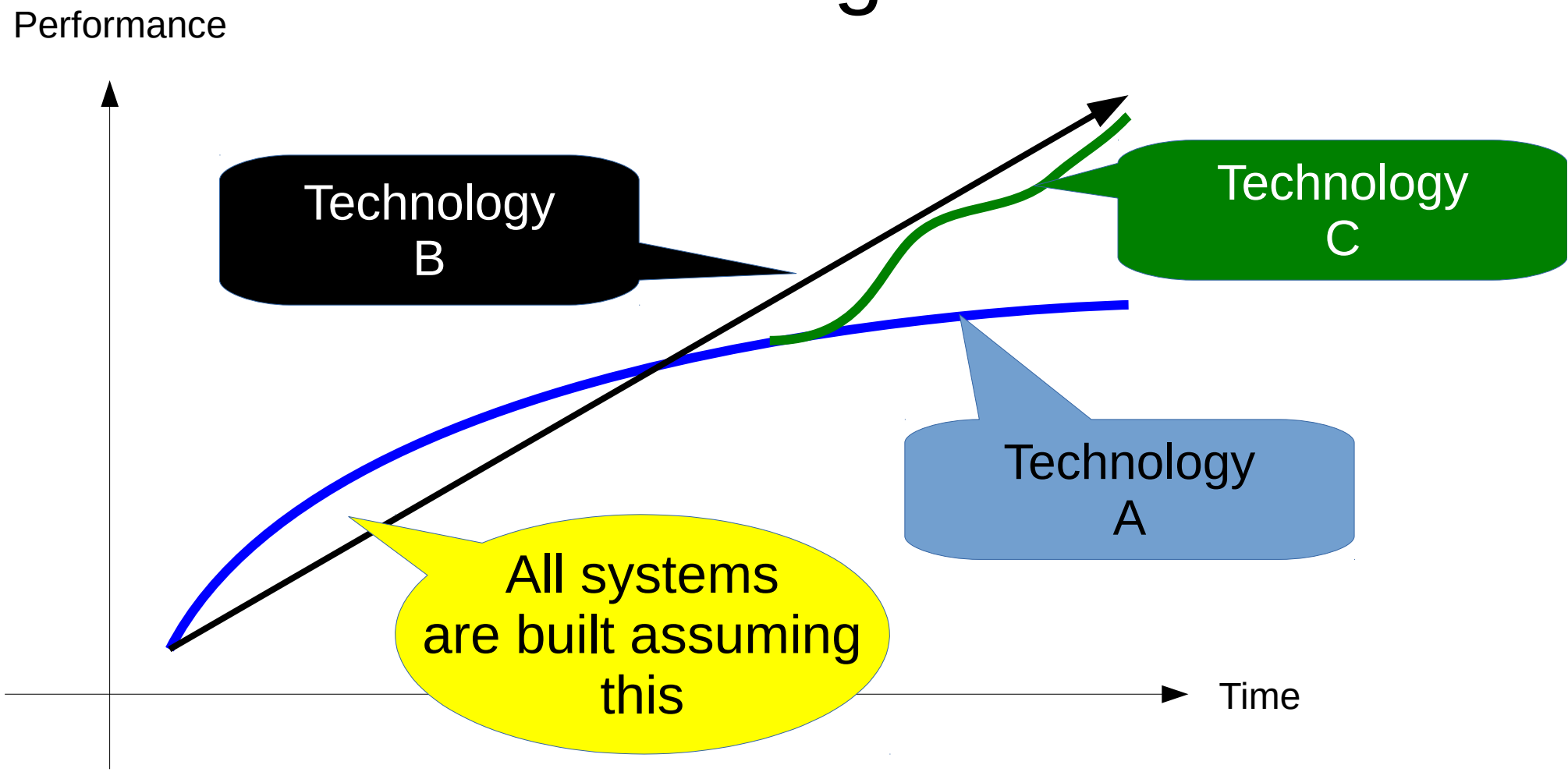
Incommensurate scaling



Incommensurate scaling

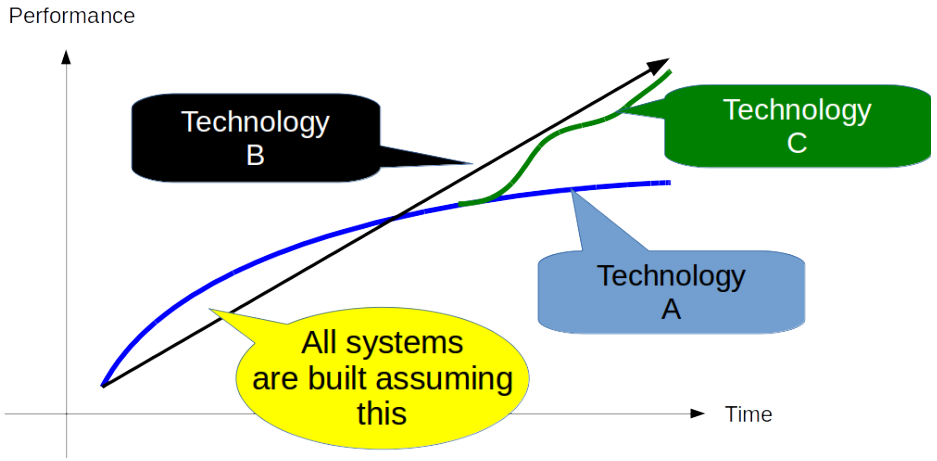


Solution: new technology with better scaling

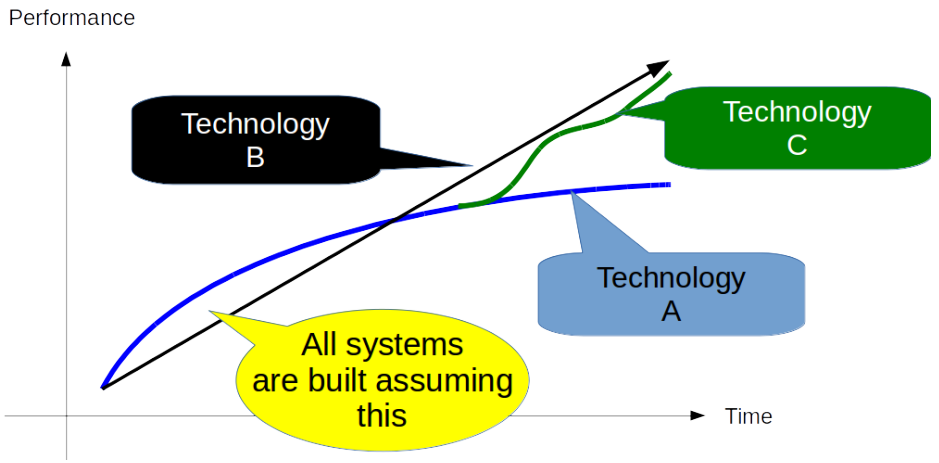


Examples

- A=DRAM bandwdith
  B=CPU capacity
  C=PIM

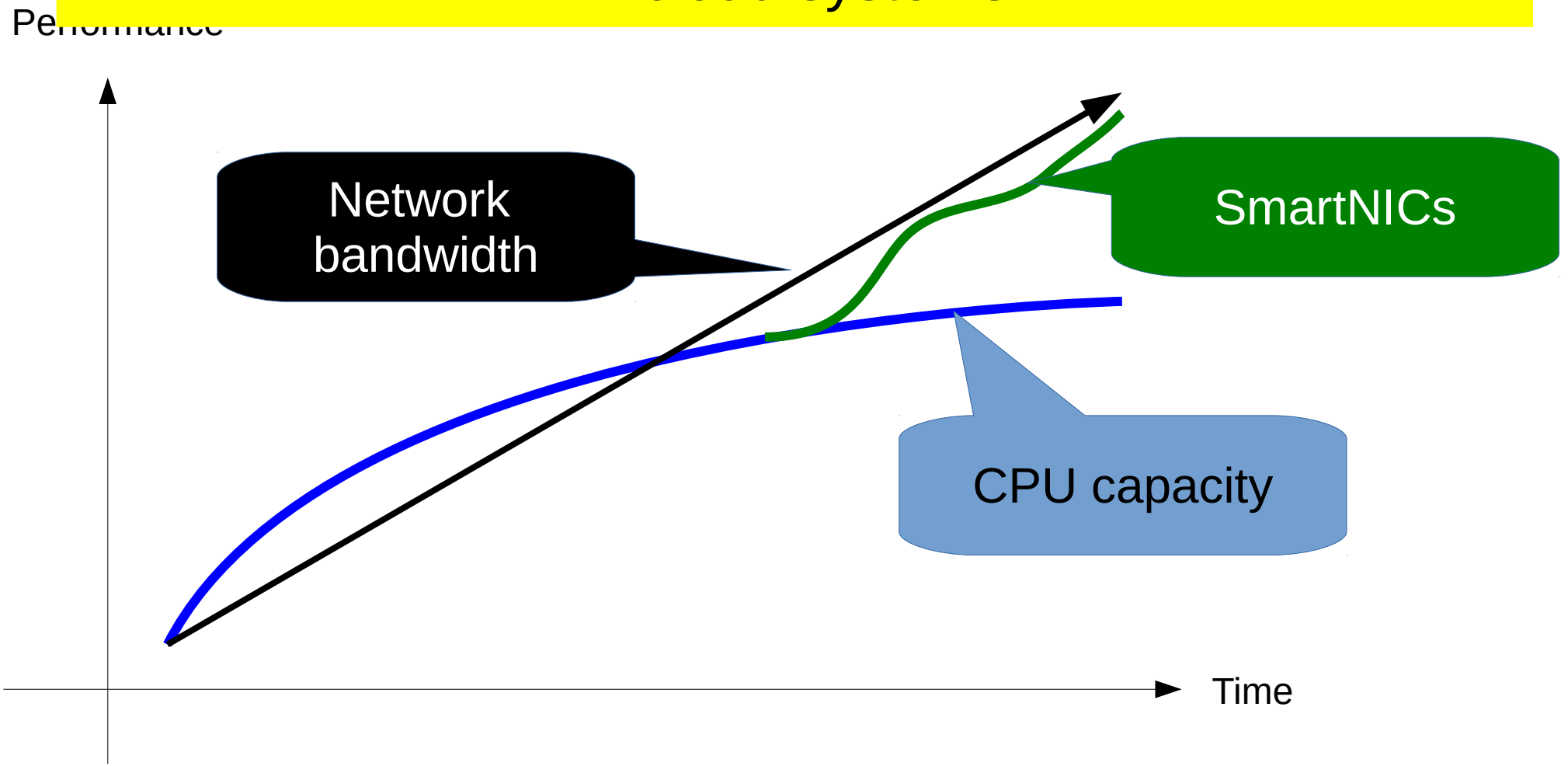


Examples



- A=DRAM bandwdith
  B=CPU capacity
  C=PIM
- A=CPU capacity
  B=I/O performance
  C=SRIOV

This talk: Software/Hardware to bridge the Network-CPU performance gap for servers in cloud systems



Agenda

- Motivation
- Challenges
- NICA
  - Programming abstractions for inline acceleration
  - Virtualization support
- Evaluation and future work

Network bandwidth scaling outpaces CPU capacity

- Shrinking CPU headroom for network application processing
  - 400Gbps network, 50 CPU cores, perfect scaling
  - 100B packets (key-value store)
  - About 300 CPU cycles per request to sustain line rate!
- As a reference
  - System call: (~250 cycles)
  - DRAM access: (~300 cycles)
  - L3 access: (~60 cycles)

Adoption of a new technology: FPGA-based SmartNICs

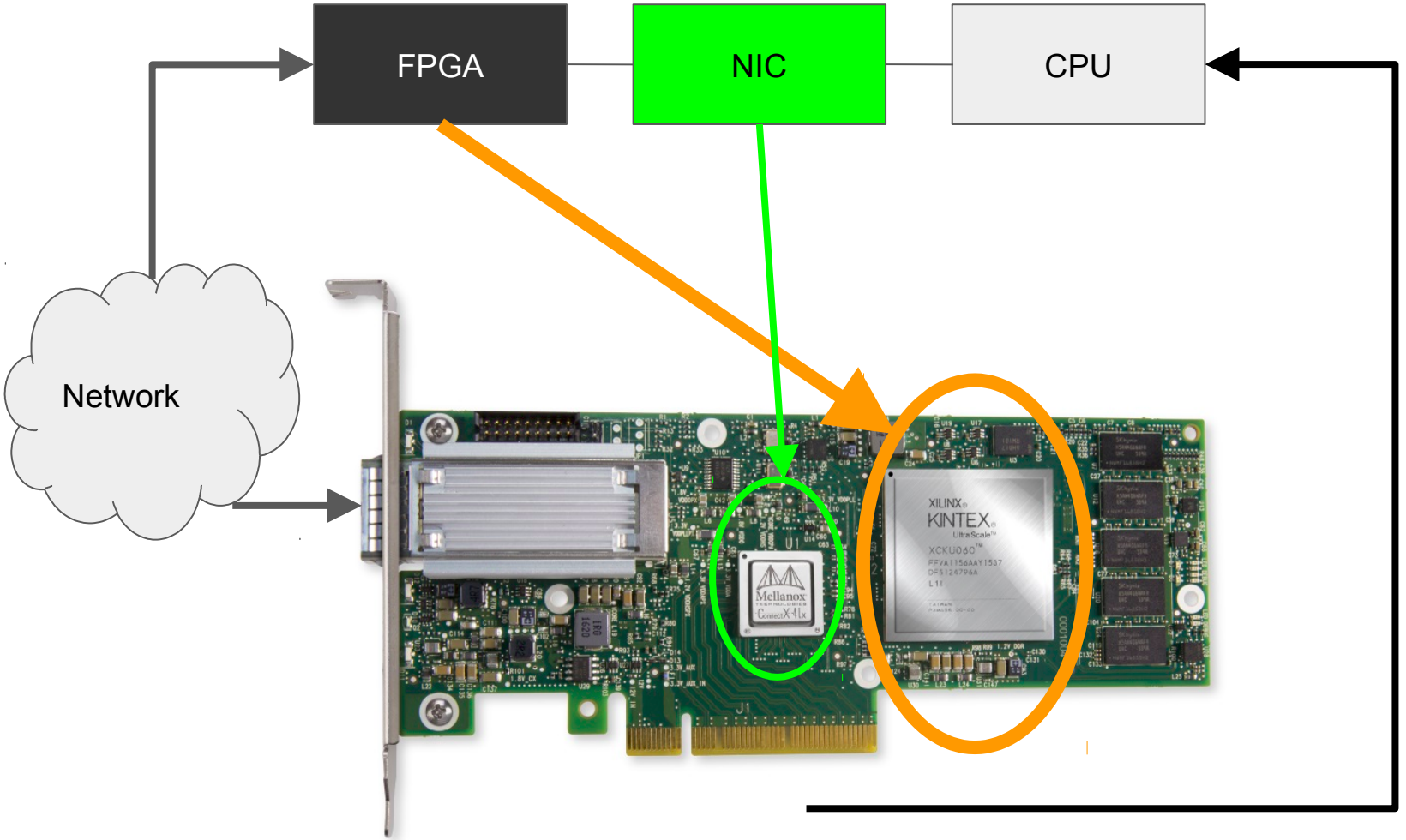
- MS Catapult Project
  - SmartNICs in every Azure server (more than million deployed)
- Other hyper-scalers:
  - China mobile, Tencent, Huawei, Selectel
- Growing number of SmartNIC vendors
  - Intel's SmartNIC is due by Feb 2019

Adoption of a new technology: FPGA-based SmartNICs

- MS Catapult Project
  - SmartNICs in every Azure server (more than million deployed)
- Other hyper-scalers:
  - China mobile, Tencent, Huawei, Selectel
- Growing number of SmartNIC vendors
  - Intel's SmartNIC is due by Feb 2019

Takeaway: Programmable NICs are here, going toward ***commoditization***

Example:
Mellanox bump-in-the-wire SmartNIC



Important properties

- ***Inline processing*** on the NIC
  - FPGA function is triggered without CPU involvement
- Efficient and flexible I/O path to the host
  - NIC ASIC provides flexible DMA
  - Support for SRIOV, RDMA
- Compatible with existing network stack

Main applications today

- Network infrastructure offloads
  - virtual networking, encryption, compression, SDN, NFV
- Stand-alone application accelerators
  - DNNWeaver, Bing, BrainWave

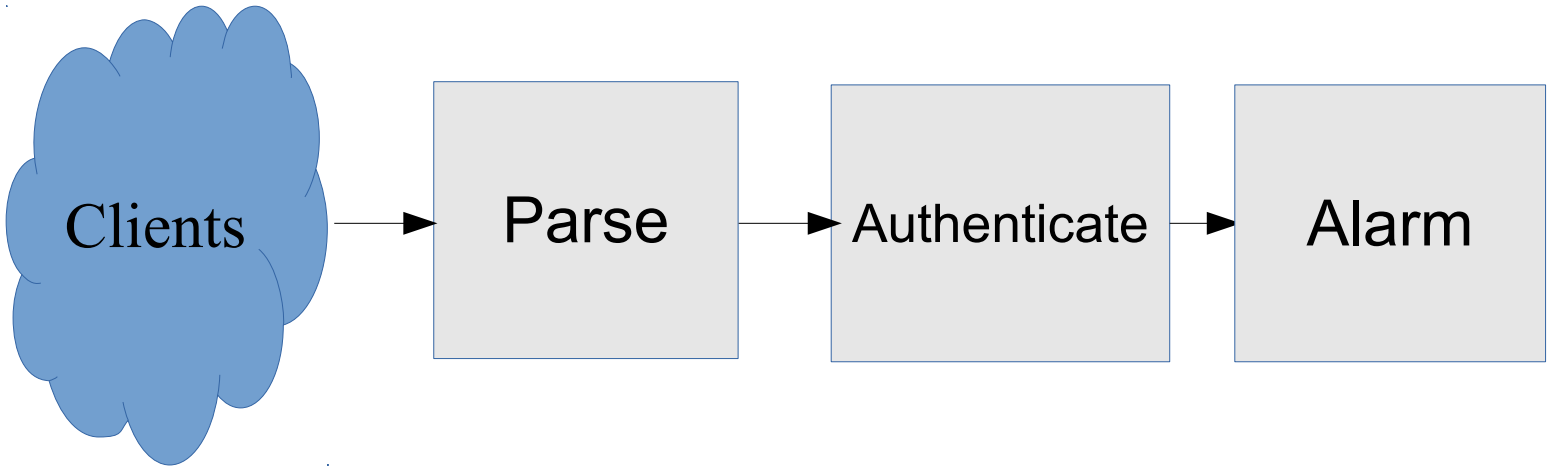
Main applications today

- Network infrastructure offloads
  - virtual networking, encryption, compression, SDN, NFV
- Stand-alone application accelerators
  - DNNWeaver, Bing, BrainWave

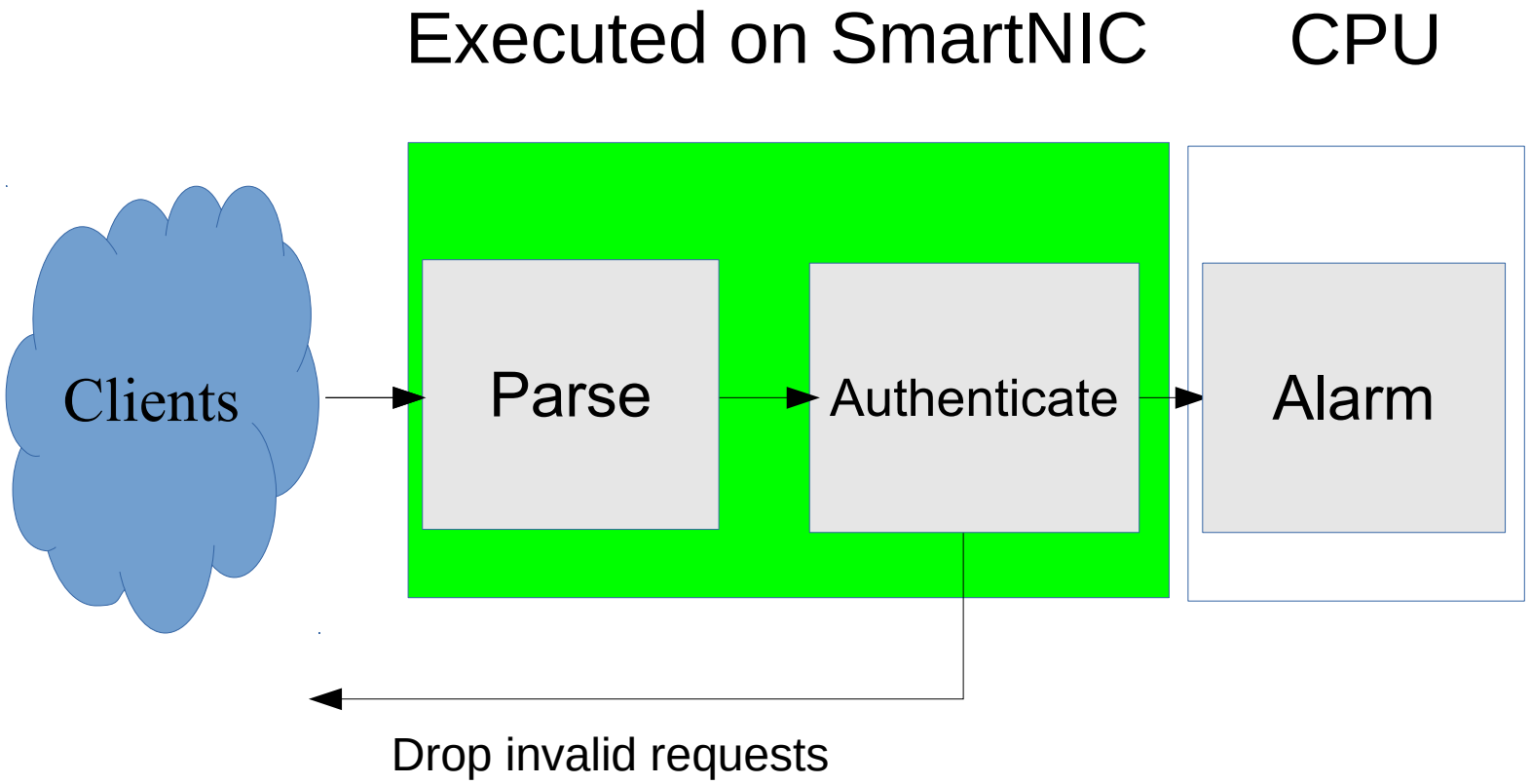
Our goal:
Server logic acceleration for cloud tenants

Example: IoT server

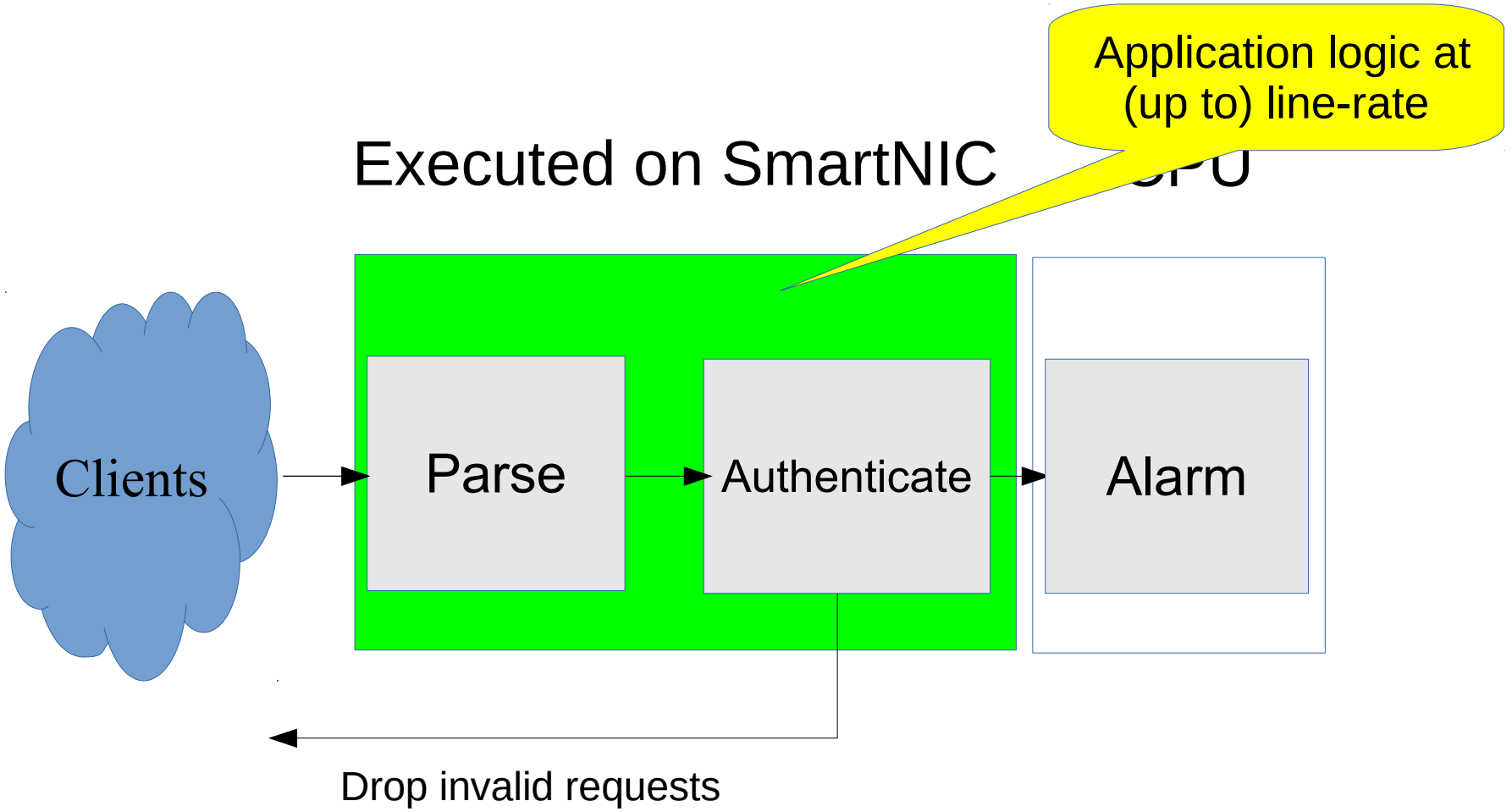
- Receive data from sensors, warn on anomaly



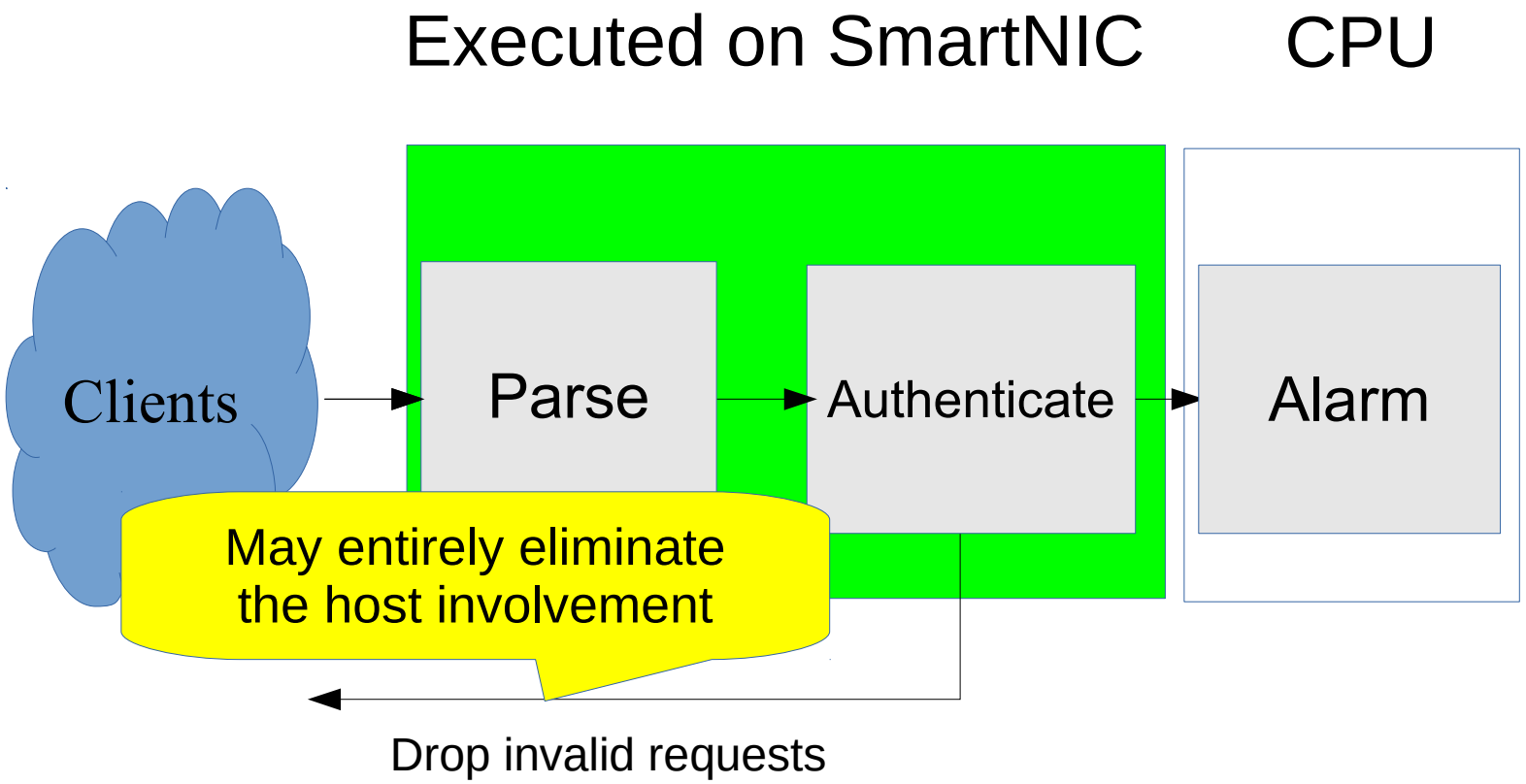
Example: IoT server **inline application** offload



Example: IoT server **inline application** offload



Example: IoT server **inline application** offload

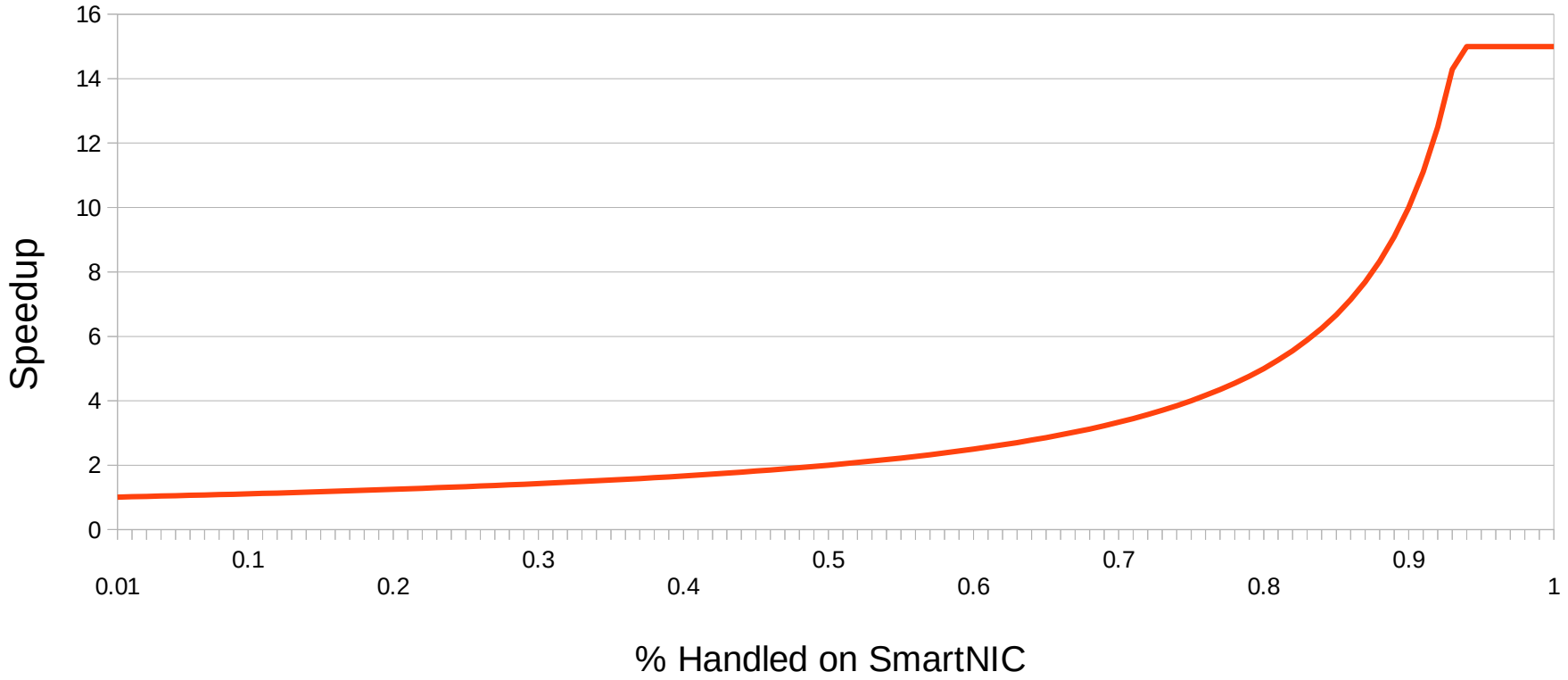


Performance benefits beyond simple acceleration

- System throughput: min(CPU_{thpt}/(1-%SmartNIC),$SmartNIC_{thpt}$)

Performance benefits beyond simple acceleration

- System throughput: min(CPU_{thpt}/(1-%SmartNIC),$SmartNIC_{thpt}$)
- Assuming $SmartNIC_{thpt}$ = 15x CPU_{thpt}



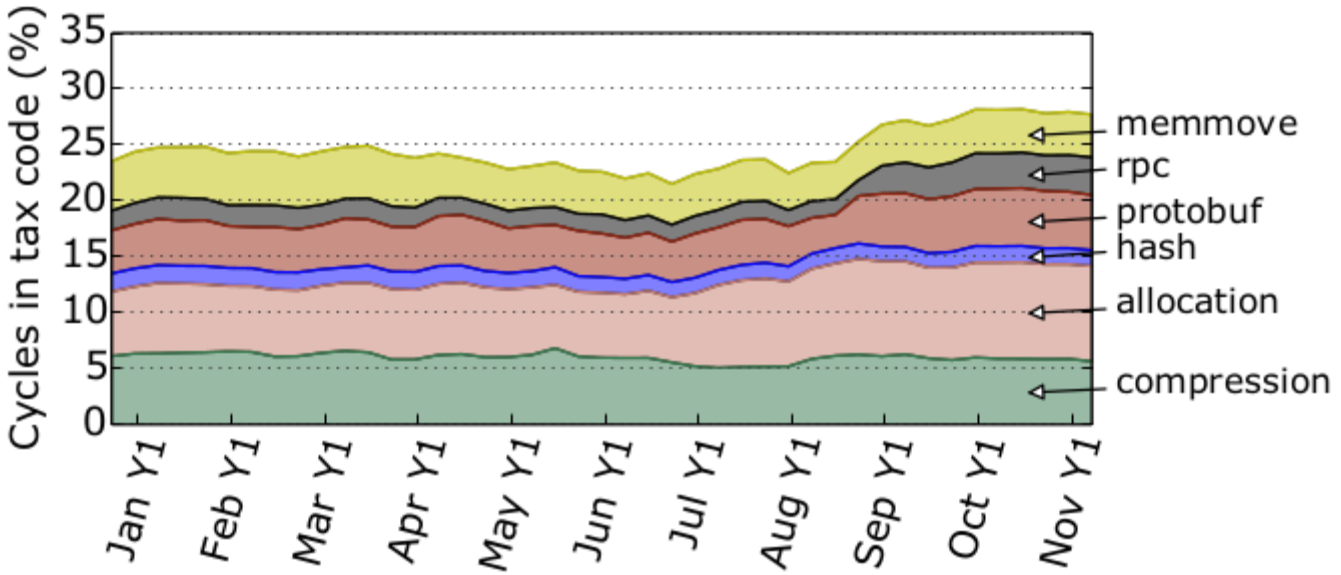
Applications for inline processing

Category	Example	SmartNIC role
Filtering	• Data analytics: sketching, classification, inference; • Caching	Fast path + response
Transformation	• (De)serialization • Batching • Layout optimization	In-transit layout change
Steering	• Load balancing • Affinity-aware placement • Direct peripheral access	Determines destination while considering system and app-specific factors
Generation	• Consensus • MapReduce Shuffle • Source coding	Constructs outgoing network messages, sends them to app-specific destinations

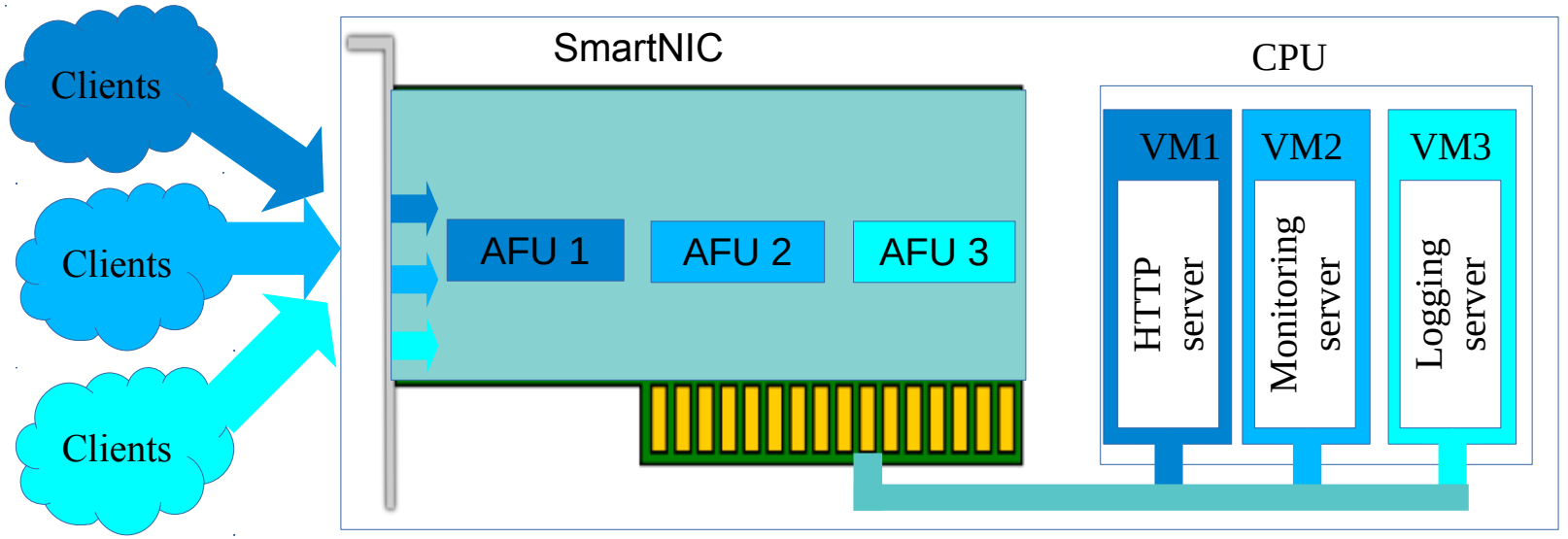
Accelerating cloud servers: common building blocks

Profiling a warehouse-scale computer

Svilen Kanev[†] Harvard University
Juan Pablo Darago[†] Universidad de Buenos Aires
Kim Hazelwood[†] Yahoo Labs
Parthasarathy Ranganathan Google
Tipp Moseley Google
Gu-Yeon Wei Harvard University
David Brooks Harvard University



Usage model: On-demand Accelerator Function Units (AFUs)



- AFUs are developed by cloud vendors
- Deployed on-demand
- Can be shared by VMs
- Emerging trend: AFU marketplace

Summary so far

- FPGA-based SmartNICs are becoming commodity
- Currently used for infrastructure offloads
- Our claim: SmartNICs can provide significant benefits for **server applications**
- On-demand deployment of AFUs developed by cloud vendors to save FPGA development cost

What is missing?

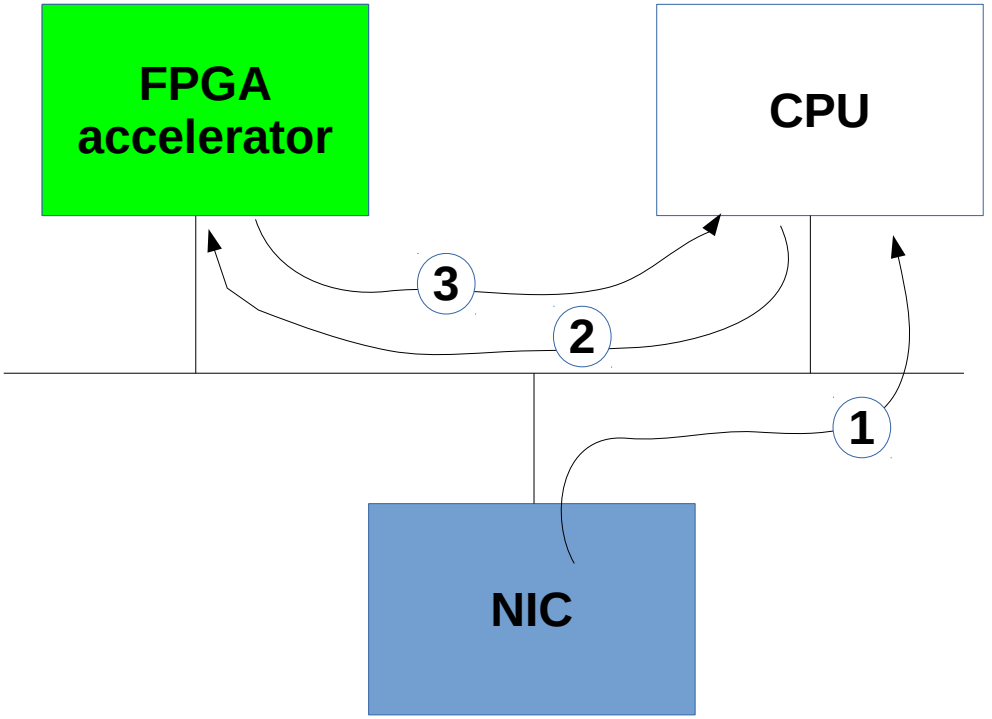
Challenges

- No adequate **OS abstractions** for **inline** acceleration

- No adequate mechanisms to **share inline accelerators among cloud tenants**

Existing accelerator-as-coprocessor model is not suitable

1 Receive data
2 Invoke offload
3 Retrieve data



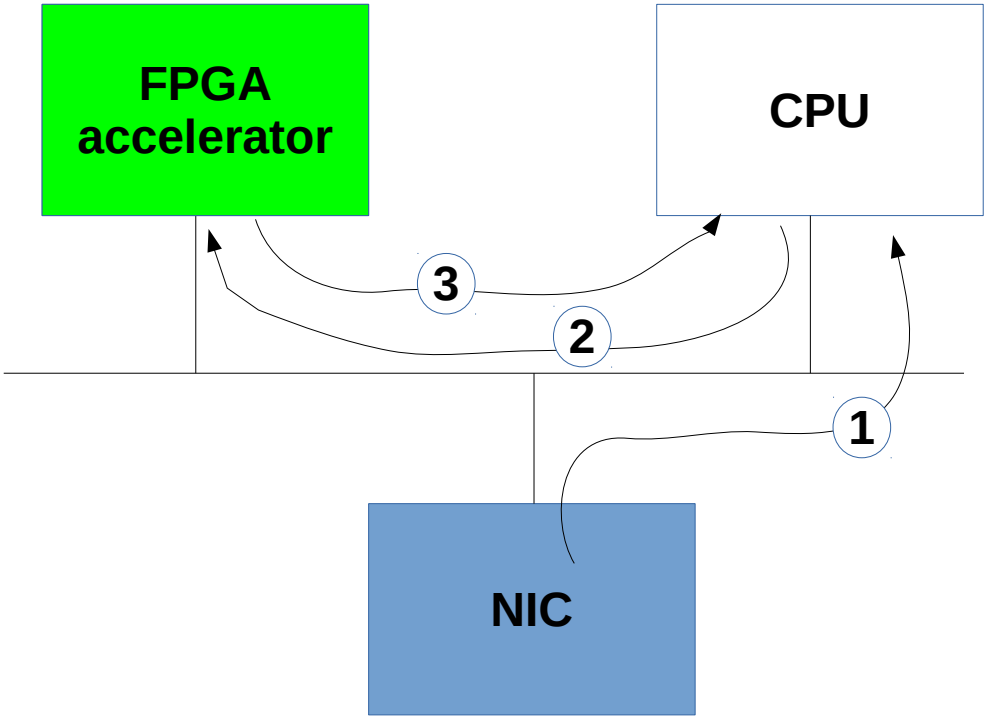
Existing accelerator-as-coprocessor model is not suitable

2-5 usec

5-7usec

1 Receive data
2 Invoke offload
3 Retrieve data

2-5 usec



Existing accelerator-as-coprocessor model is not suitable

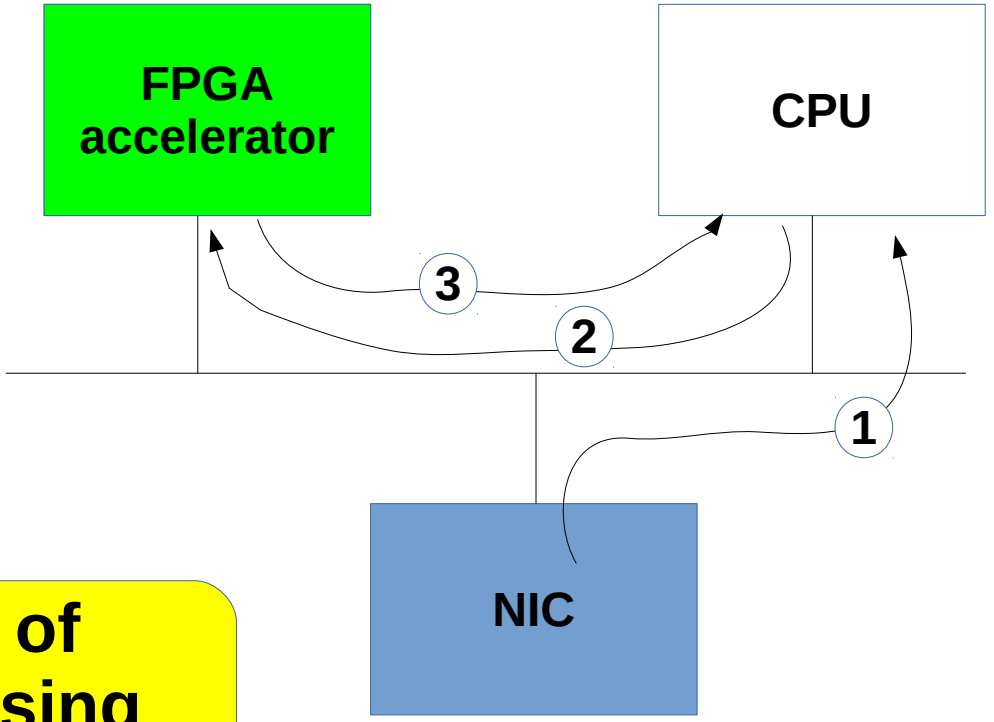
2-5 usec

5-7usec

Busy

1 Receive data
2 Invoke offload
3 Retrieve data

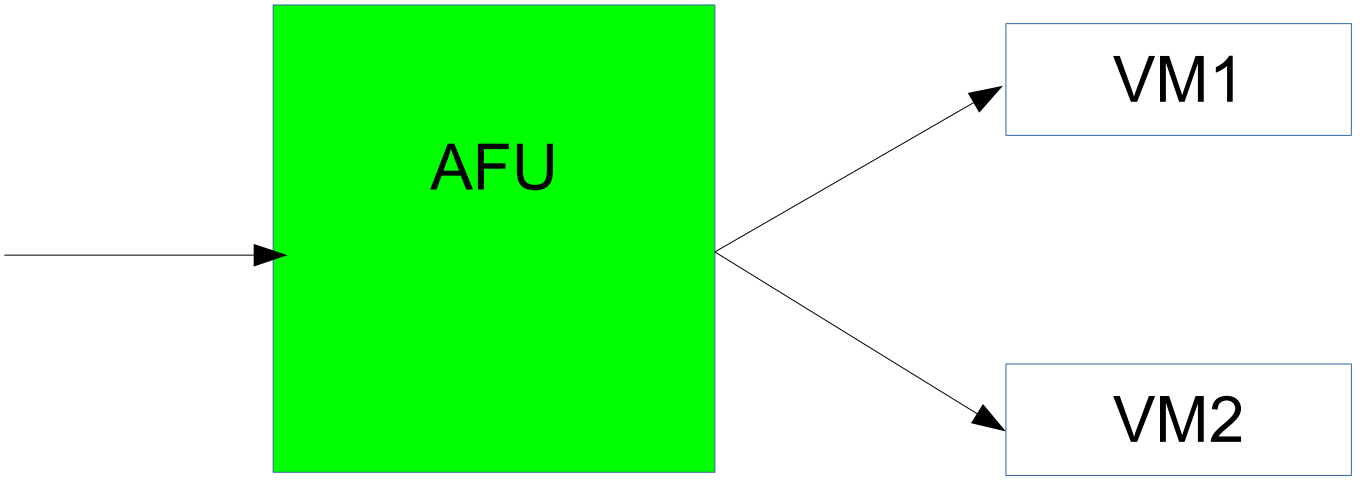
2-5 usec



Loosing the benefits of SmartNIC inline processing capabilities!

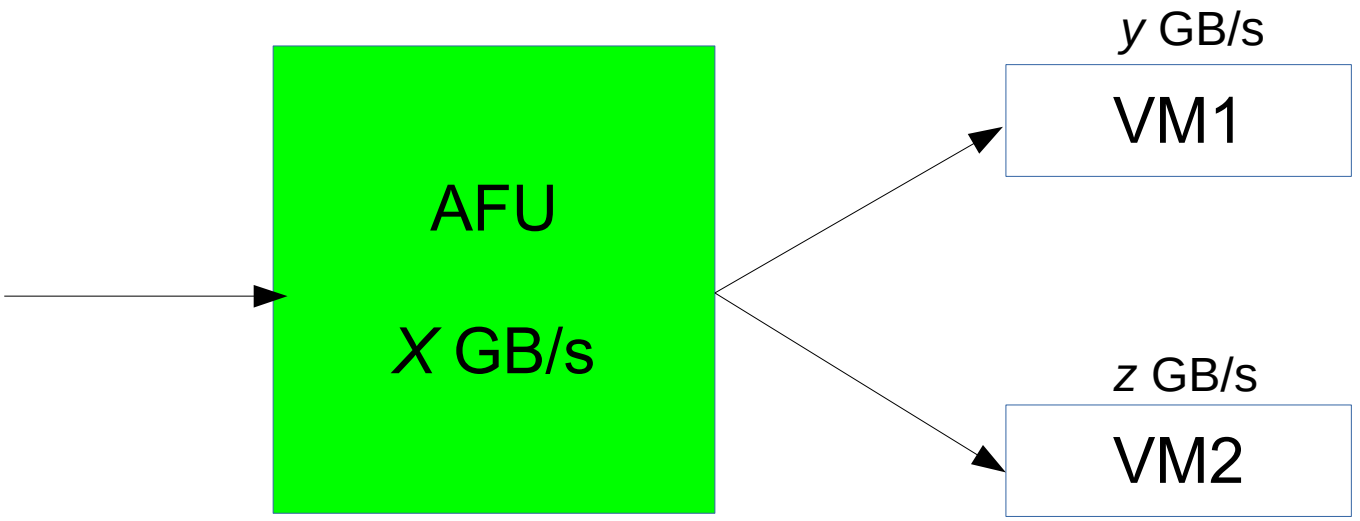
SmartNIC sharing: missing pieces

- Goal: maximize utilization



SmartNIC sharing: missing pieces

- Goal: maximize utilization
- SmartNICs might allow “free” sharing



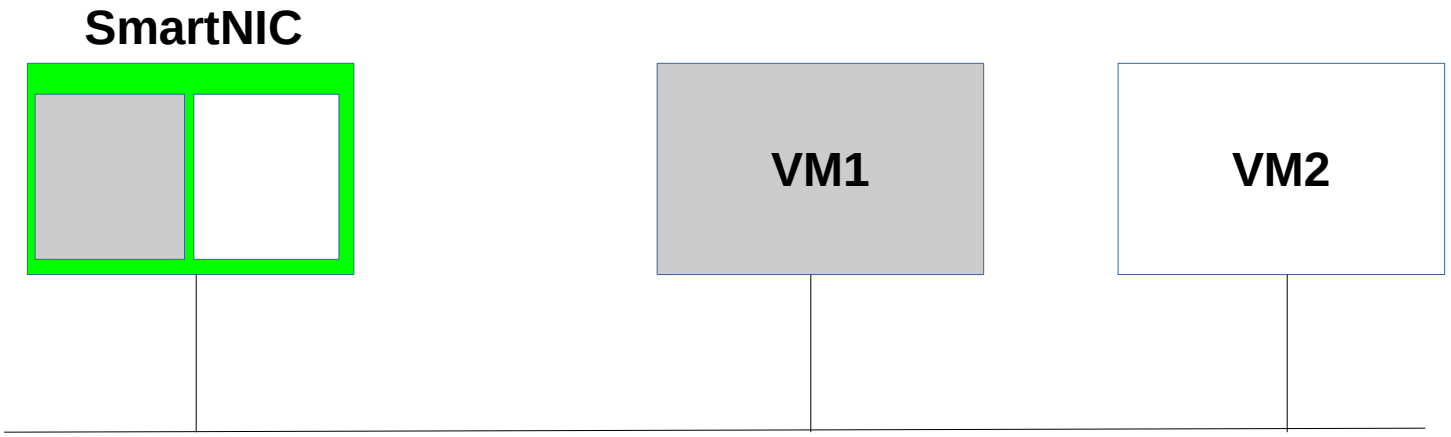
if *X*>*y*+*z* sharing is free

SmartNIC sharing: missing pieces

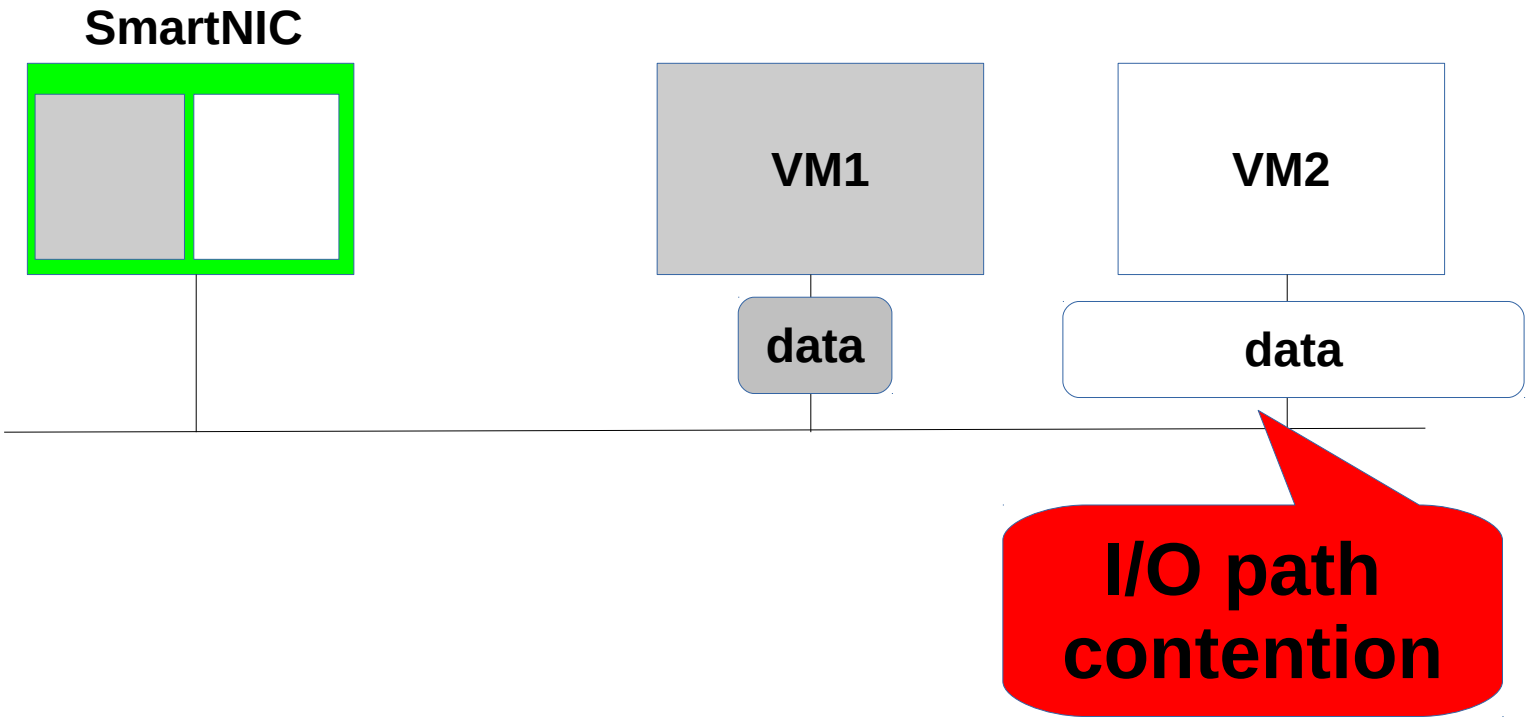
- Goal: maximize utilization
- SmartNICs might allow “free” sharing

- Needed: time sharing
- Needed: policy for I/O sharing

SmartNIC virtualization: not just compute fairness



SmartNIC virtualization: not just compute fairness



NICA: infrastructure for network application acceleration in clouds

- **OS abstractions for inline processing**
- **SmartNIC virtualization architecture**
- **Implementation on Mellanox SmartNICs**
- **End-to-end evaluation**

Abstractions for inline processing Design considerations

Coprocessor model

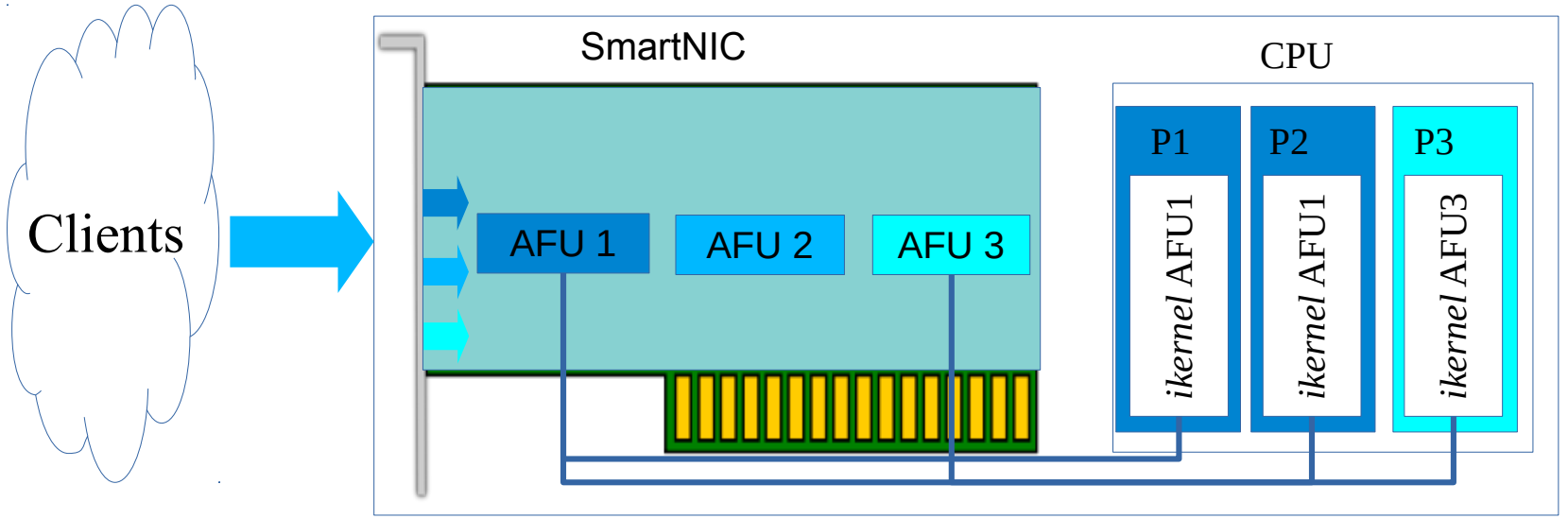
- Task invocation
- Independent runtime
- Explicit Data Transfers
- Access to application state
- Application state isolation

Abstractions for inline processing Design considerations

Inline model	Coprocessor model
• Activation/deactivation	• Task invocation
• Integrated with N/W stack	• Independent runtime
• *High-speed data plane*	• Explicit Data Transfers
• Access to application state	• Access to application state
• Application state isolation	• Application state isolation

New OS abstraction: *ikernel*

- Encapsulates application code and state on AFU
- *Private* to a process – extends isolation guarantees into the NIC



New OS abstraction: *ikernel*

- To activate: a*ttach* to an open socket on the host
- All the traffic routed to/from socket passes through the ikernel

New OS abstraction: *ikernel*

- To activate: a*ttach* to an open socket on the host
- All the traffic routed to/from socket passes through the ikernel
- Behavior is protocol-dependent
  - UDP: all messages to/from the port
  - TCP: according to five-tuple. Listening sockets handled on SmartNIC
- *ikernel* lifetime determined by the process lifetime

New OS abstraction: *ikernel*

- To activate: a*ttach* to an open socket on the host
- All the traffic routed to/from socket passes through the ikernel
- Behavior is protocol-dependent
  - UDP: all messages to/from the port
  - TCP: according to five-tuple. Listening sockets handled on SmartNIC

Take away: *ikernel* support is integrated with the network stack and process abstraction

Control plane to AFU

- No direct memory sharing with the host
  - AFU state might be stored in registers
- Need consistent access to transient application state

Control plane to AFU

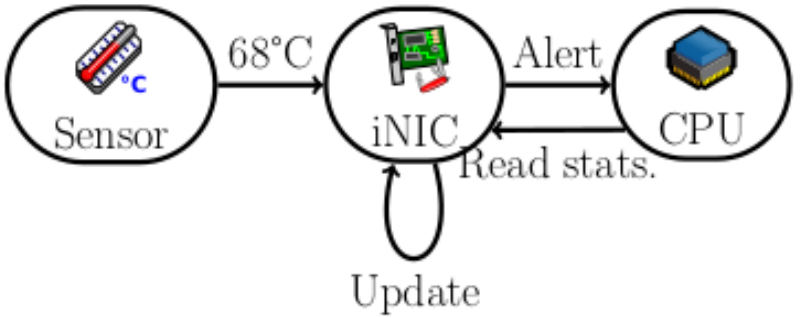
- No direct memory sharing with the host
  - AFU state might be stored in registers
- Need consistent access to transient application state

Remote Procedure Call interface to AFU

- Similar to ioctl
- Invokes operations on AFU

Example: IoT server

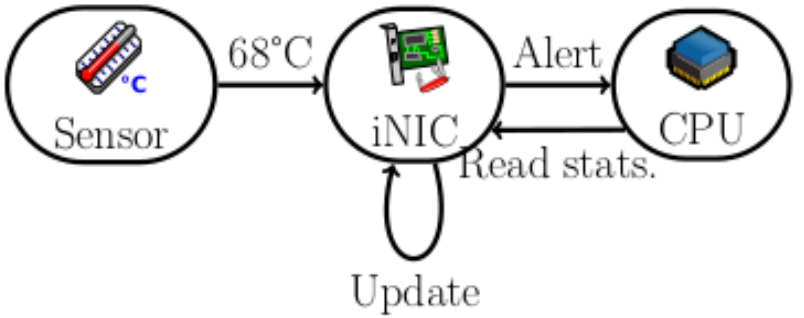
- Receive measurements
- Update statistics
- Alert on critical events
- Retrieve statistics



Example: IoT server

On AFU

- Receive measurements
- Update statistics



- Alert on critical events
- Retrieve statistics

On the host

IoT server: CPU code

```
// Create handle
k = ik_create(THERMAL_APP);
// Set parameters
ik_command(k, SET_THRESH, 55, NULL);
// Init a socket.
s = socket(...); bind(s, ...);
// Activate the ikernel
ik_attach(k, s);
// Use standard receive call
while (recvmsg(s, buf, ...)){
  // Read the current average from the ikernel
  ik_command(k, AVERAGE_TEMP, 0, &t_avg);
  // Do CPU processing
  trigger_alert(buf, t_avg);
}
// Stop the ikernel
ik_detach(k, s); ik_destroy(k);
```

IoT server: CPU code

```
// Create handle
k = ik_create(THERMAL_APP);
// Set parameters
ik_command(k, SET_THRESH, 55, NULL)
// Init a socket.
s = socket(...); bind(s, ...);
// Activate the ikernel
ik_attach(k, s);
// Use standard receive call
while (recvmsg(s, buf, ...)){
  // Read the current average from the ikernel
  ik_command(k, AVERAGE_TEMP, 0, &t_avg);
  // Do CPU processing
  trigger_alert(buf, t_avg);
}
// Stop the ikernel
ik_detach(k, s); ik_destroy(k);
```

Dynamically invoke

Regular POSIX call

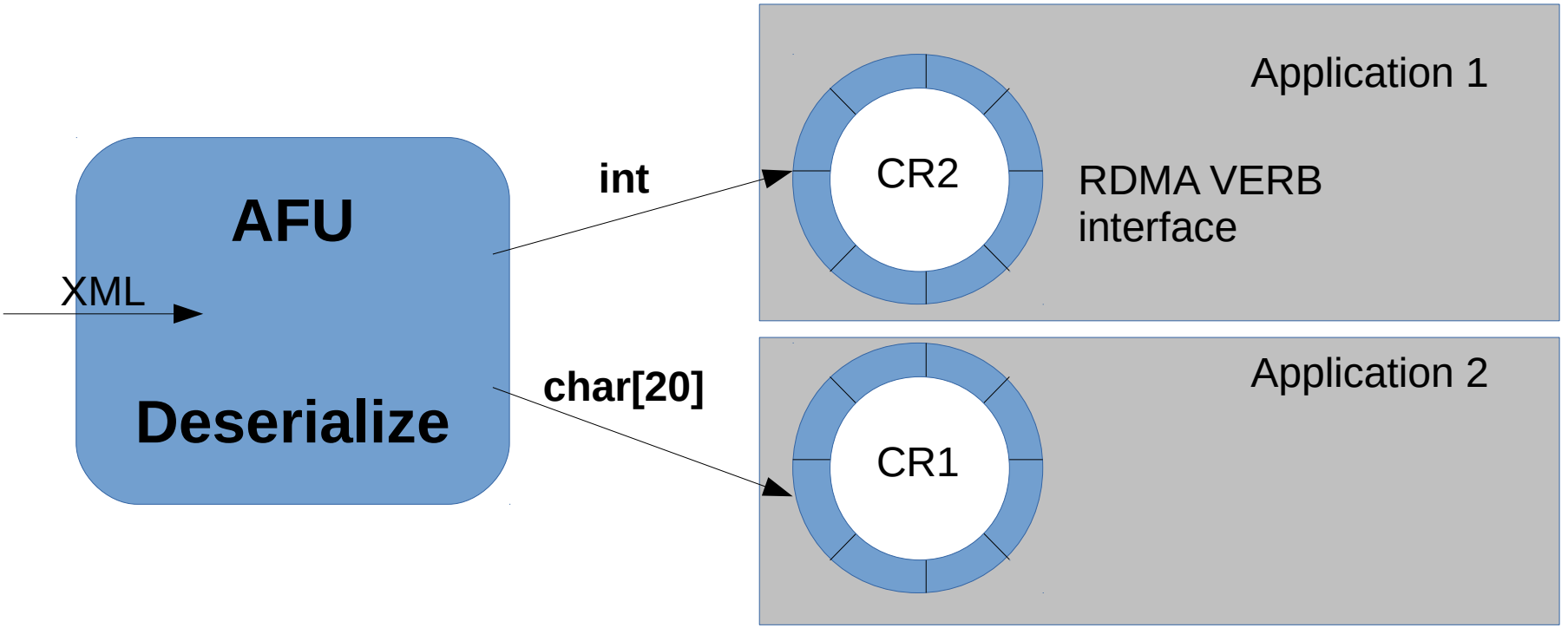
Access ikernel state (control path)

Optimized data path

- Custom ring: zero-copy low-latency application-level messaging with AFU
- Bypasses host network stack

Optimized data path

- Custom ring: zero-copy low-latency application-level messaging with AFU
- Bypasses host network stack



Virtualization

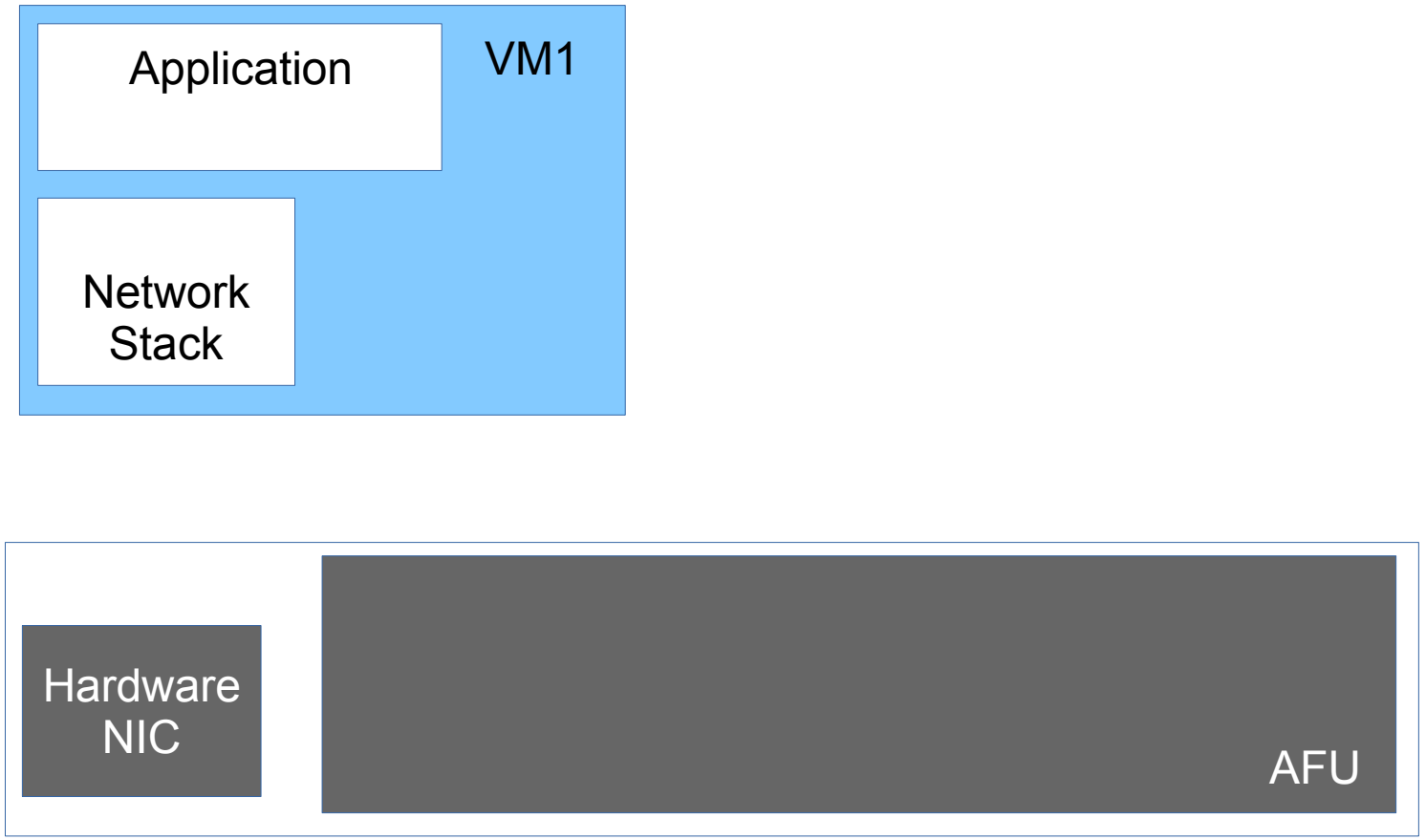
Requirements

- QoS and Performance isolation
  - I/O to host, I/O to network, Compute
- Time sharing
  - Fine-grain hardware context-switching
- Application state protection
- High-speed data path

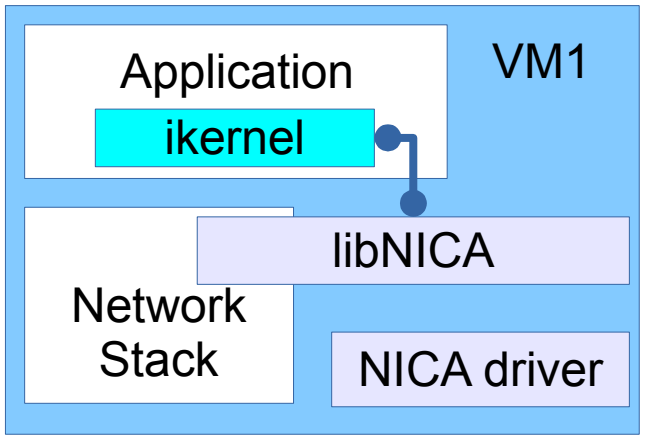
Summary so far

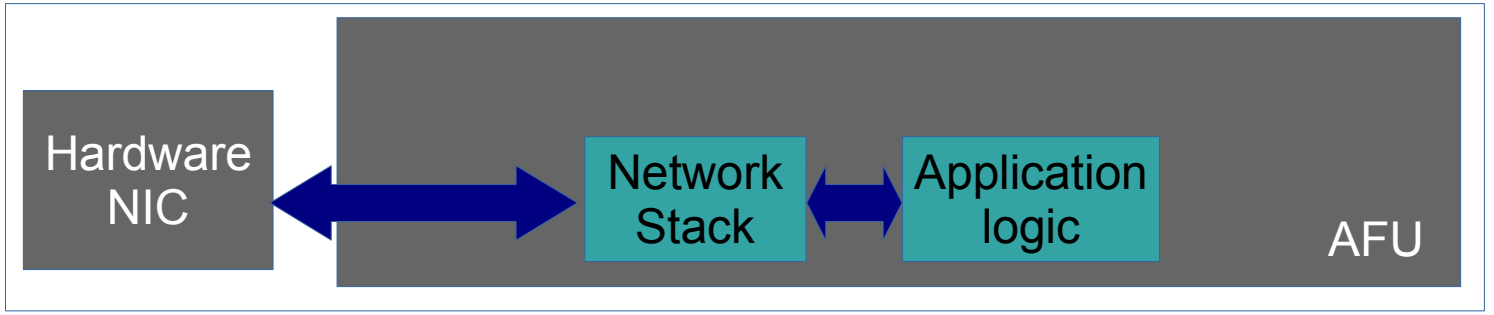
- ikernel: new OS abstraction for inline processing, data plane vs. control plane
- Virtualization combines compute and I/O isolation, and fine-grain sharing

NICA: implementation

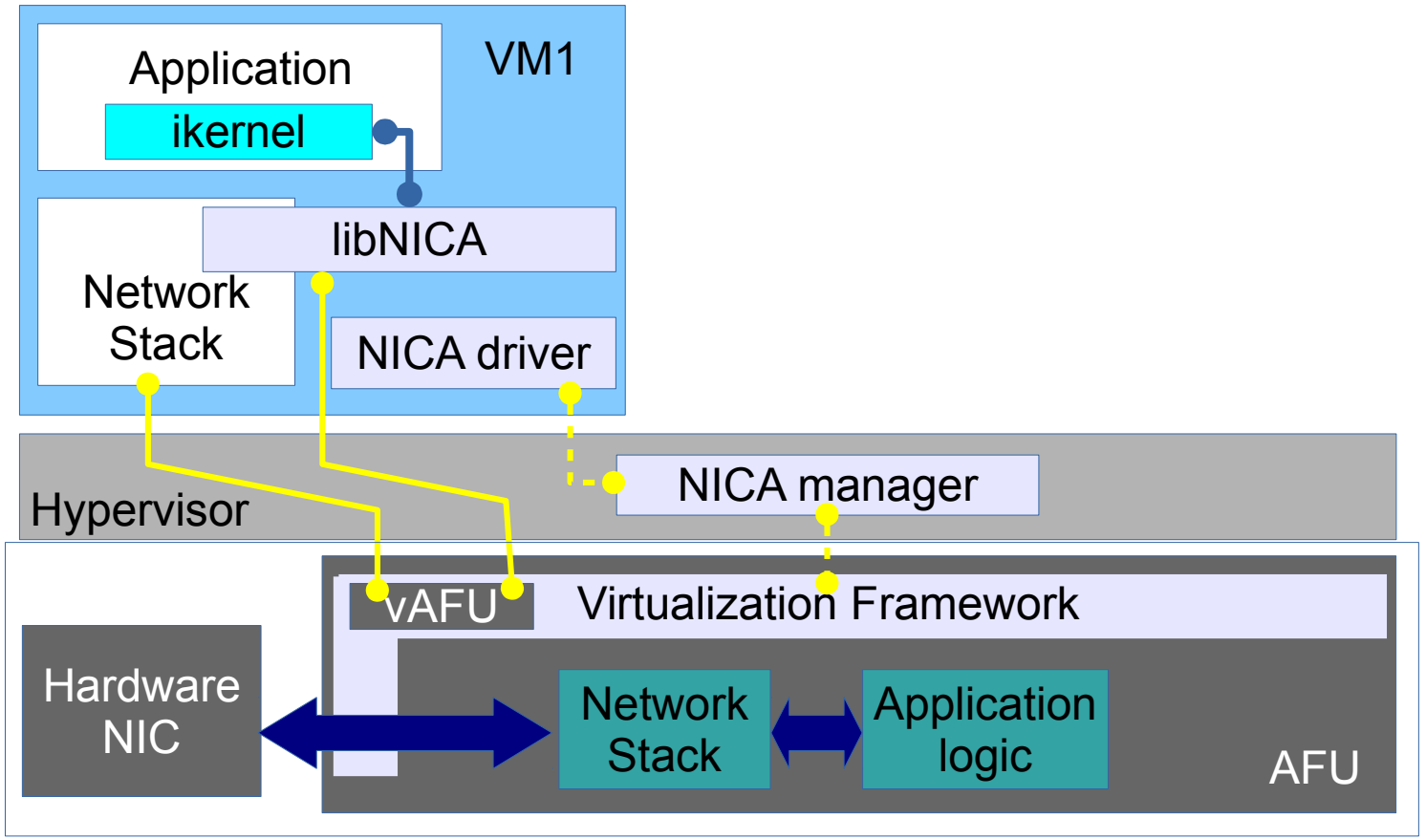


NICA: software/hardware co-design

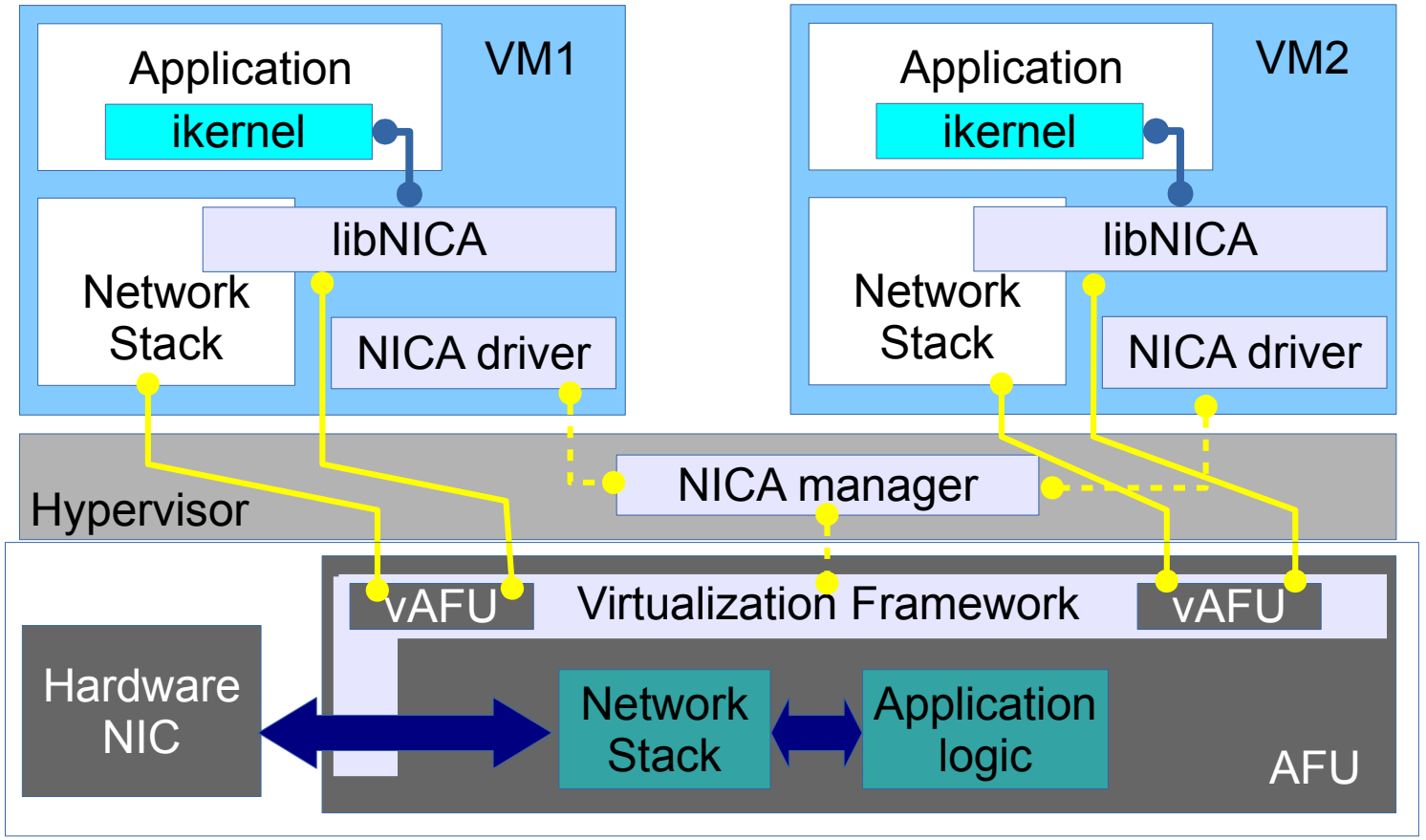




NICA: software/hardware co-design



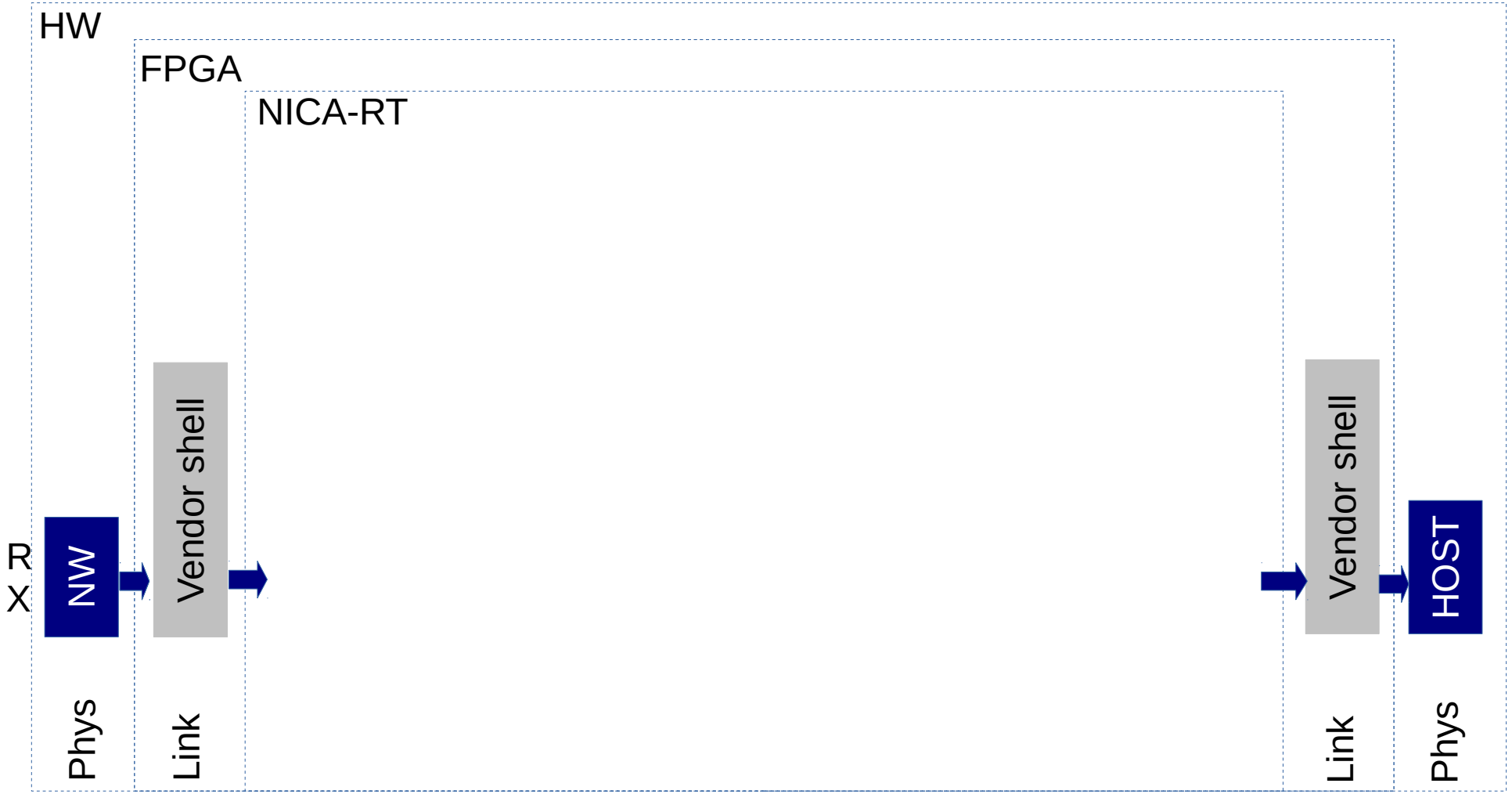
NICA: software/hardware co-design



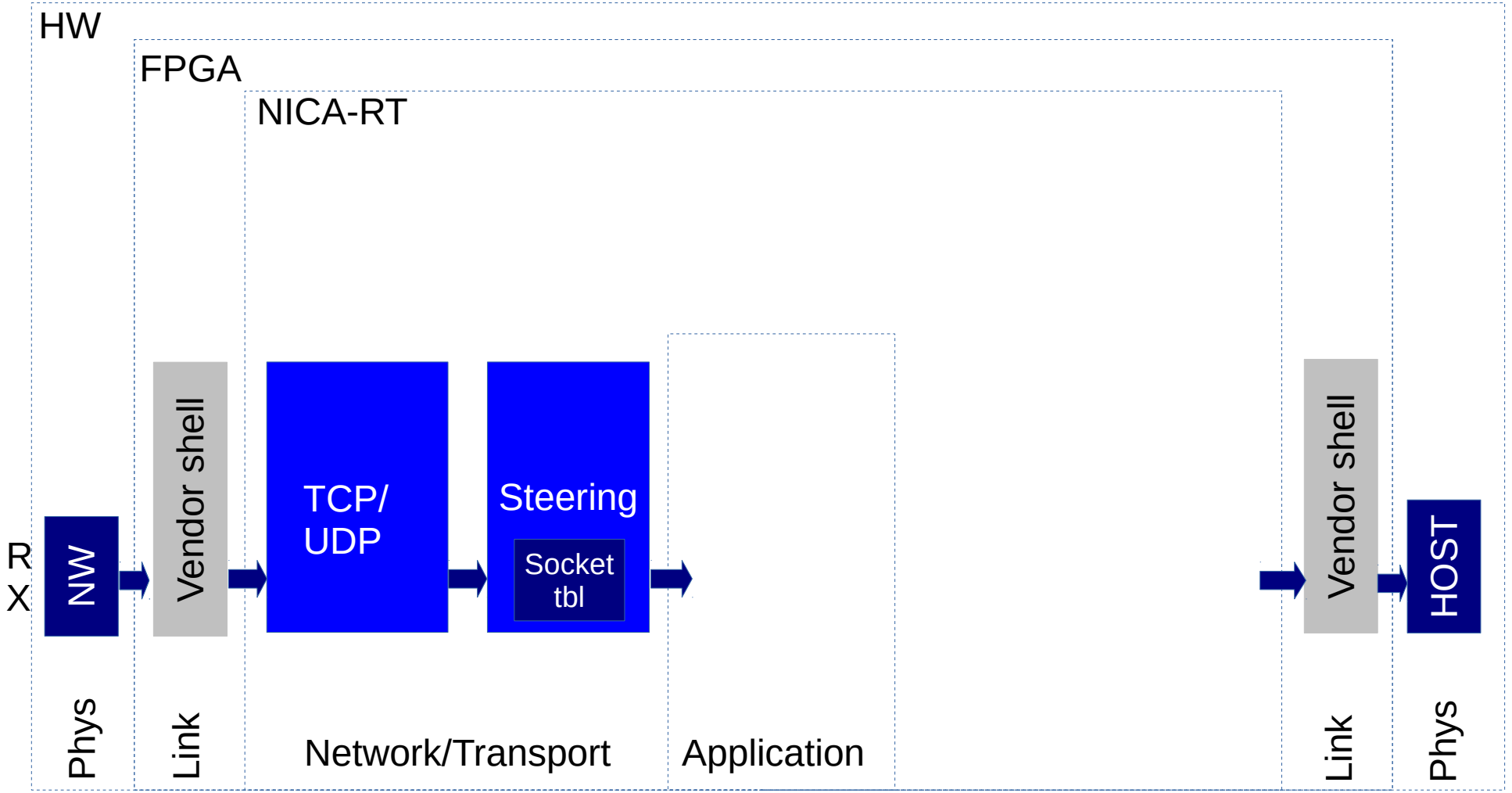
NICA Implementation

- Mellanox Innova Flex 4 Lx EN NIC
  - With Xilinx Kintex Ultrascale XCKU060
- Vivado HLS with Verilog patches
- CPU: libVMA user-space networking library

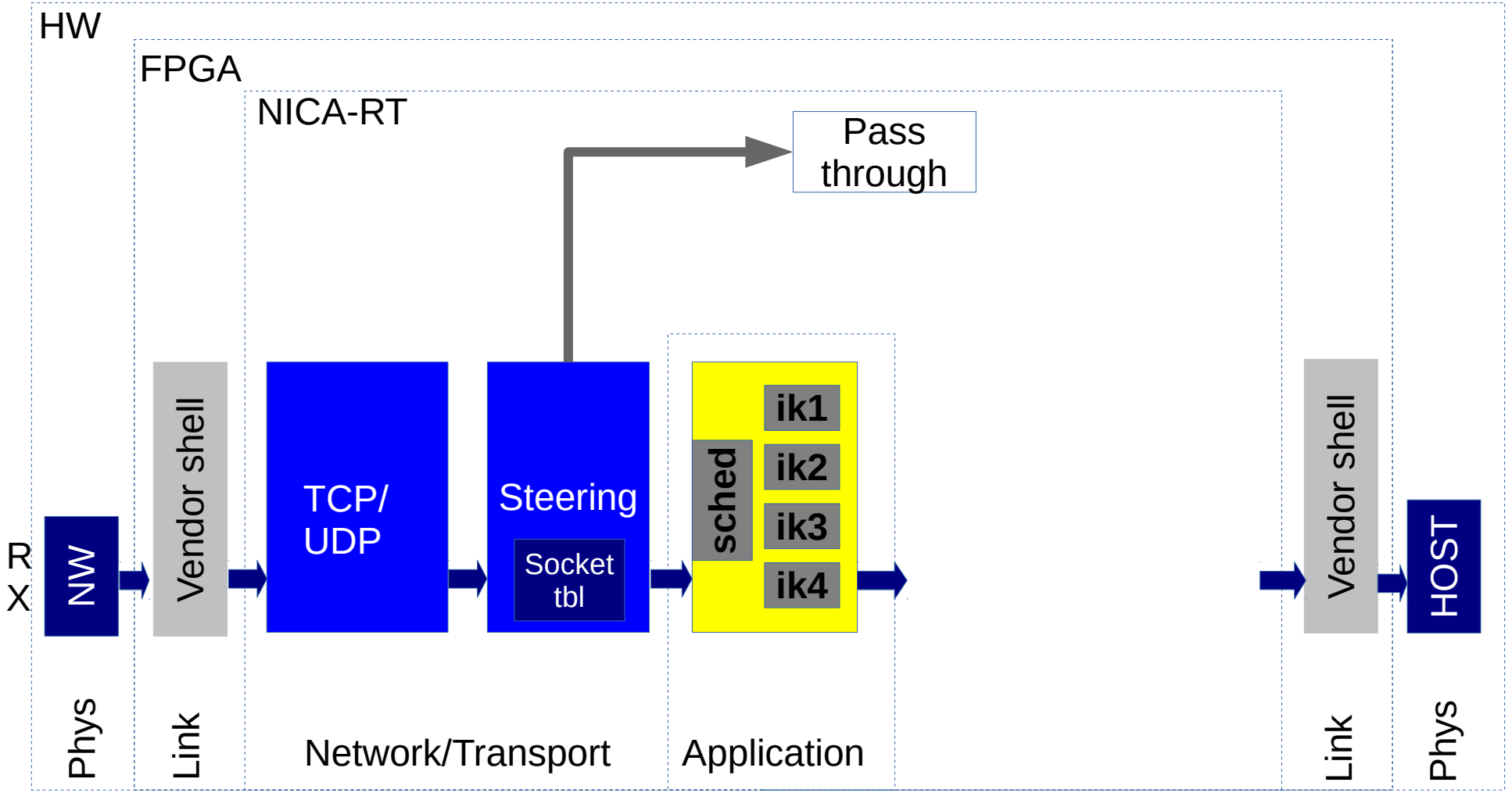
NICA FPGA runtime



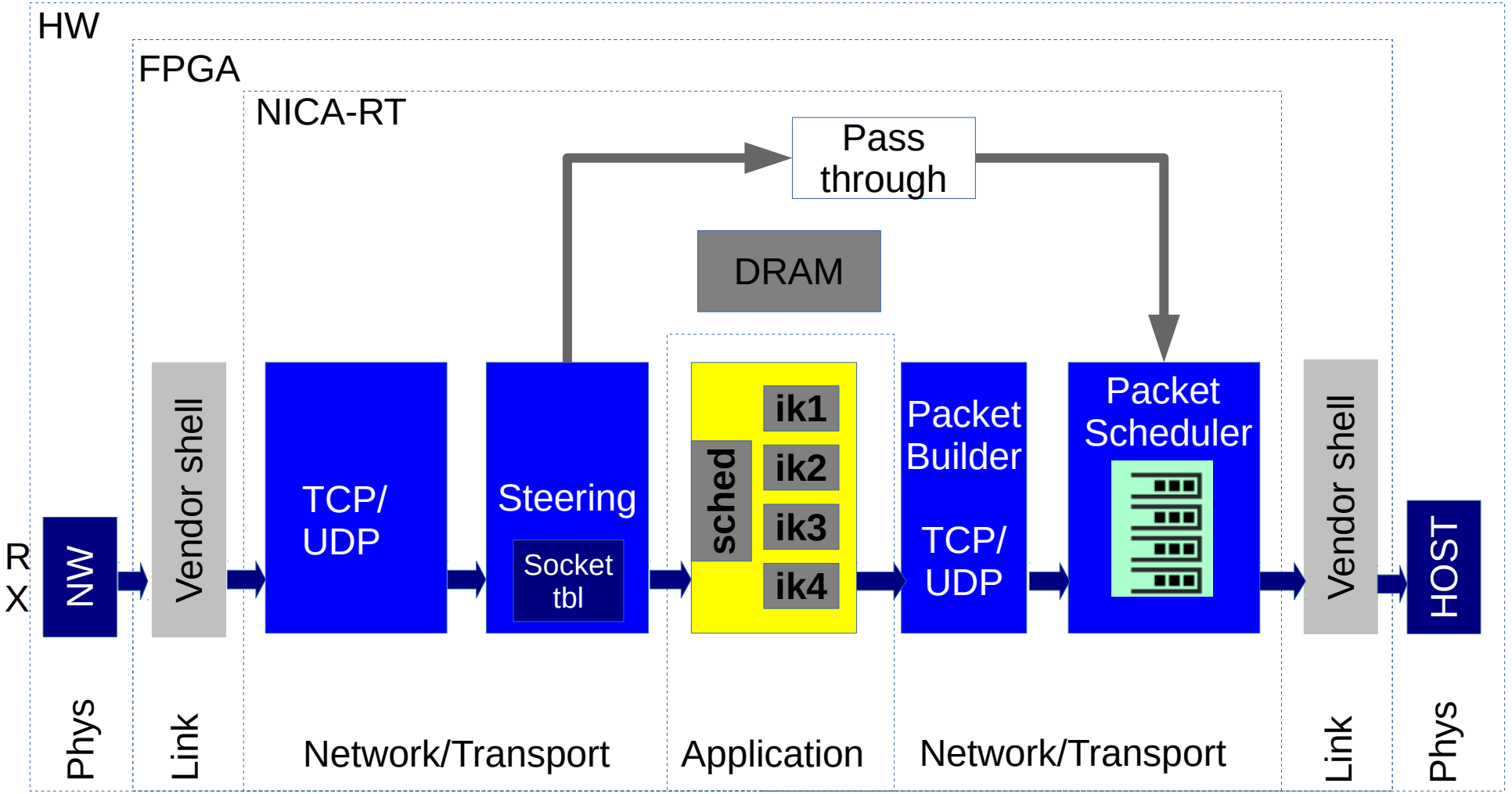
NICA FPGA runtime



NICA FPGA runtime



NICA FPGA runtime

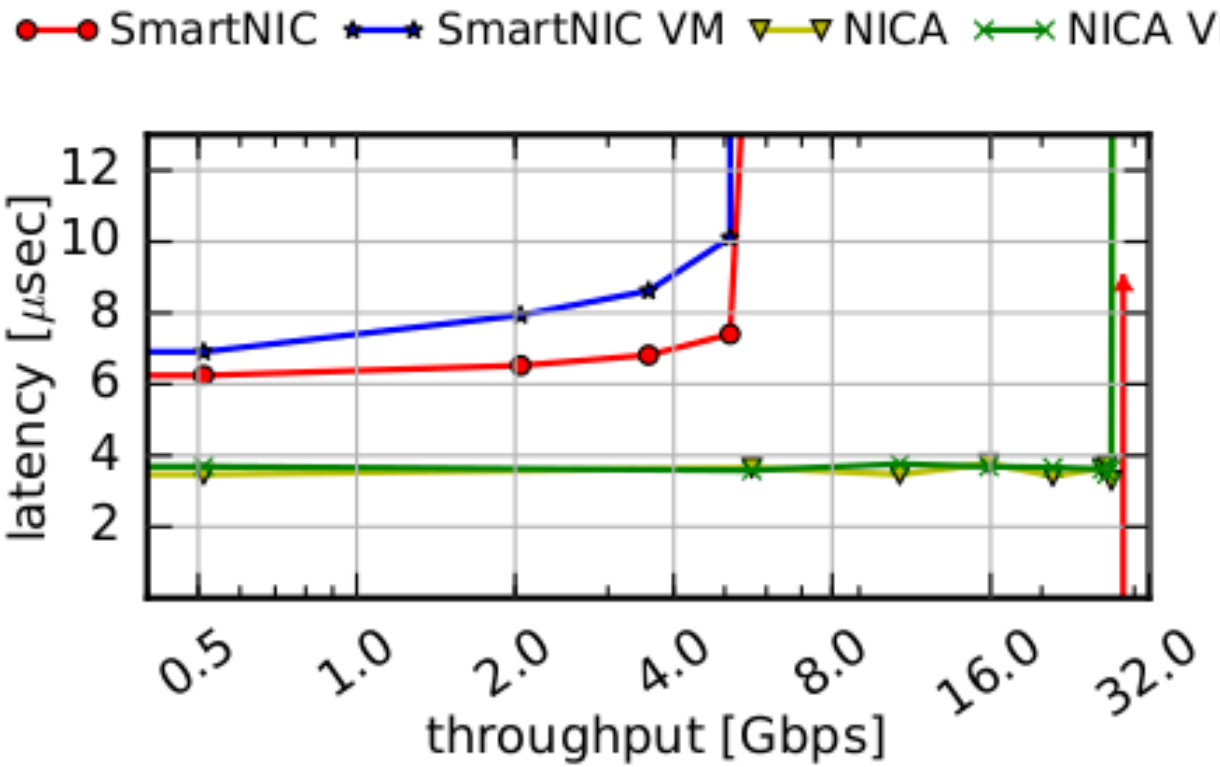


Evaluation

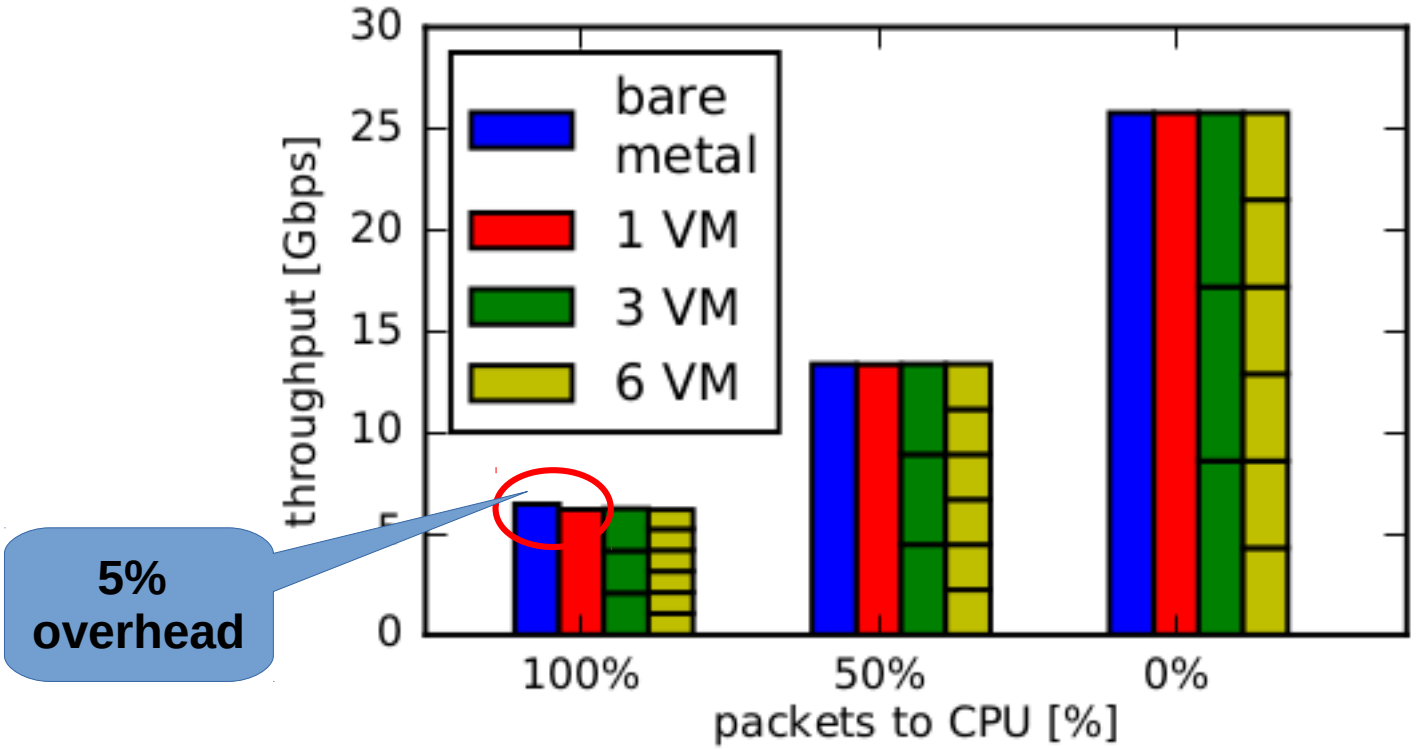
- Microbenchmarks:
  - Latency / throughput
  - Inline processing overheads
  - Virtualization overheads
  - QoS correctness
- Applications:
  - Memcached cache
  - IoT server

Virtualized AFU with NICA:

UDP echo latency/throughput



Virtualization overheads for filtering applications

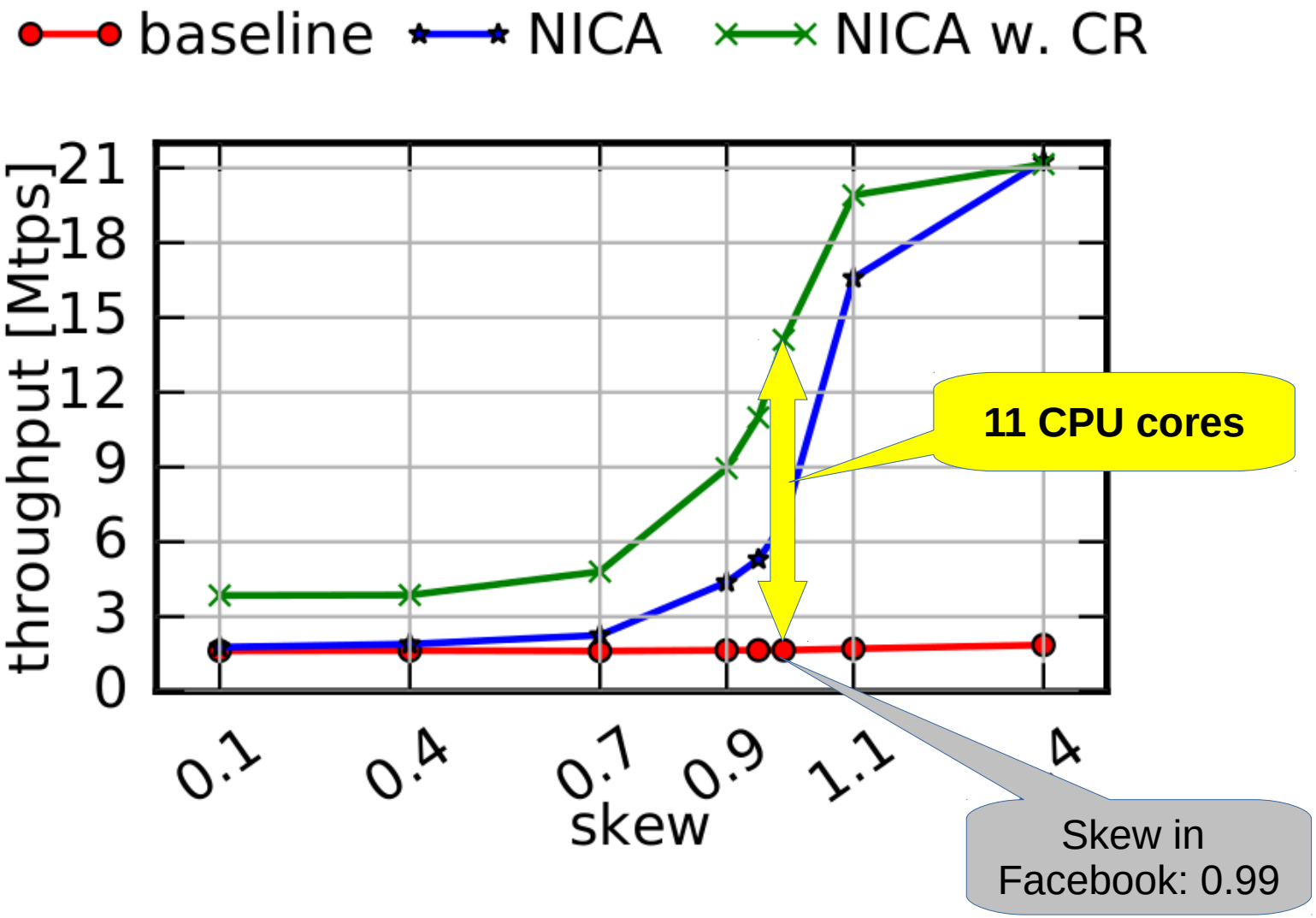


memcached inline cache

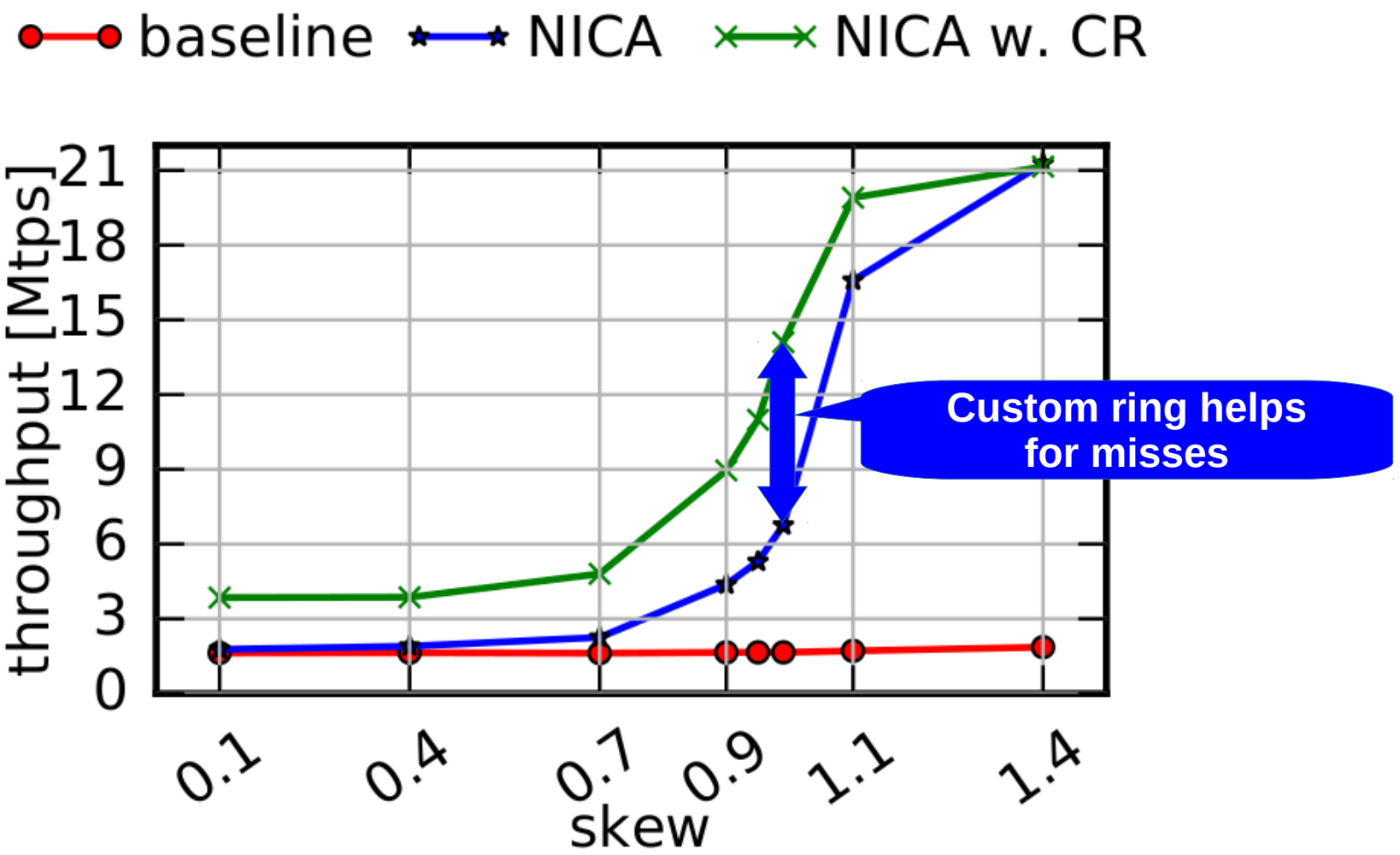
- A cache hosted in iNIC (128MB cache)
- Consistent, look-through, 1-way associative
- Populated upon GET
  - ikernel intercepts the host response
- Invalidated upon SET
  - ikernel frees the entry and passes the message to host

* Works only with fixed key/values sizes(16 bytes each)
* Requires ASCII protocol

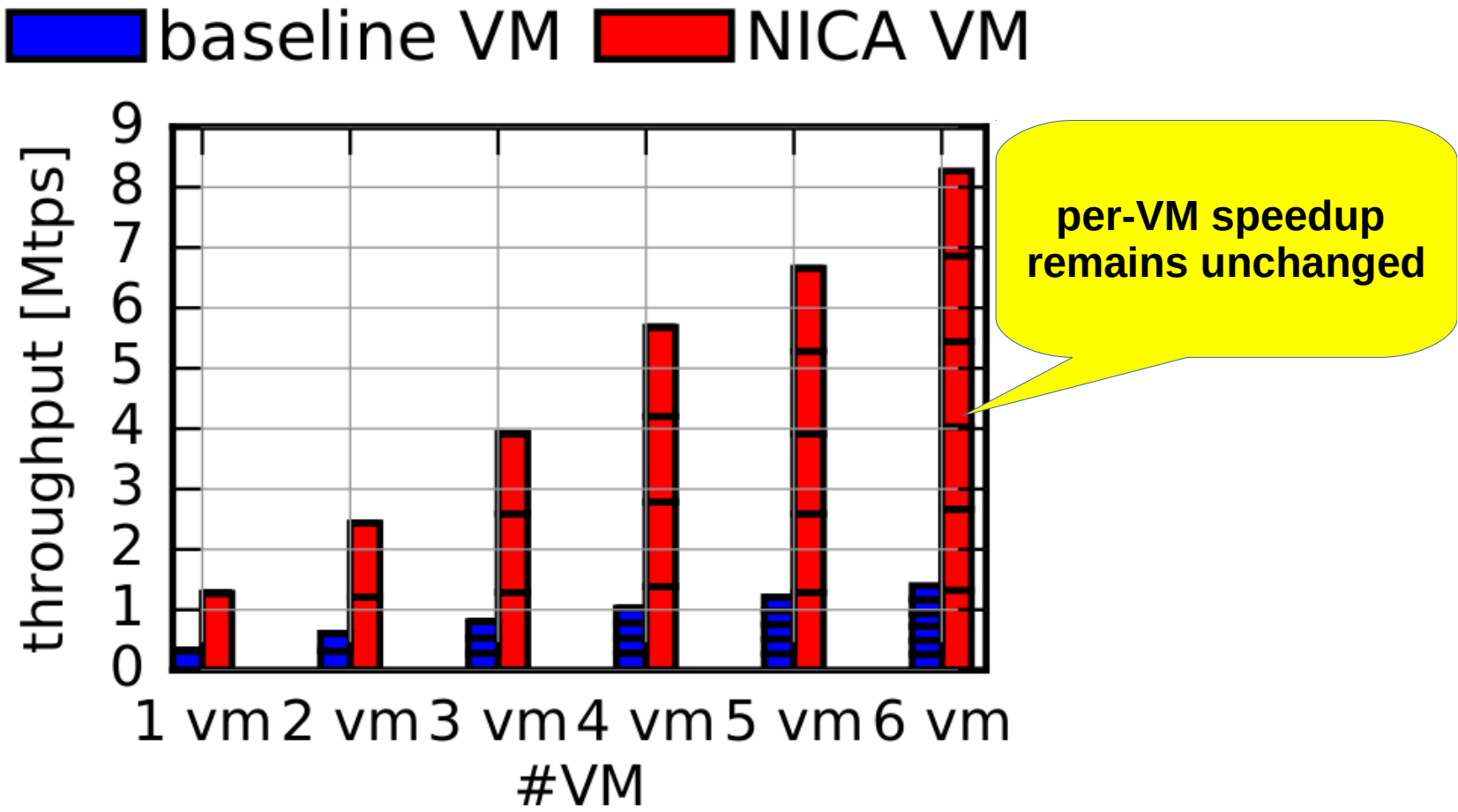
GET performance for different Zipf(skews)



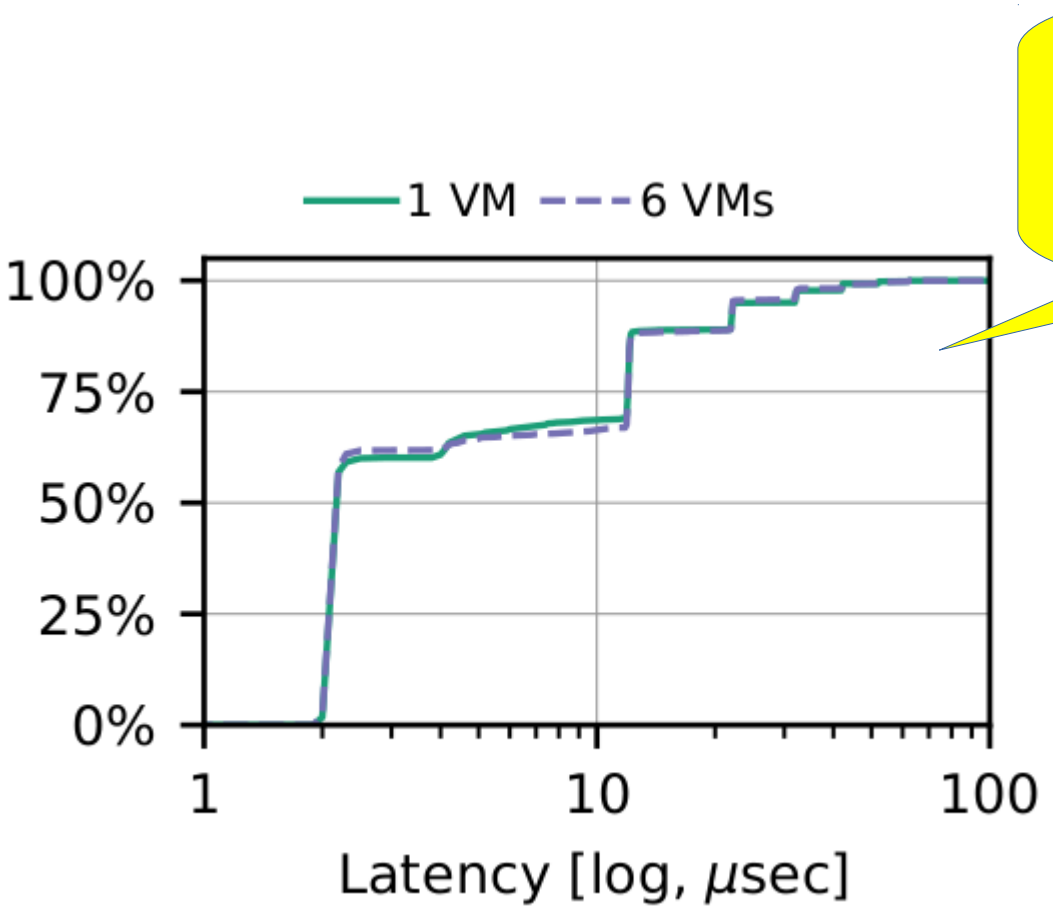
Custom ring is important



Virtualization overhead: throughput

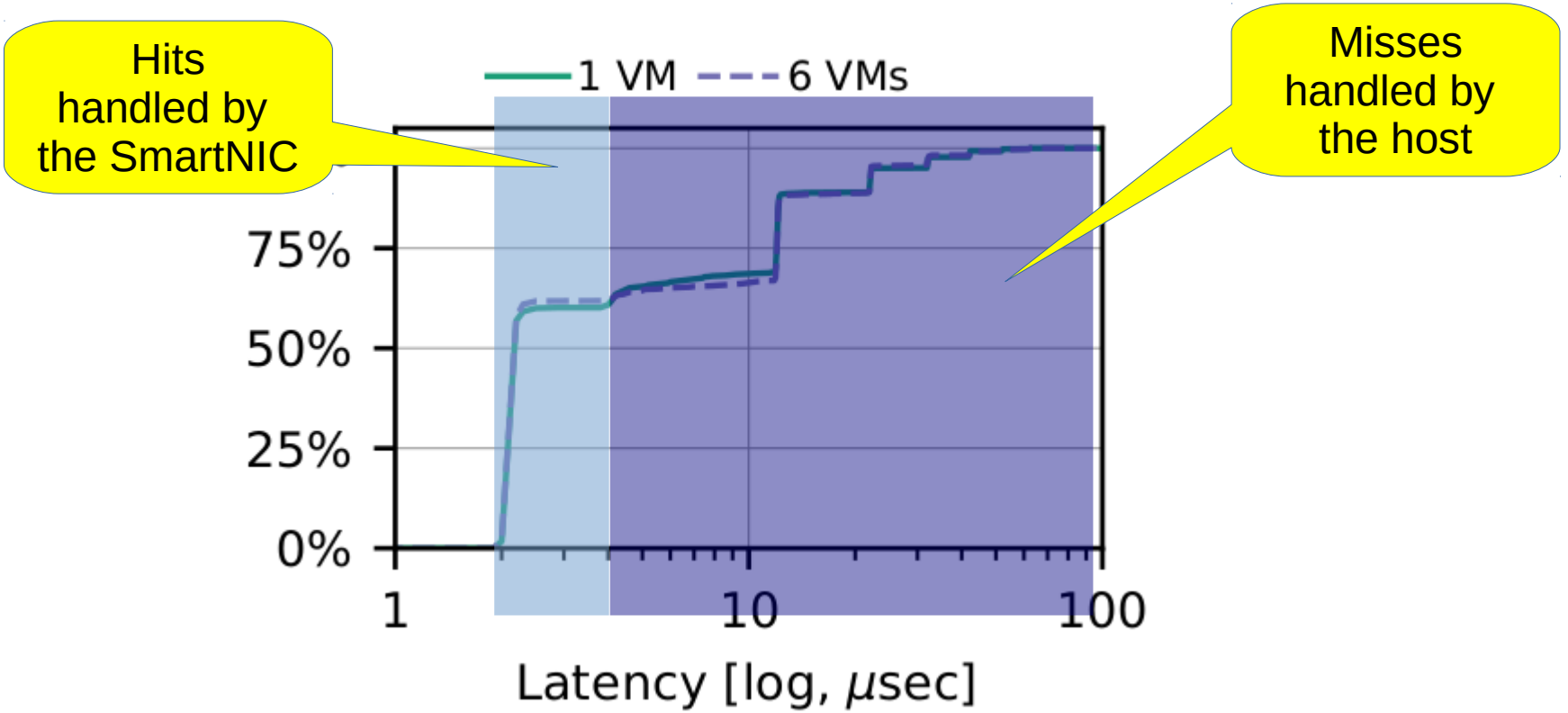


Virtualization overhead: latency (hit rate: 60%)



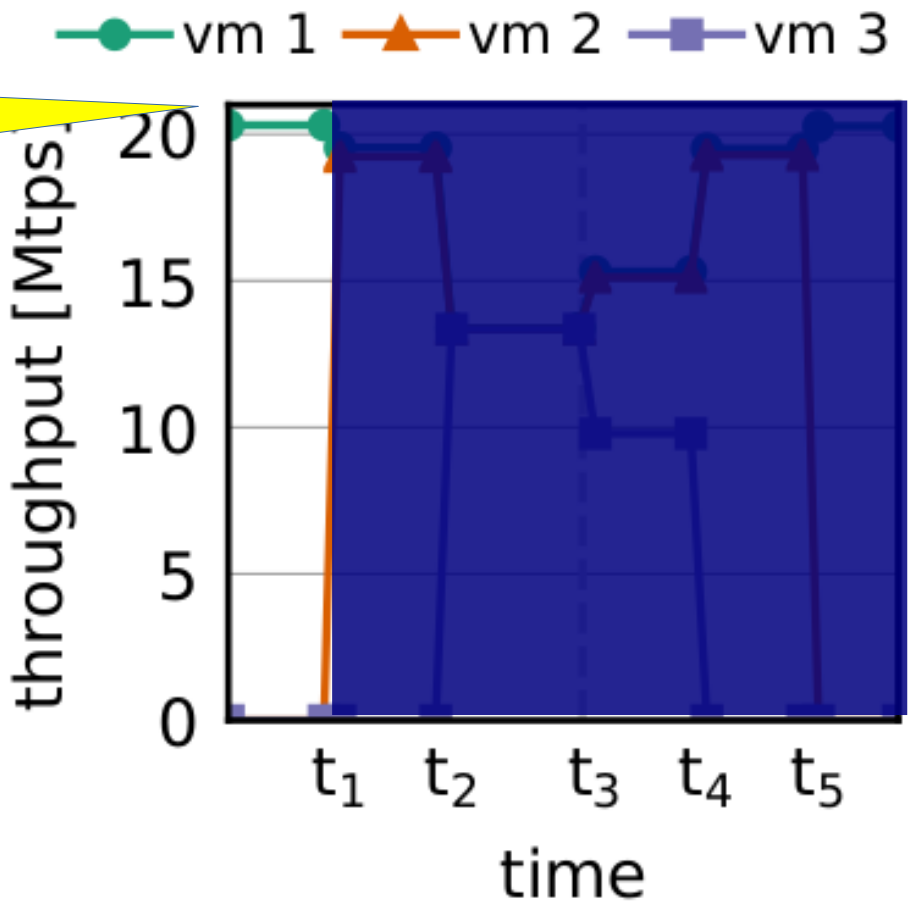
Virtualized memcached – no visible latency overehad

Virtualization overhead: latency (hit rate: 60%)



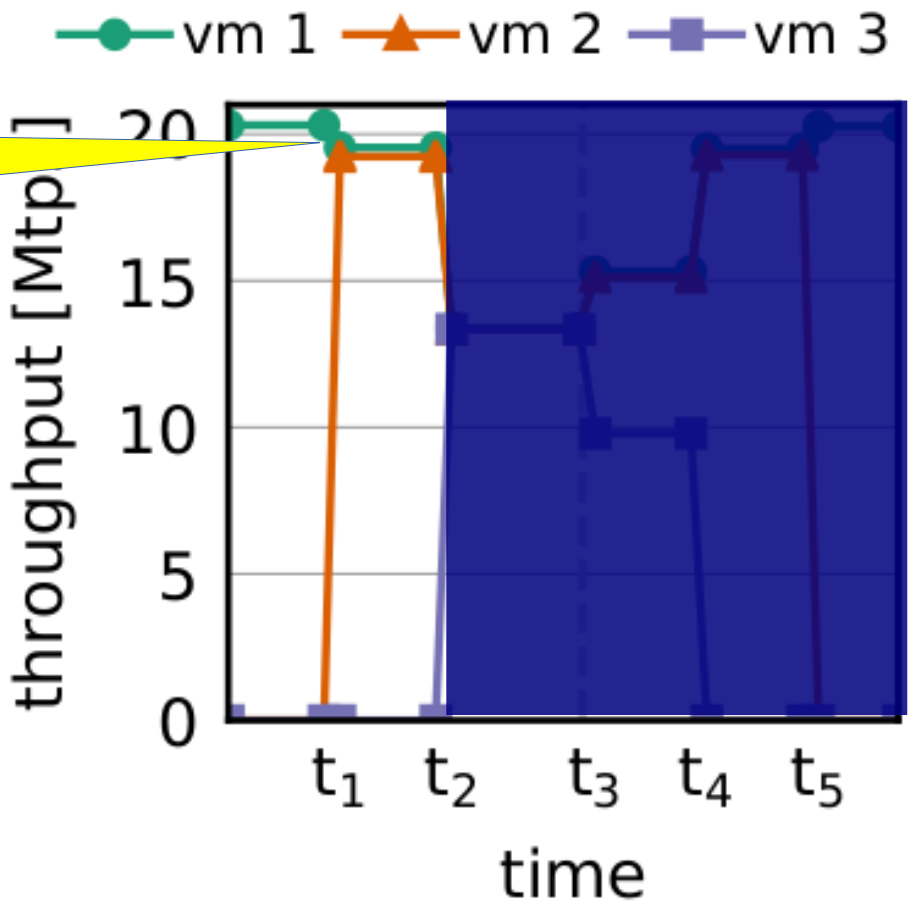
Bandwidth isolation: fair share

1 VM – 20Mtps

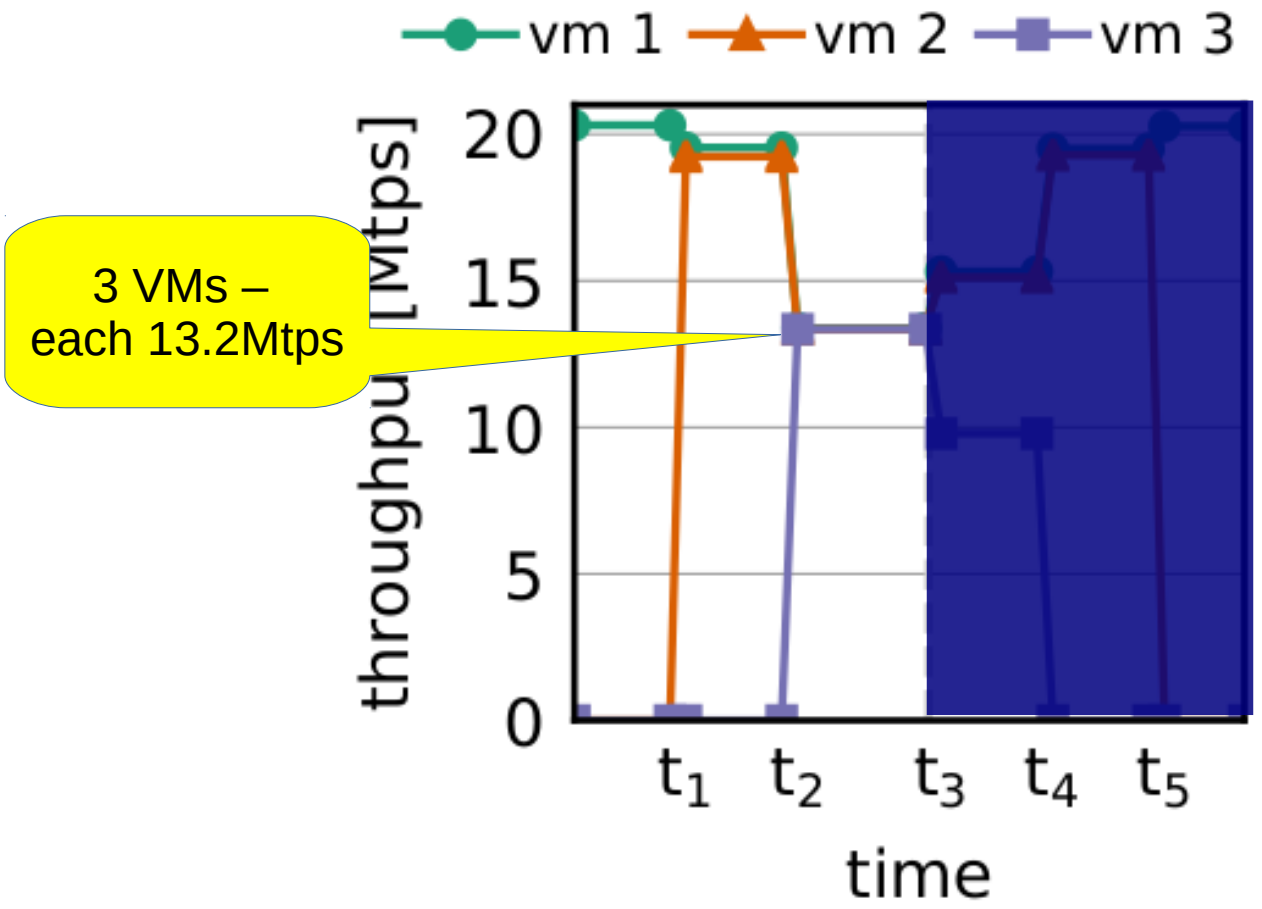


Bandwidth isolation: fair share

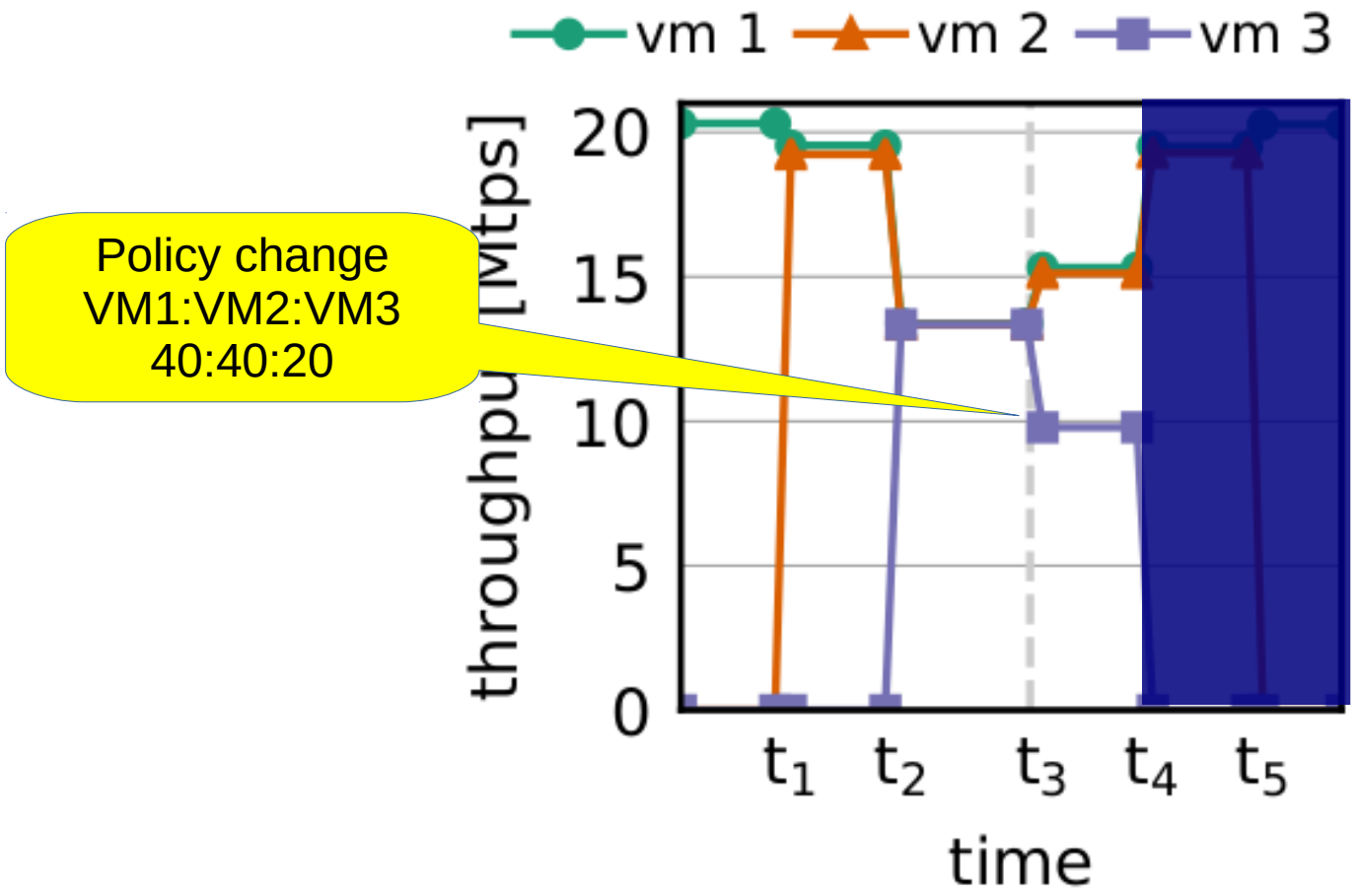
2 VMs –
each 19.6Mtps



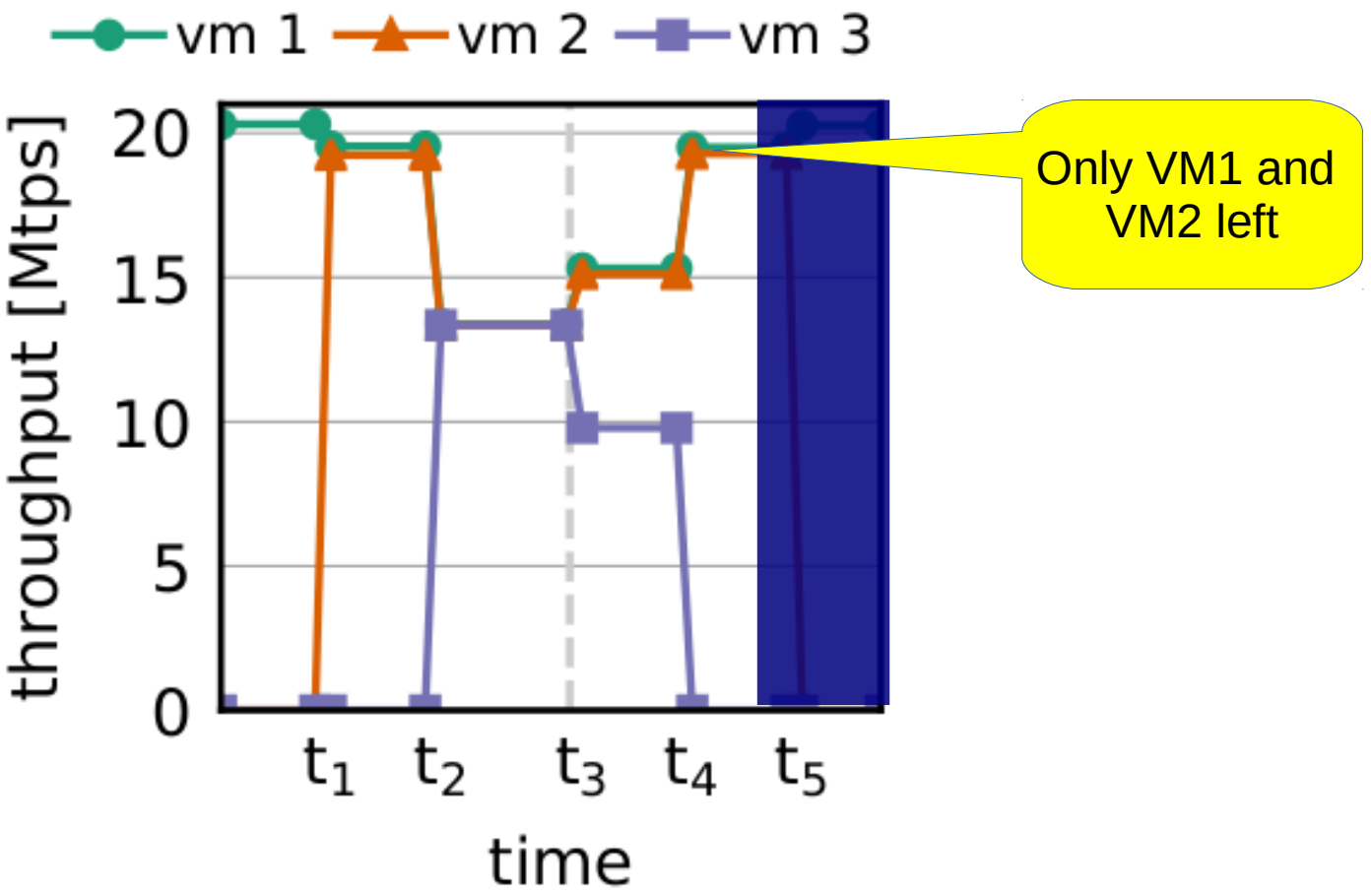
Bandwidth isolation: fair share



Bandwidth isolation: fair share



Bandwidth isolation: fair share



Summary

- SmartNICs are here
- NICA enables application offload and sharing
  - *ikernel* abstraction and OS support for inline acceleration
  - low-overhead virtualization framework
- End-to-end implementation on real SmartNIC

Future work

- Access to other peripherals
- Reliable communications: TCP stack, RDMA
- Higher-level languages
- NFV/DPDK, other applications
- Distributed systems and ikernel chaining

Thank you!

mark@ee.technion.ac.il