

Cornell University



Towards CPU-free Data Movement in the Cloud (hardware/software approach)

Presenter: *Nikita Lazarev*

Advisors/Committee Members: *Christina Delimitrou, Zhiru Zhang, Rachit Agarwal*

Computer Systems Laboratory, Cornell University, USA



Trends in Cloud Computing

Existing Technologies

Dagger: Three Design Principles

Dagger: Design and Implementation

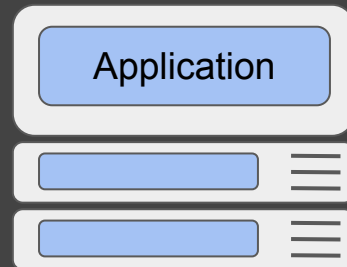
Evaluation

Future Work



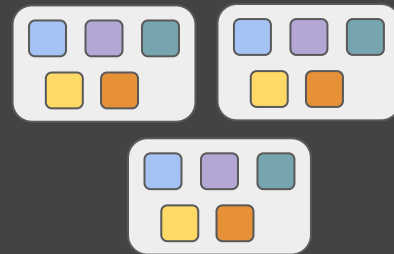
Traditional Cloud Applications (Monoliths)

- *Tightly-coupled application logic in a single statically/dynamically linked binary*



Shift Towards Microservices

- *Loosely-coupled application logic split into many independent small applications*



Shift Towards Serverless

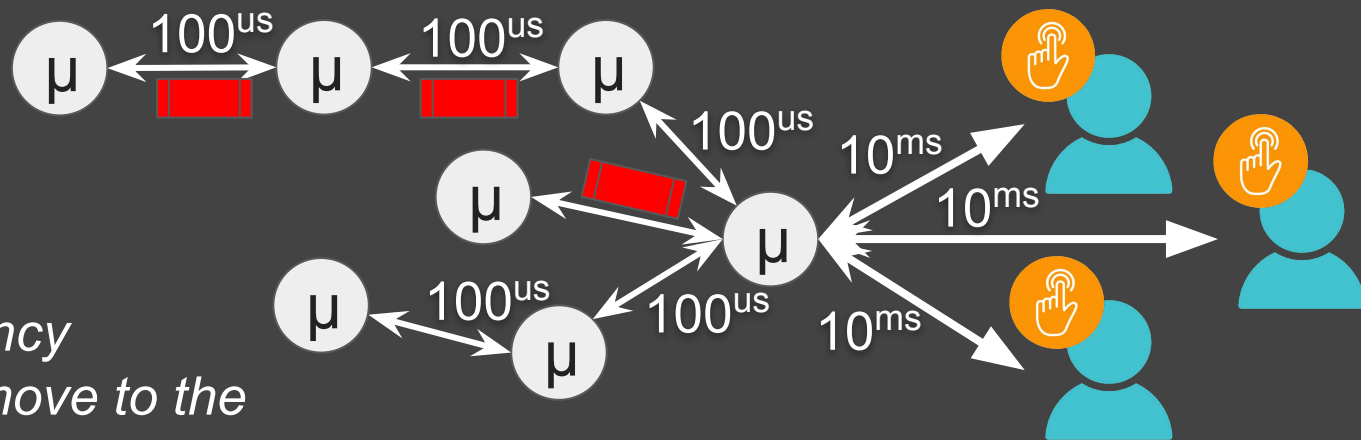
- *Fine application granularity*
- *Fine lifetime granularity*





Cloud Applications Today are Interactive

- *Frequent interaction with large sets of users*
- *Strict performance requirements as SLO*
 - *low tail latency under high load*
 - *performance predictability*



- *even low-latency applications move to the cloud*



Technology Trend: physical networking becomes fast

- *networking at the speed of light [1]*
- *optical ns-scale switching (H. Ballani, SIGCOMM'20 [2])*
- *TbE networking*

Question:

- *if physical networking is ns-scale, can end-host networking be there too?*
- *can we do more than ~Mrps per CPU core to saturate TbE links with fewer cores?*

[1] <https://www.sikich.com/insight/networking-at-the-speed-of-light-understanding-fiber-optics/>

[2] <https://www.microsoft.com/en-us/research/publication/sirius-a-flat-datacenter-network-with-nanosecond-optical-switching/>



Some Numbers: median latency

- *context switching time: 1.2 us*
- *local DRAM access latency: 100 ns*
- *remote (NUMA) DRAM access latency: 300 ns*
- *Linux Kernel networking stack median rtt: 30 us, tail: ~10 ms+*
- *RPC serialization: 10 us and more (Z. Pourhabibi, ASPLOS'20)*

We need to rethink both SW and HW layers to take advantage of the future technologies!

Can we make communication SW-free? CPU-free?



Trends in Cloud Computing

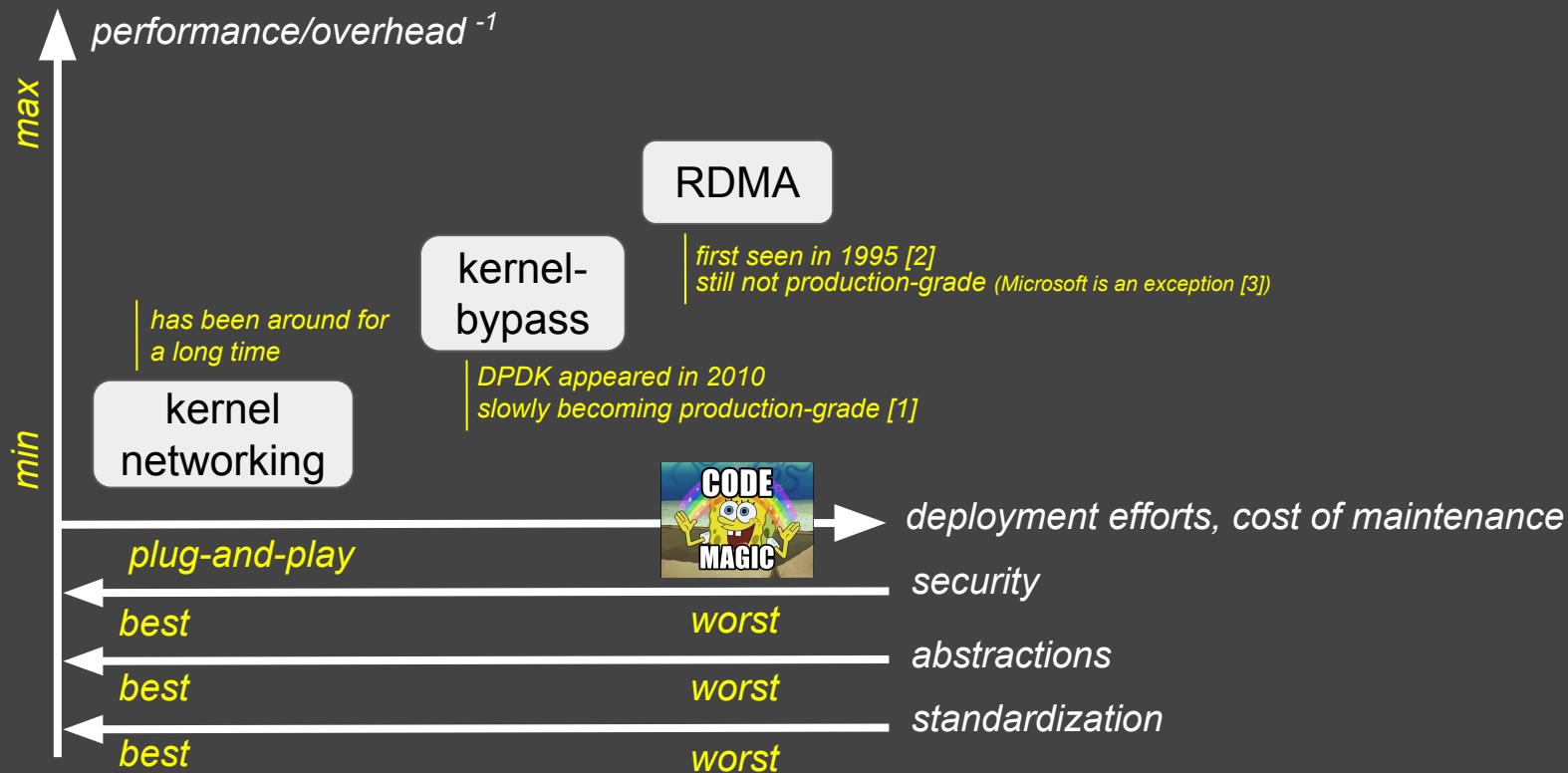
Existing Technologies

Dagger: Three Design Principles

Dagger: Design and Implementation

Evaluation

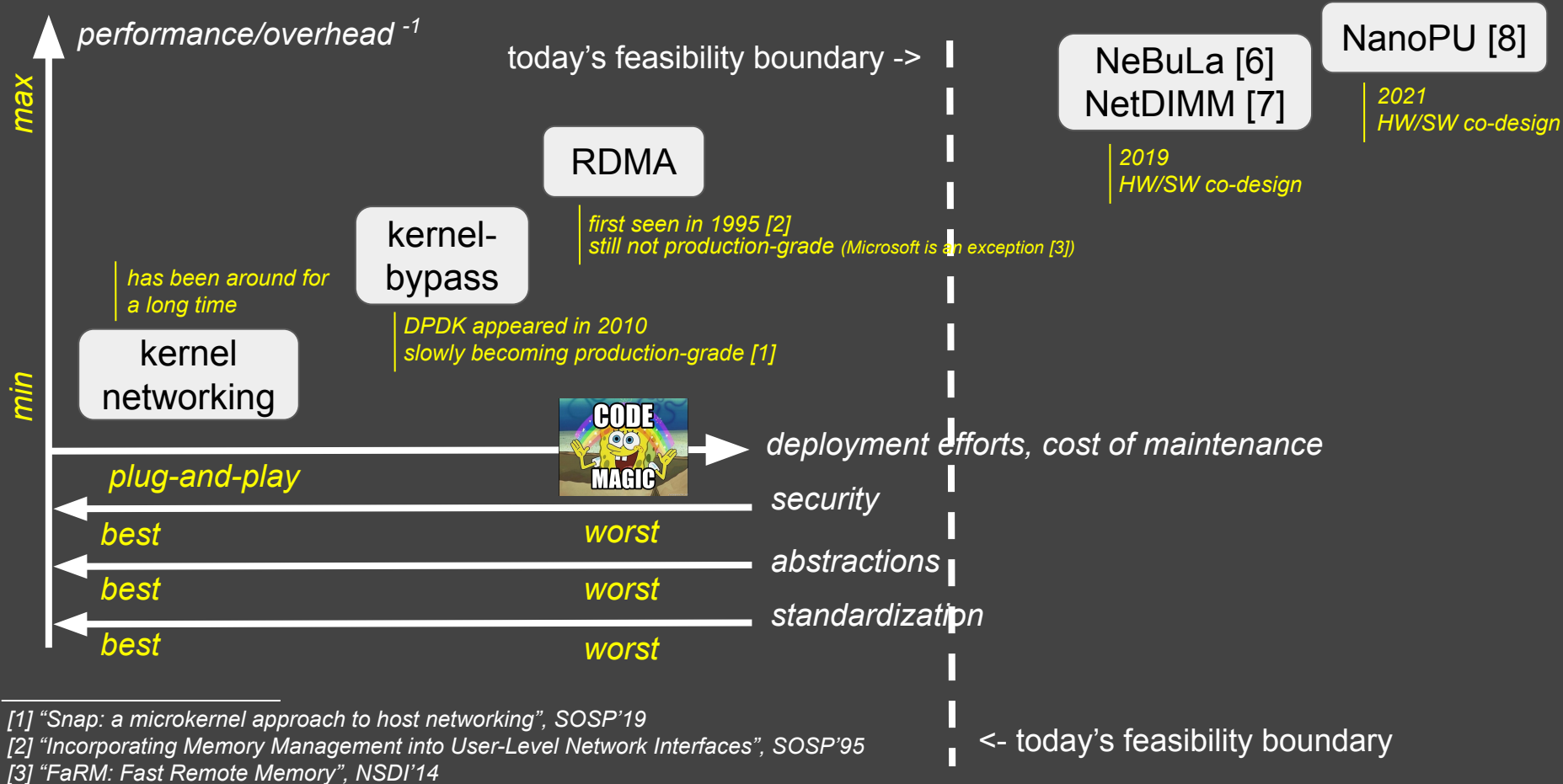
Future Work



[1] "Snap: a microkernel approach to host networking", SOSP'19

[2] "Incorporating Memory Management into User-Level Network Interfaces", SOSP'95

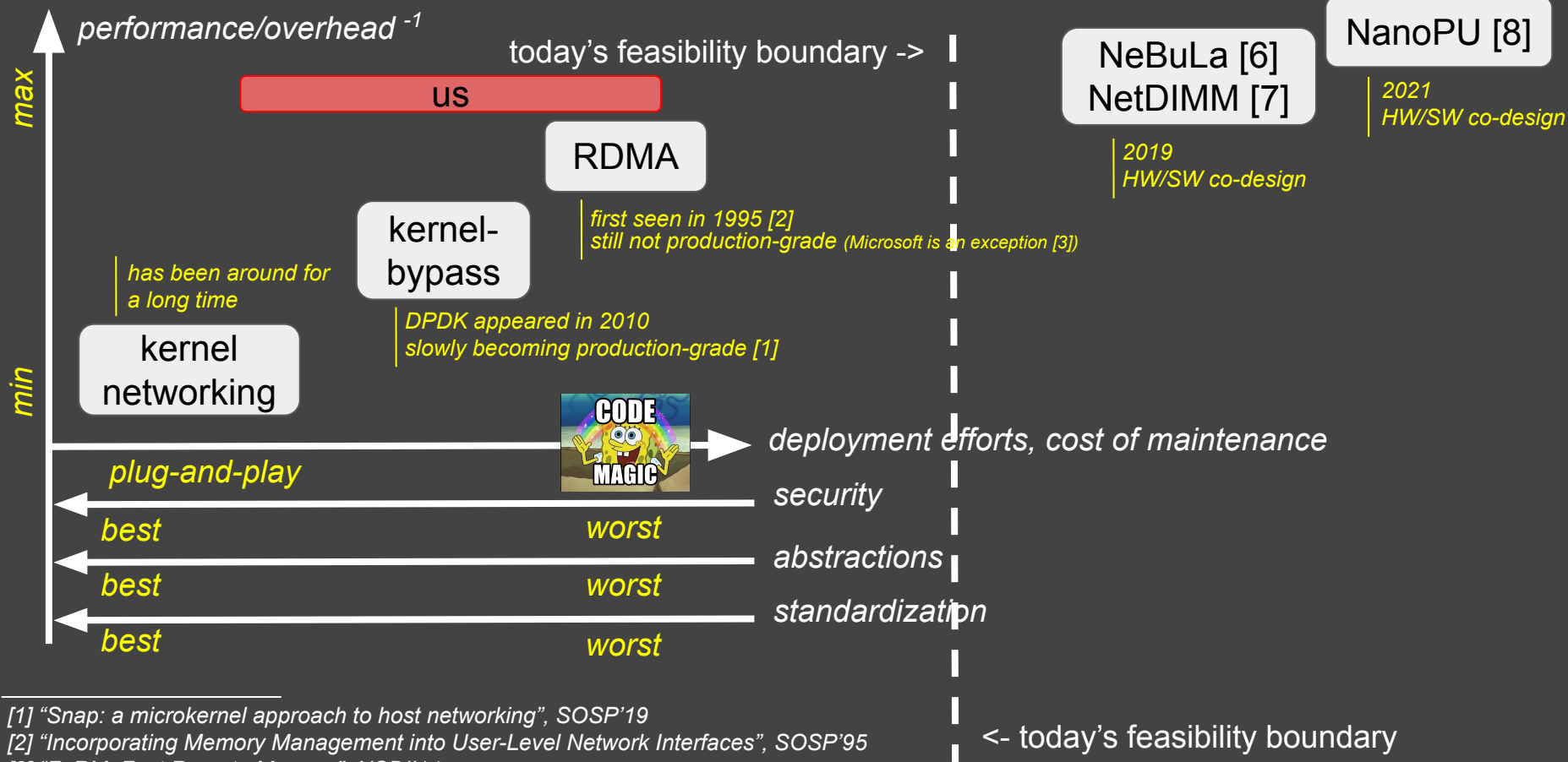
[3] "FaRM: Fast Remote Memory", NSDI'14





Existing Technologies -- Where We Are

9



[1] "Snap: a microkernel approach to host networking", SOSP'19

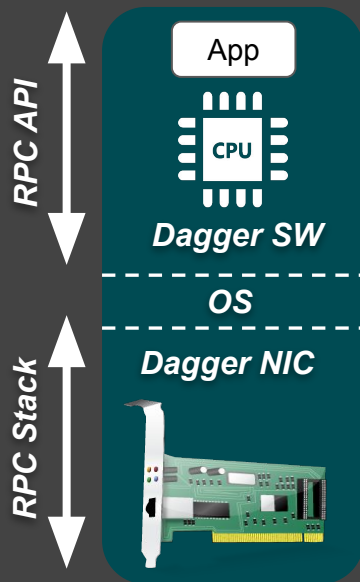
[2] "Incorporating Memory Management into User-Level Network Interfaces", SOSP'95

[3] "FaRM: Fast Remote Memory", NSDI'14



Dagger:

a HW/SW Co-Designed End-Host RPC Stack





Trends in Cloud Computing

Existing Technologies

Dagger: Three Design Principles

Dagger: Design and Implementation

Evaluation

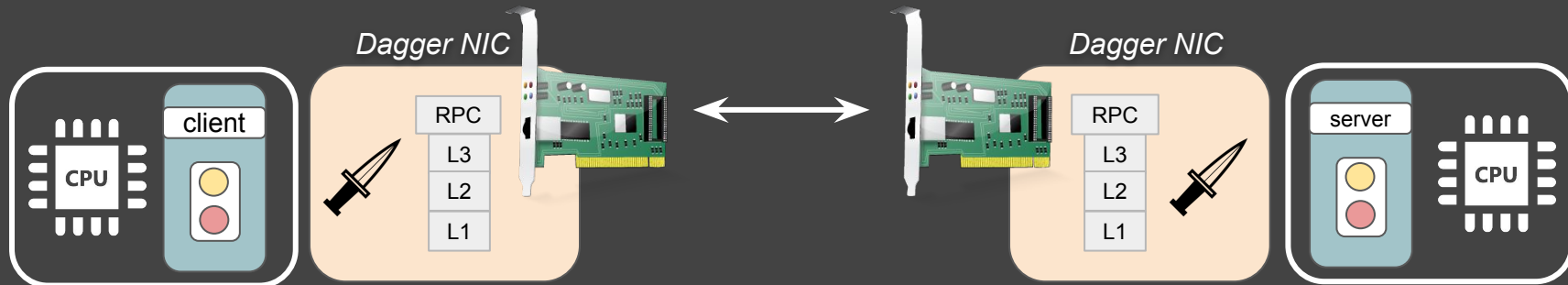
Future Work



Dagger's Principle

Design a hardware NIC for end-host communication stacks, from the L1 (PHY) layer, and all the way up to the application (RPC) layer

Completely free CPU from doing any work related to data exchange





- Networking protocols
- Load balancers
- Threading models
- Data representation
- Data manipulation, etc.

For hardware-based RPC stack to be equivalently flexible, they should also be configurable



Dagger is Based on an FPGA!



FPGAs have been already identified as a good physical medium for networking devices

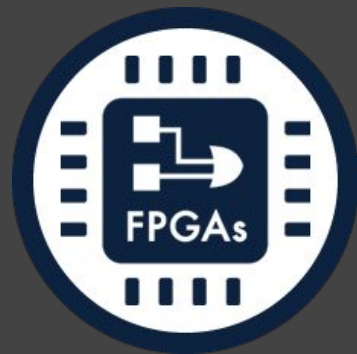
With FPGAs, hardware-offloaded RPC stacks can be fully customizable

100
GbE



Configurable Transport

- UDP (current transport)
- TCP for FPGA (M Ruiz, D Sidler, FPL'19)
- mTCP (E. Jeong, NSDI'14)
- HOMA (B. Montazeri, NSDI'20)
- TONIC (M. Arashloo, NSDI'20)

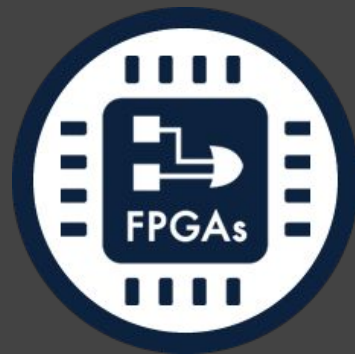


Transport



Configurable Load Balancer/Flow Controller

- *Static*
- *Round-robin*
- *Random*
- *Application-specific*
(e.g., key-based for partitioned KVS systems)



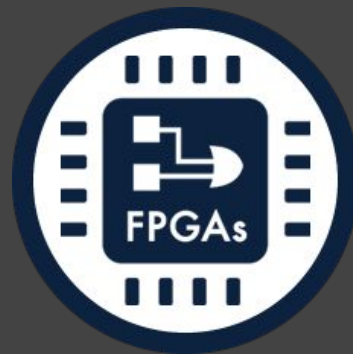
Load
Balancer

Transport



Configurable Host-NIC Interface

- *PCIe doorbells*
- *PCIe MMIOs*
- *Coherency-based*



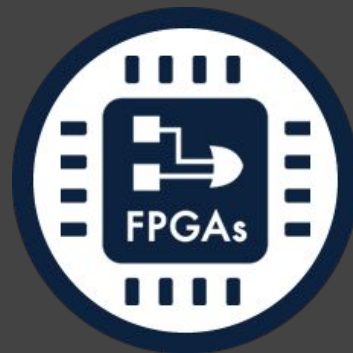
Host-NIC
interface

Load
Balancer

Transport



Configurable Threading Model



Host-NIC
interface



Load
Balancer

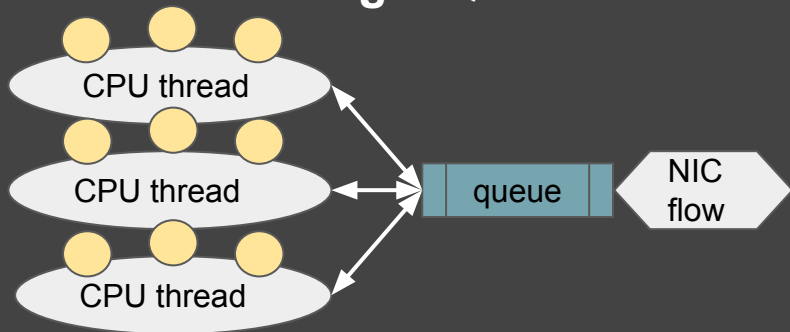
Transport



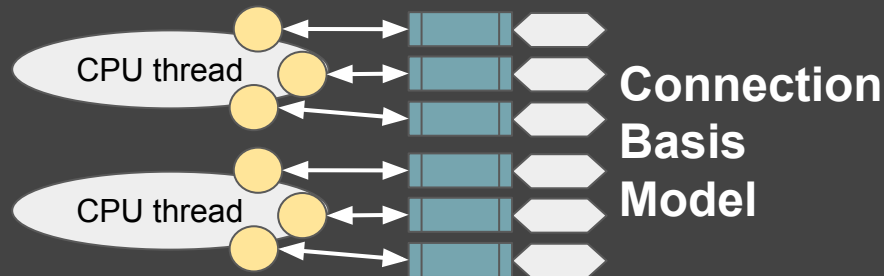
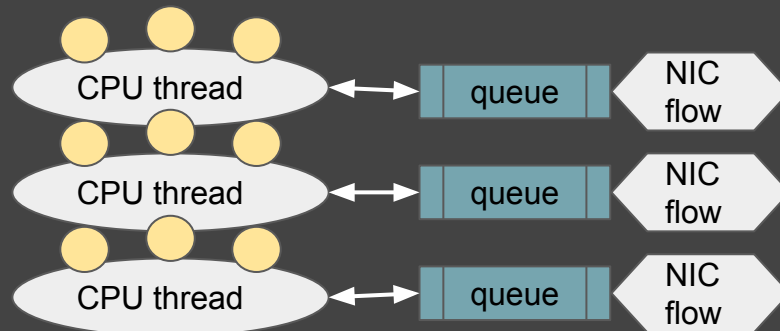
Dagger's Threading Model

- *Connection/thread/queue/flow mapping*
- *Number of NIC flows/queues*

Single Queue Model



Shared Queue Model



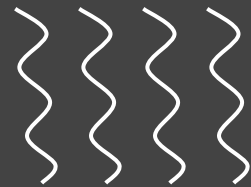


Dagger's Threading Model

➤ RPC handling

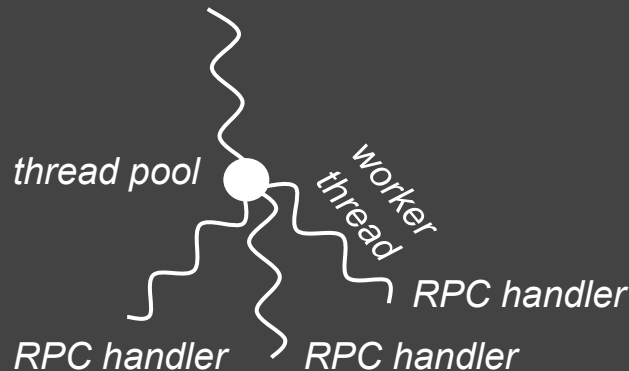
- *dispatch threads* (good for short RPCs with low latency requirements)
- *worker threads* (good for long-running RPCs)

dispatch threads
(4 flows shown)



RPC handlers

dispatch thread
(1 flow shown)





Dagger is Based on a Cache-Coherent FPGA Tightly-Coupled with the Host CPU over a NUMA interconnect

Inspired by:

- *series of RDMA/Kernel-bypass studies: FaSST (A. Kalia, OSDI'16), eRPC (A. Kalia, NSDI'19), and [1]*

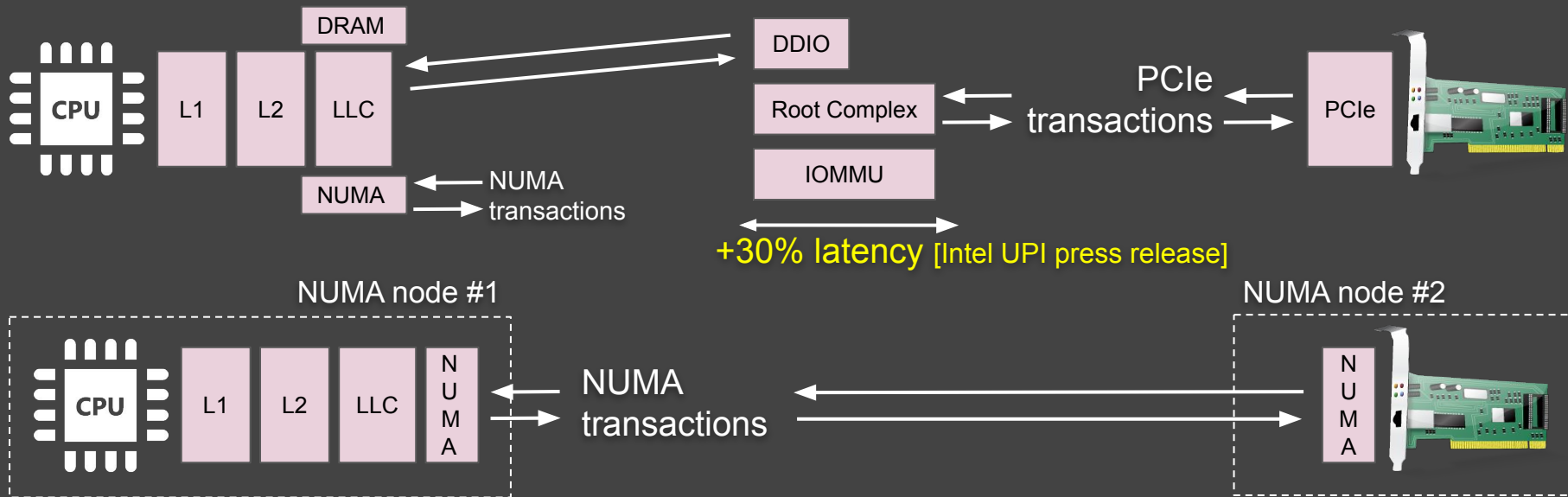
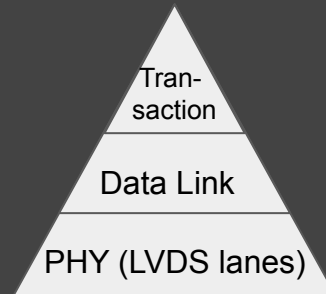
Known fact: *PCIe introduces non-negligible overheads for small messages in high-performance request-response stacks*

[1] Anuj Kalia, Michael Kaminsky, and David G. Andersen. *Design Guidelines for High Performance RDMA Systems. USENIX Annual Technical Conf. (ATC) (2016)*



What's the issues with PCIe?

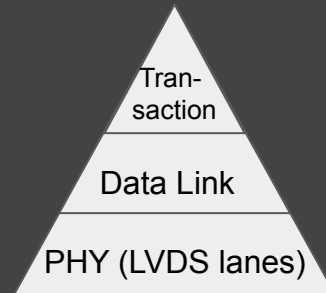
- physically, it's same as NUMA interfaces
- the **first** difference is in integration



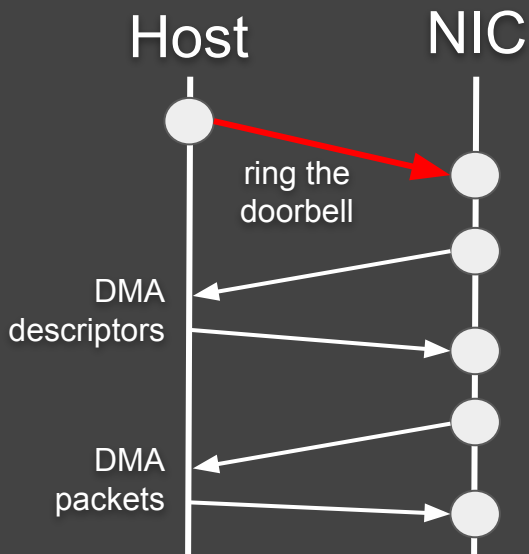


What's the issues with PCIe?

- physically, it's same as NUMA interfaces
- the **second** difference is in transactions



The Doorbell scheme

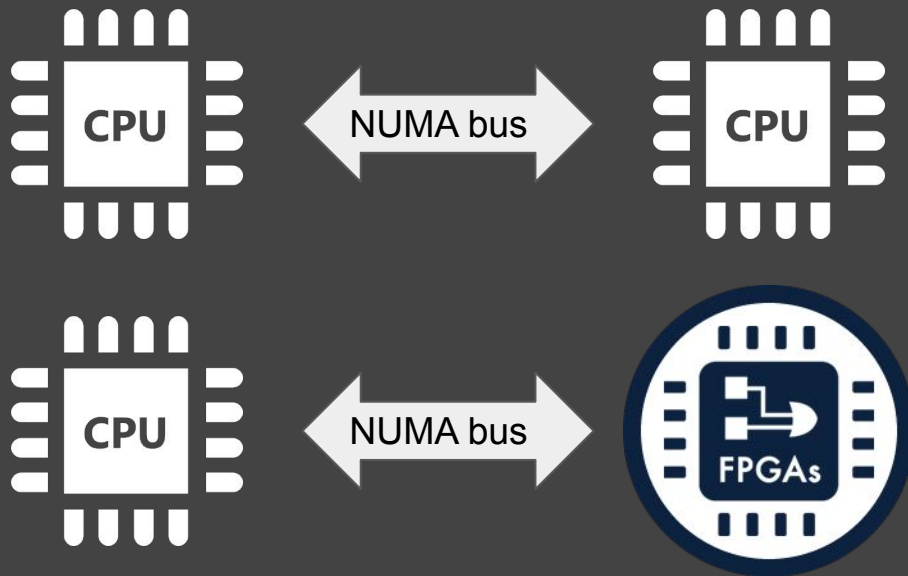
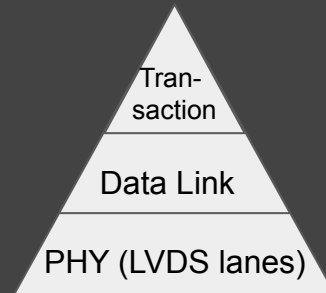


- multiple PCIe roundtrips
- expensive and CPU-inefficient strongly-ordered rings based on MMIOs, might take x100 CPU cycles (*R. Zambre, ICPP'19*)
- the CPU needs to do the job, even with DMA



What's the issues with PCIe?

- physically, it's same as NUMA interfaces
- the **second** difference is in transactions



- ✓ no DMAs are required to exchange data between NUMA nodes
- ✓ no explicit MMIO requests
- ✓ **minimal** software overhead
- ✓ NUMA interconnects have lower latency: *~250ns* vs *~450ns* for PCIe



What happens behind the NUMA protocols?

- exact transactions depend on exact NUMA specification
- so far, we do whatever we can do with CCI-P*

Throughput: low

- when CPU writes objects, it first needs to **Invalidate** the cache line in the FPGA
- FPGA is listening for **Invalidation** messages
- when **Invalidation** is received, FPGA requests data as read **Shared**

“interrupts”

Throughput: high

- FPGA starts polling CPU to get data fast

“polling”

[*] CCI-P, open-specification wrapper on Intel UPI



Trends in Cloud Computing

Existing Technologies

Dagger: Three Design Principles

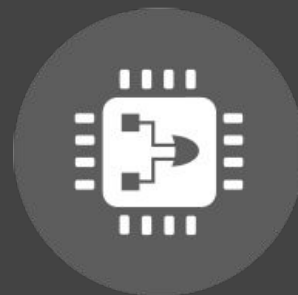
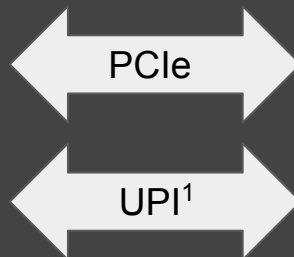
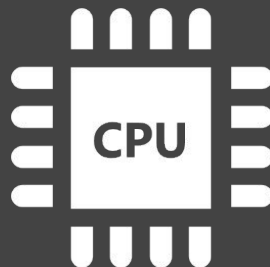
Dagger: Design and Implementation

Evaluation

Future Work



Hardware Platforms: CPU/FPGA hybrid

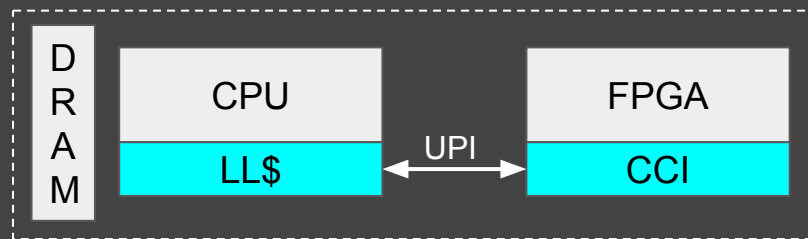


¹ Ultra Path Interconnect

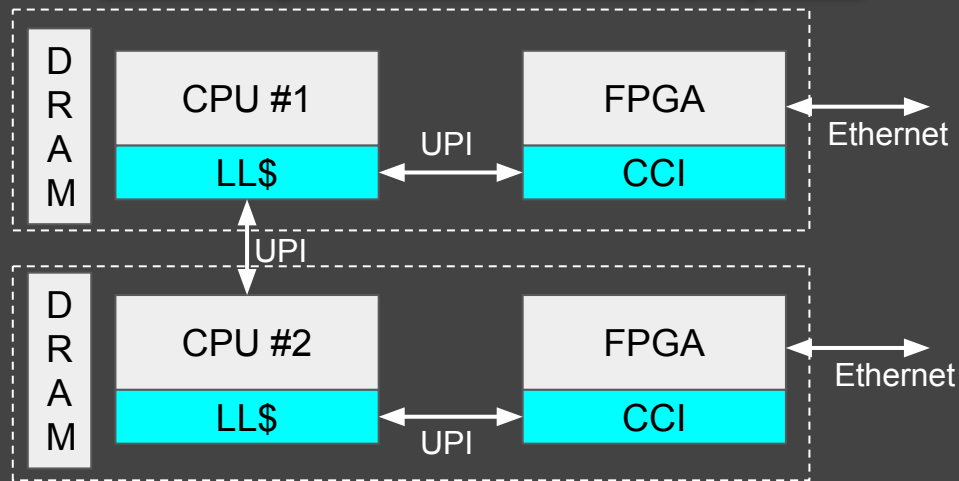
- *Intel Broadwell Xeon/FPGA hybrid, 2016*
- *Intel Skylake Xeon/FPGA hybrid, 2018*
- *Skylake PAC Arria10 discret FPGA platform, 2018*



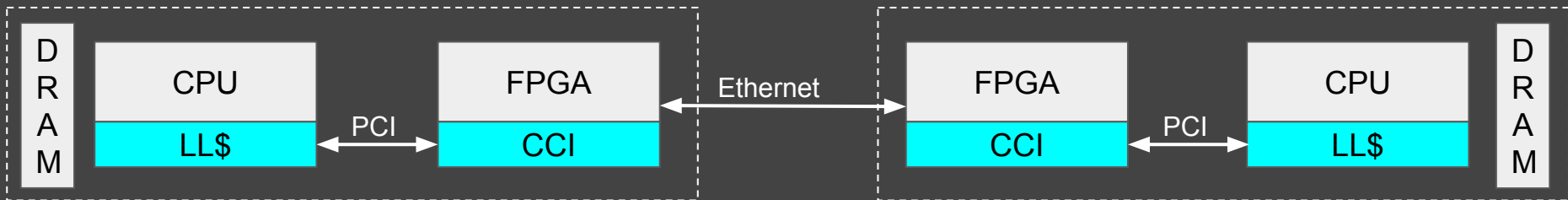
Intel Broadwell Xeon/FPGA hybrid



Intel Skylake Xeon/FPGA hybrid



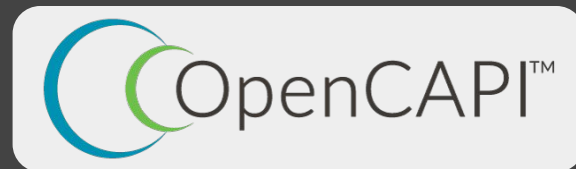
Skylake PAC Arria10 discret FPGA platform





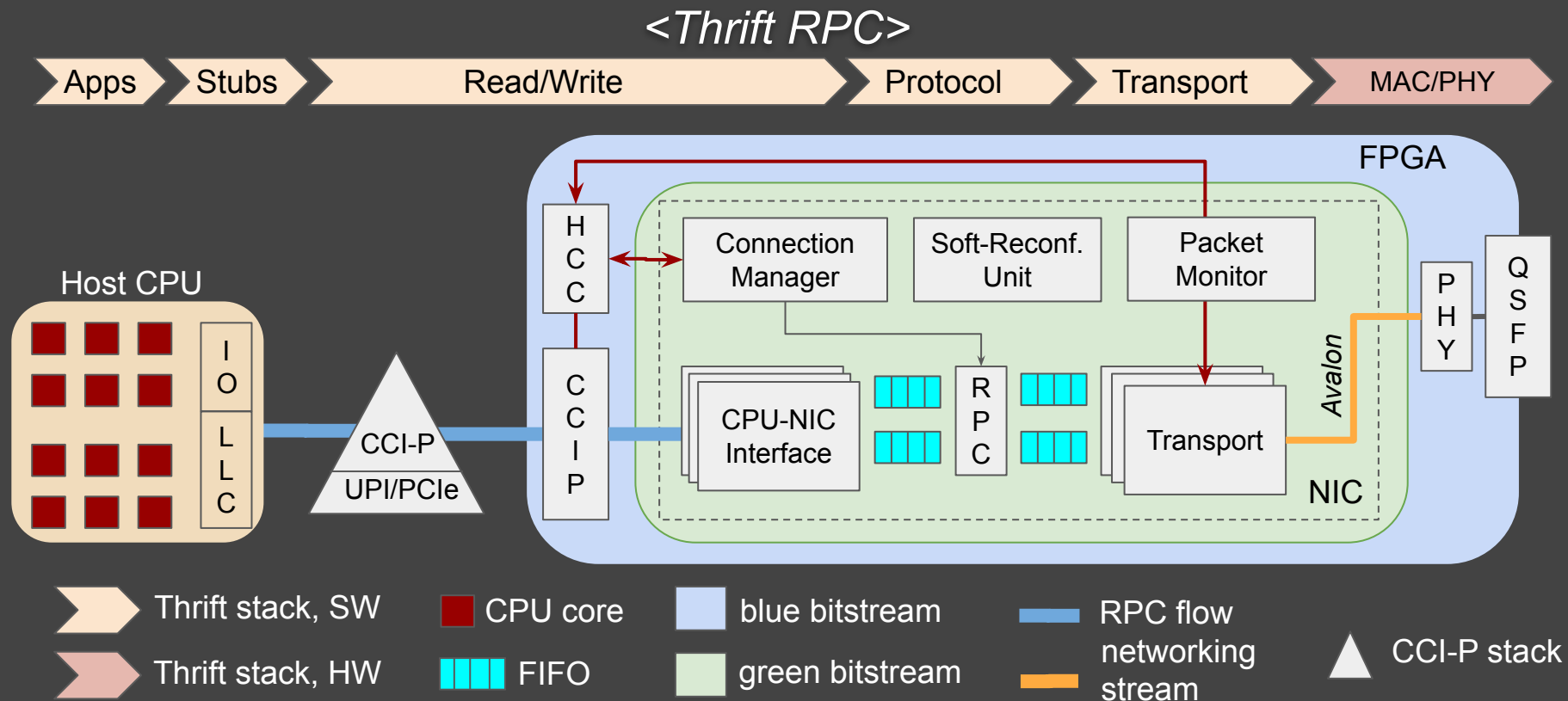
Alternative platforms:

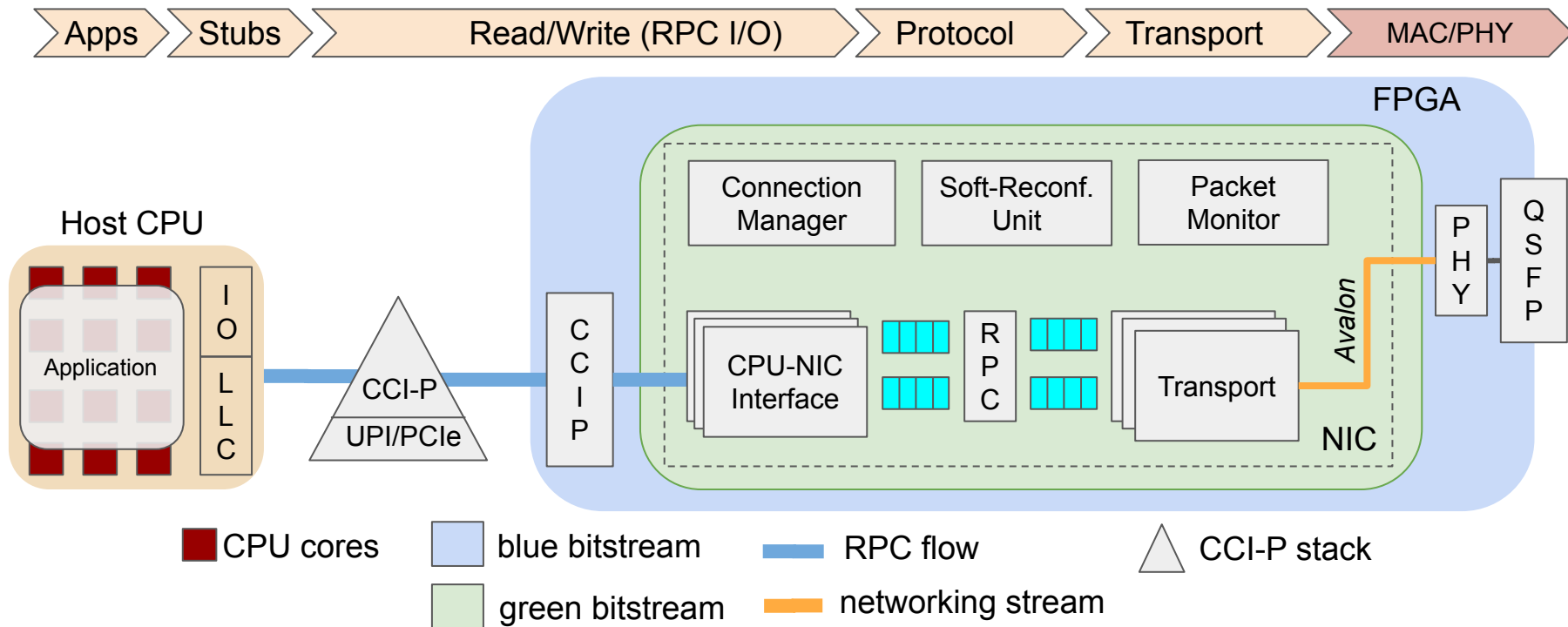
- *IBM POWER9 with OpenCAPI, 2017*
- *Intel Stratix 10 DX with hard UPI IP, 2019*
- *Intel Agilex FPGA with CXL¹, upcoming*
- *ETH Enzian with ECI², upcoming*



¹ Compute Express Link, <https://www.computeexpresslink.org/>

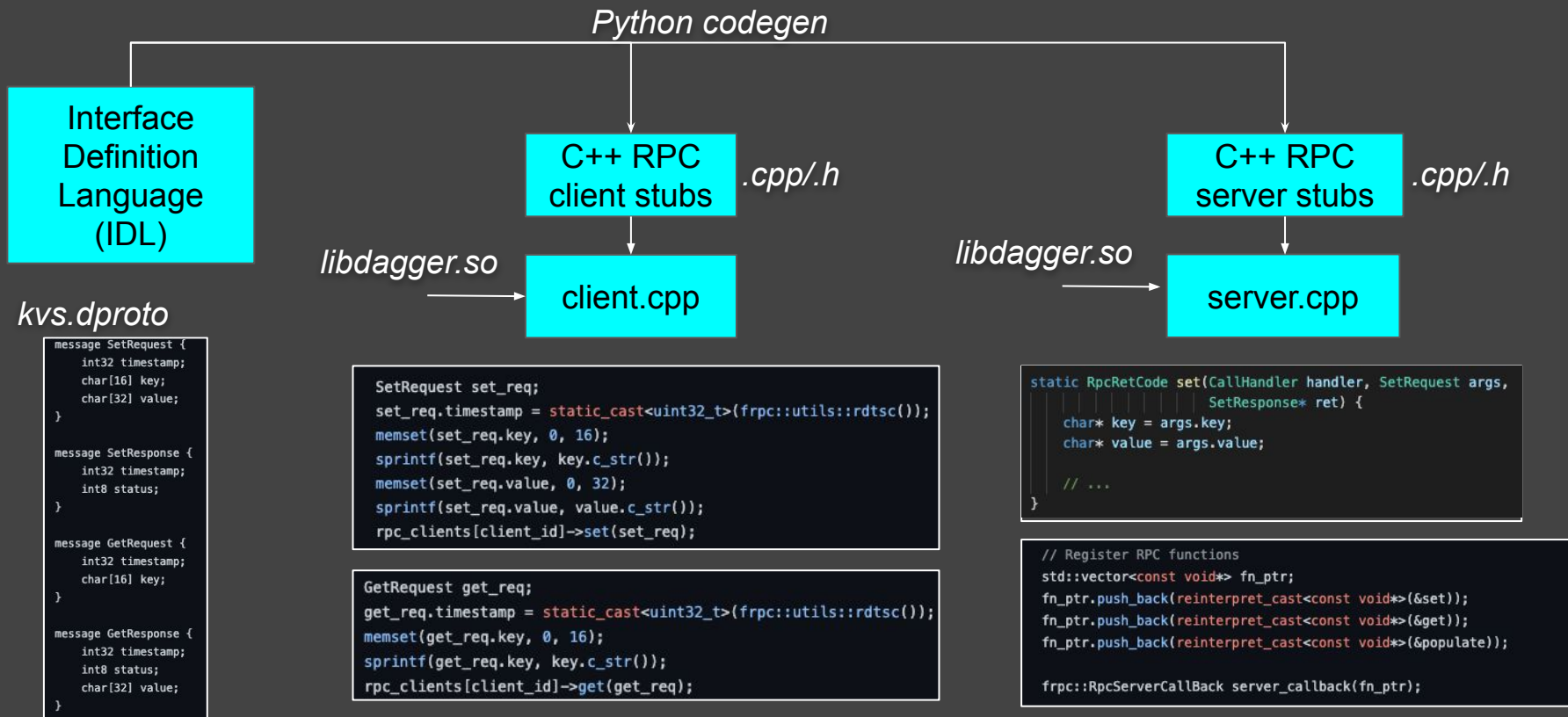
² Enzian Coherence Interface, <http://enzian.systems/>







High-level RPC calls (similar to gRPC, Thrift)





Trends in Cloud Computing

Existing Technologies

Dagger: Three Design Principles

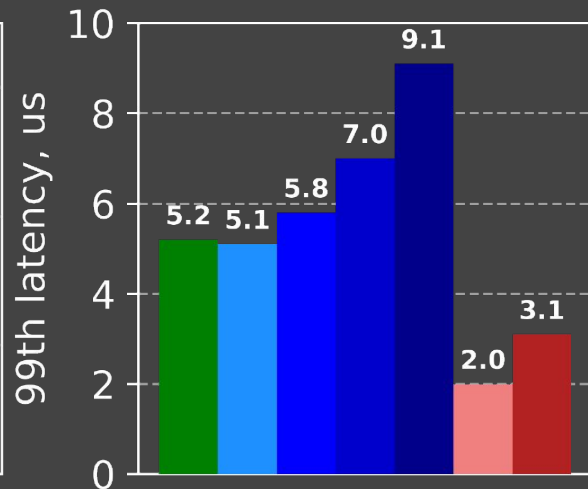
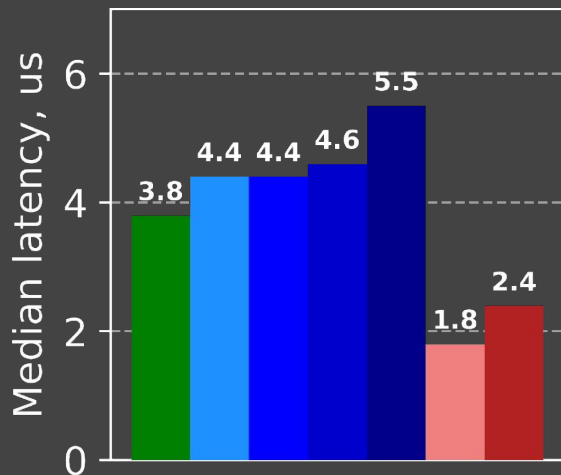
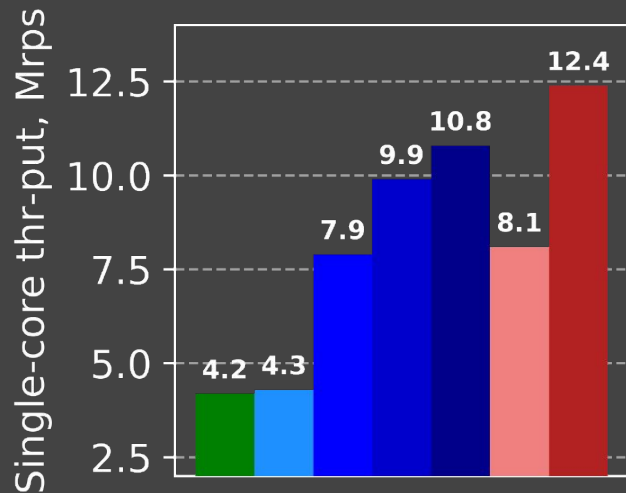
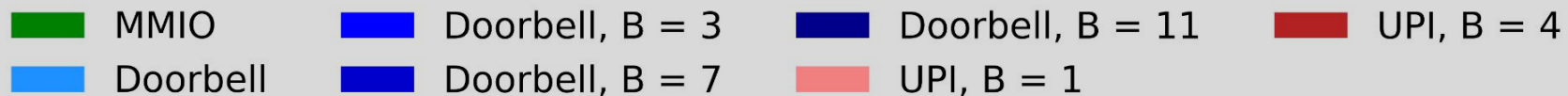
Dagger: Design and Implementation

Evaluation

Conclusion



UPI vs PCIe: *end-to-end, single core, loopback mode, 64B requests*

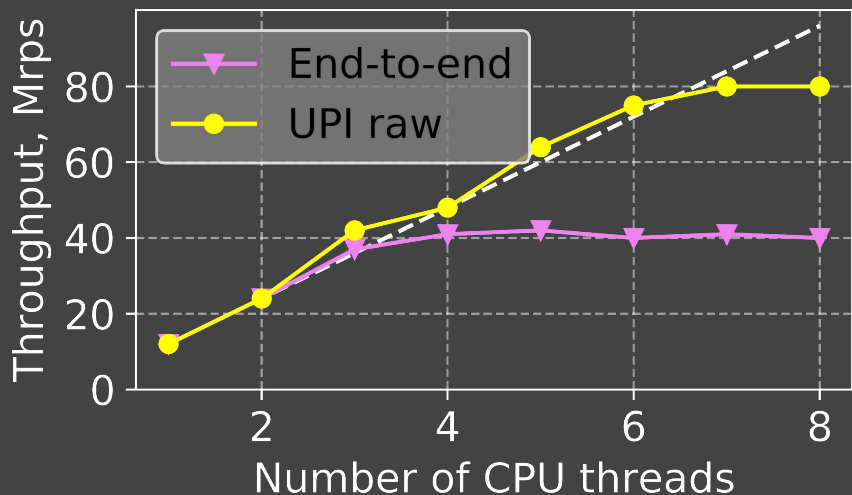


x1.1 - x1.8 higher throughput, **x2.5** lower latency



Throughput vs Number of Cores:

end-to-end performance, loopback mode, 64B requests



Scalability limitation:

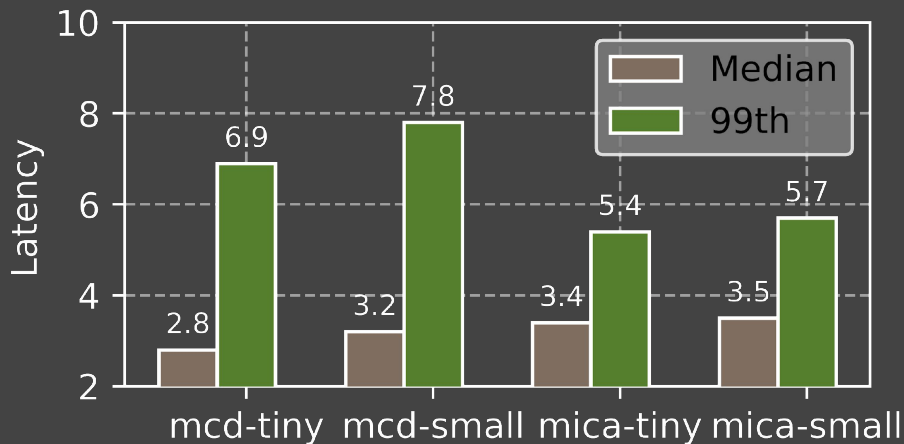
- *Socket-to-Socket (NUMA) path is faster*
- *UPI is implemented using soft logic on the FPGA*
- *Upcoming generation of FPGAs address it*

*Throughput scales linearly up to **4 cores to 40 (80) Mrps***



Memcached and MICA [1]: end-to-end performance, loopback mode

- *tiny*: $\langle k, v \rangle = \langle 8B, 8B \rangle$
- *small*: $\langle k, v \rangle = \langle 16B, 32B \rangle$



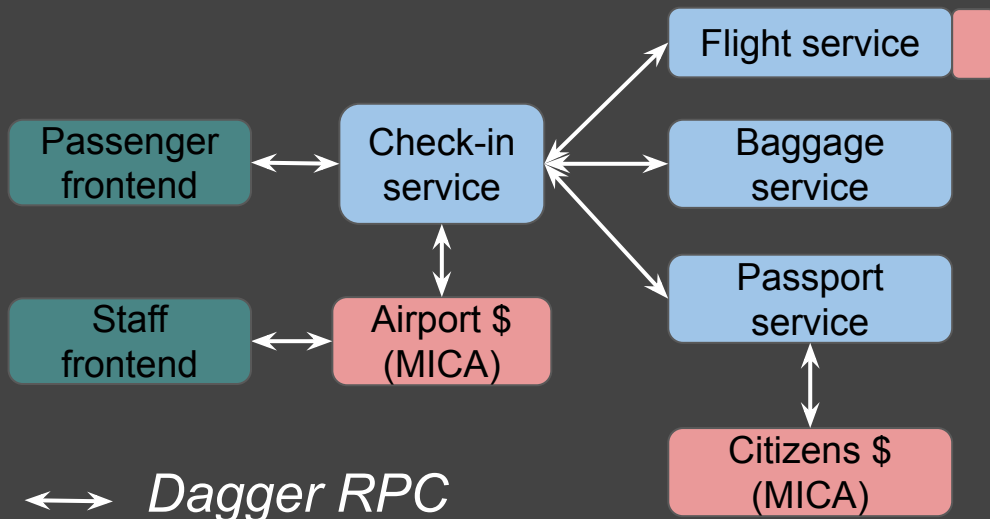
✓ **50 LoC/200 LoC** for Memcached/MICA

✓ *us-Scale remote KVS access latency, **x4.4 lower** than [1] (DPDK)*



Dagger Supports Multi-Tenancy and Can Be Used in Chains of Microservices

Synthetic flight registration application



➤ *Instance of Dagger per microservice*

➤ *Different configuration per instance*

- *load balancer*
- *threading model*



After applying some more aggressive optimizations:

- *single-core throughput boosted to 21.5 Mrps*
- *no latency penalty*

| | DPDK (IX[1]) | DPDK (eRPC[2]) | RDMA (FaSST[3]) | Dagger | Dagger (potential*) |
|---------------------------------|-----------------|-------------------|--------------------|--------|------------------------|
| median latency, us | 11.4 | 2.3 | 2.0 | 2.1 | 1.3 |
| single-core throughput, Mrps | 0.7 | 10 | 9.6 | 21.5 | 21.5 |

[*] *hard logic-based NUMA implementation*

[1] "IX: A Protected Dataplane Operating System for High Throughput and Low Latency", OSDI'14

[2] "Datacenter RPCs can be General and Fast", NSDI'19

[3] "FaSST: Fast, Scalable and Simple Distributed Transactions with Two-Sided (RDMA) Datagram RPCs", OSDI'16



Trends in Cloud Computing

Existing Technologies

Dagger: Three Design Principles

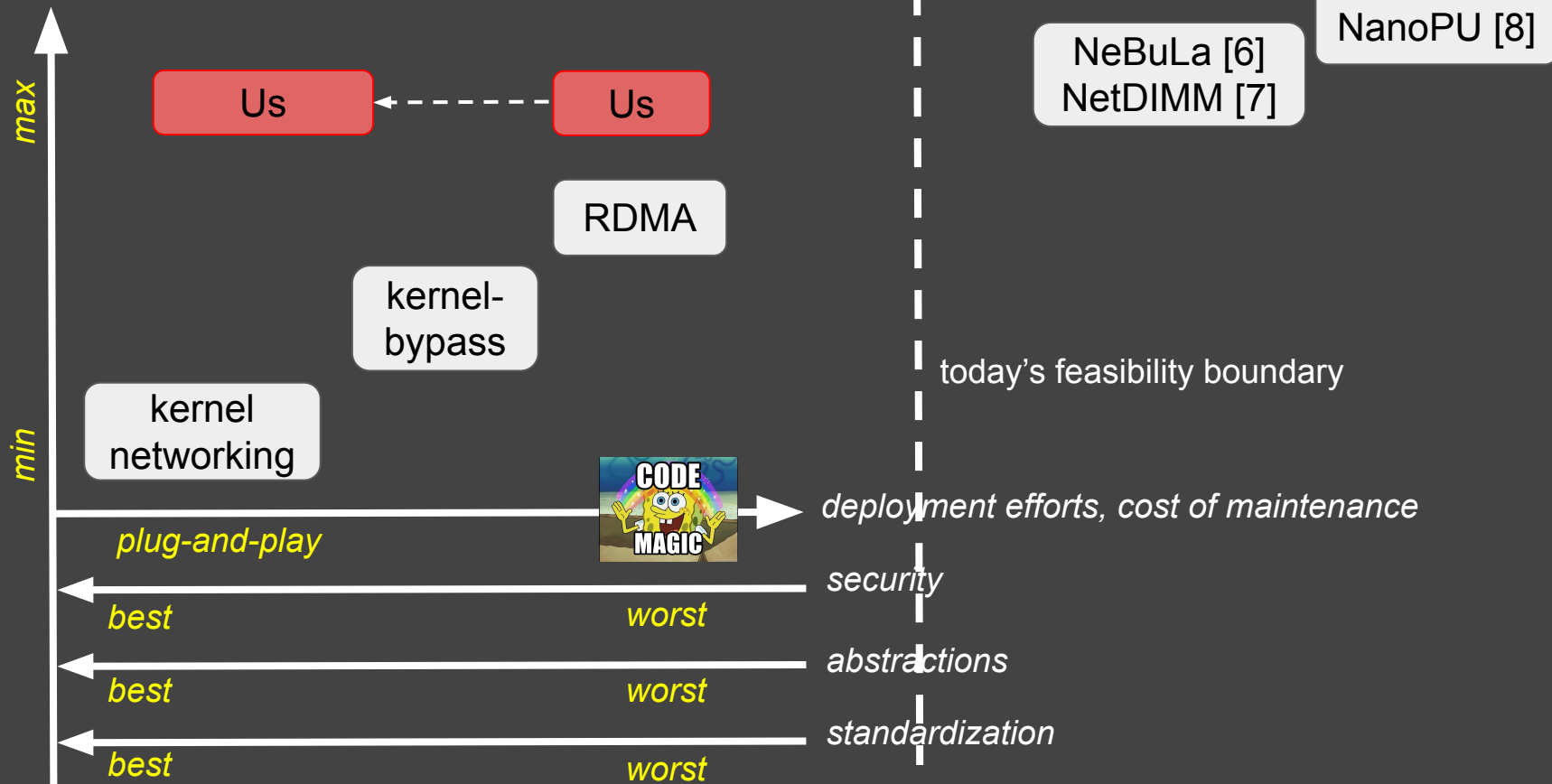
Dagger: Design and Implementation

Evaluation

Future Work



performance/overhead⁻¹





Data transformation (marshaling) to strength abstractions:

- *can take a lot of time and load the CPU [1]*
- *previous proposals suggest to offload to hardware [1,2,3]*
- *but what should be the right way to do this?*
 - *PCIe scatter/gather DMA?*
 - *NUMA shared memory?*
 - *something else like DSA [4]?*

[1] "Optimus Prime: Accelerating Data Transformation in Servers", ASPLOS'20

[2] "Breakfast of Champions: Towards Zero-Copy Serialization with NIC Scatter-Gather", HotOS'21

[3] "Zerailizer: Towards Zero-Copy Serialization", HotOS'21

[4] <https://software.intel.com/content/www/us/en/develop/articles/intel-data-streaming-accelerator-architecture-specification.html>



Multitenancy support and isolation **to strength security:**

- *how to partition FPGA?*
- *how to allocate FPGA resources between different “virtual” NICs?*
- *how to schedule PCIe/NUMA bandwidth between tenants?*
- *how to provide physical memory isolation?*



Do we really need NUMA interconnects?

- *physical layer is the same*
- *can we get similar performance/efficiency with optimized PCIe?*
 - *we believe, yes!*
 - *explore the new features of upcoming server processors:*
 - *weekly-ordered PCIe writes/reads*
 - *DSA[1]*
 - *CXL[2]*

[1] <https://software.intel.com/content/www/us/en/develop/articles/intel-data-streaming-accelerator-architecture-specification.html>

[2] <https://www.computeexpresslink.org/>



Thank You for Your Attention!

Q&A