Characterizing Off-path SmartNIC* for Accelerating Distributed Systems

Xingda Wei, Rongxin Cheng, Yuhan Yang

Rong Chen & Haibo Chen

IPADS, Shanghai Jiao Tong University

* We use the shorter version—**SmartNIC -**—to term **off-path SmartNIC** in this talk.

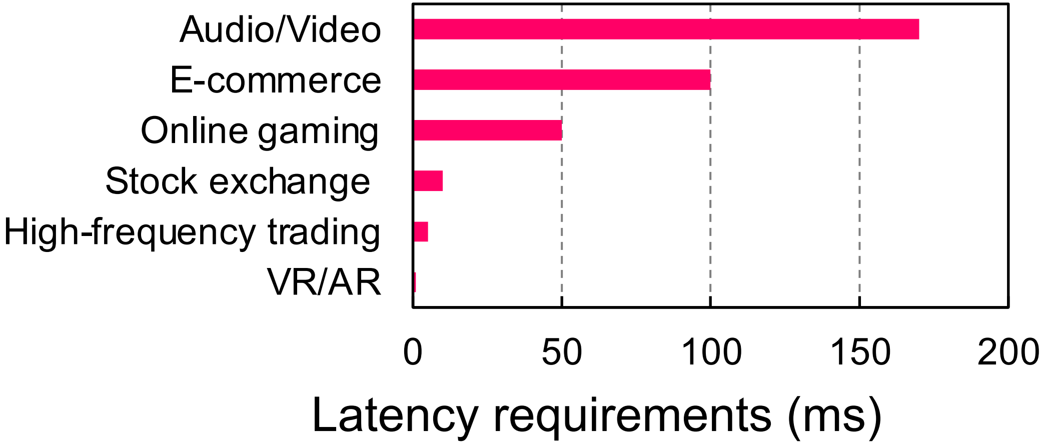
The demand for low latency & the trend for fast networking

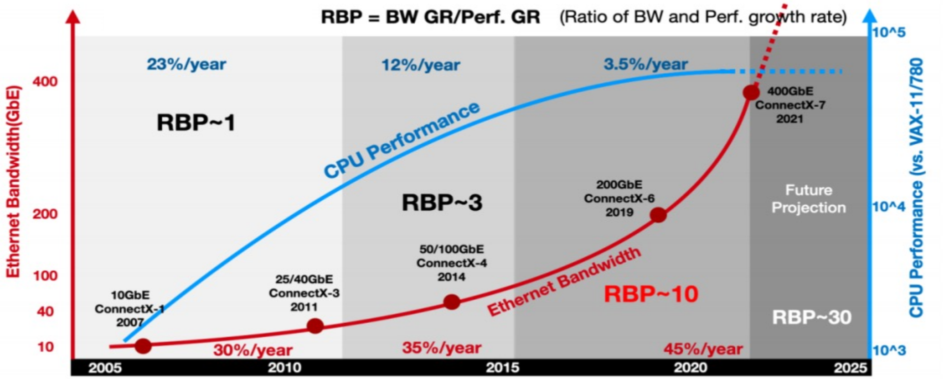
Applications require lower latency, even on the order of microseconds & high throughput

- E.g., VR/AR, high frequency trading, etc.

The networking is ultra-fast in terms of low latency & high throughput (bandwidth)

- Represent example: RDMA (Remote Direct Memory Access) & SmartNIC



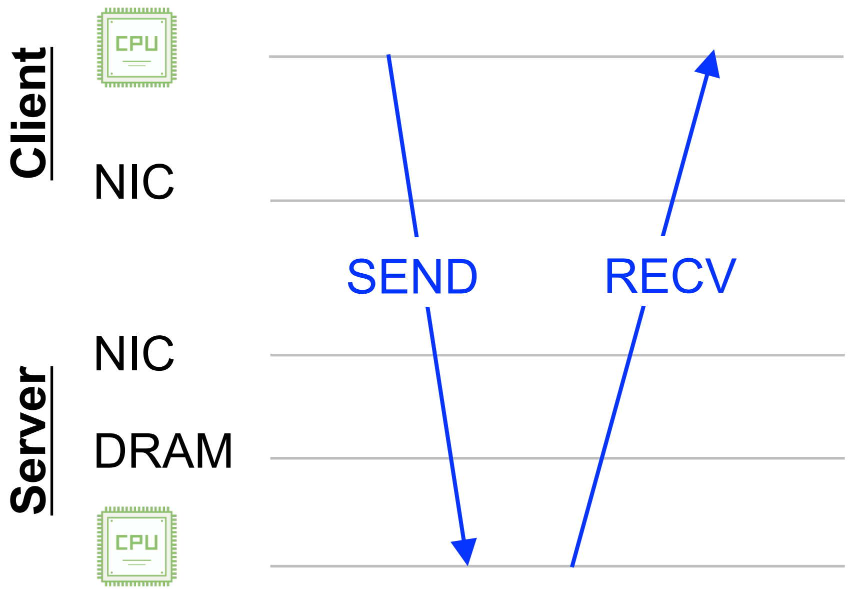


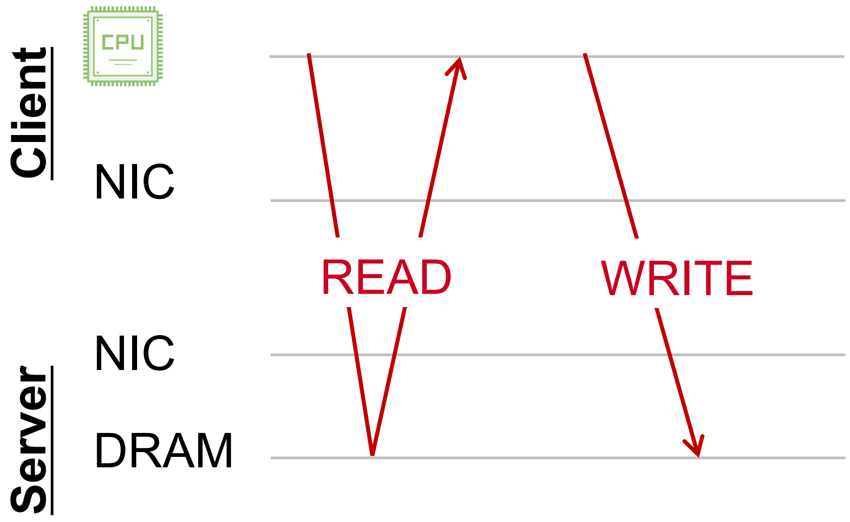
Source: https://redian.news/wxnews/78686

Before SmartNIC, RDMA is prevalent

From a system perspective, RDMA provides two primitives:

- Two-sided RDMA : SEND/RECV (like messaging in traditional network)
- One-sided RDMA: offloading primitive for memory READ/WRITE

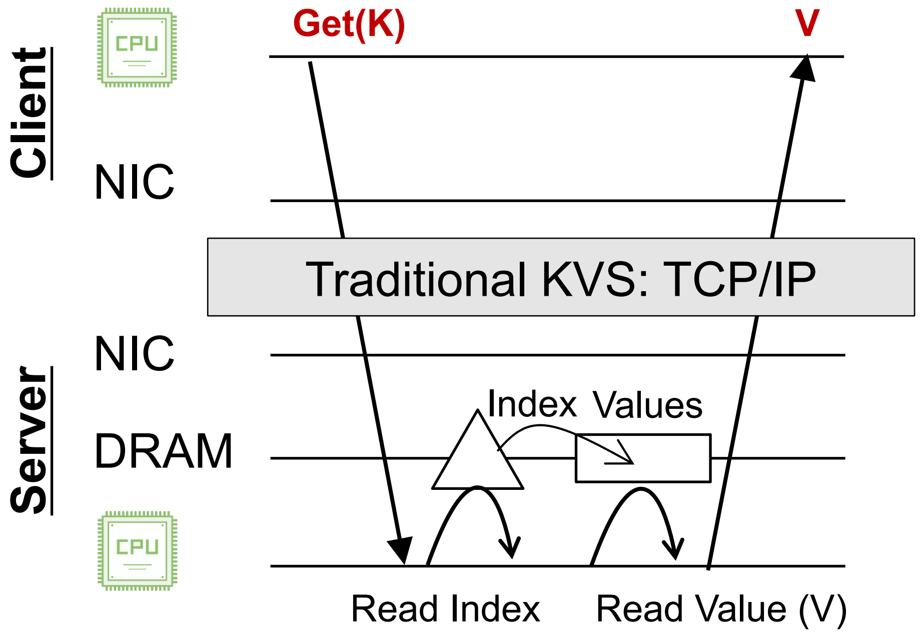




Optimizing systems w/ RDMA: basic approaches

Case study: in-memory key-value store (KVS), e.g., Redis

- **Key operation**: `Get(K) -> V` where `K,V` are stored on a server
- `Get(K)` requirement: high throughput & low latency

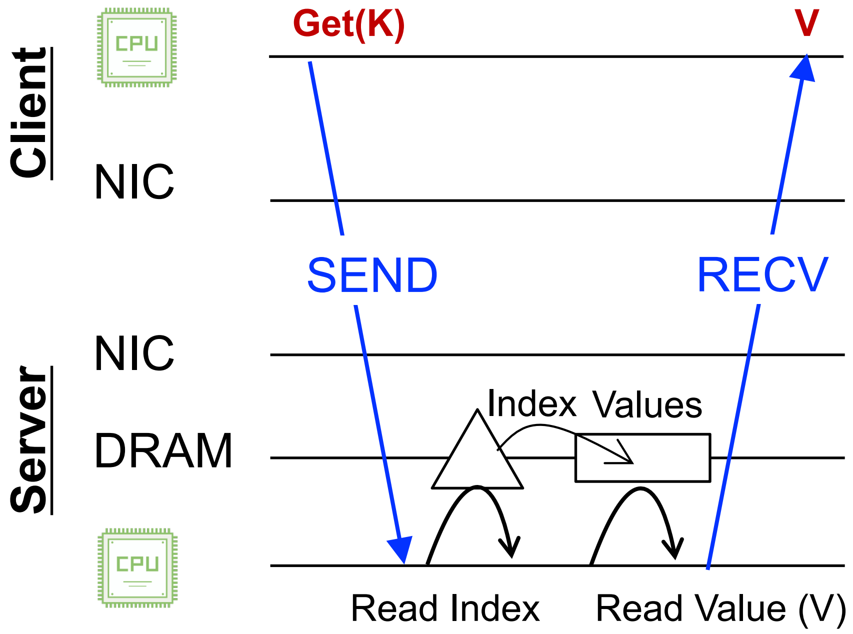


Optimizing systems w/ RDMA: basic approaches

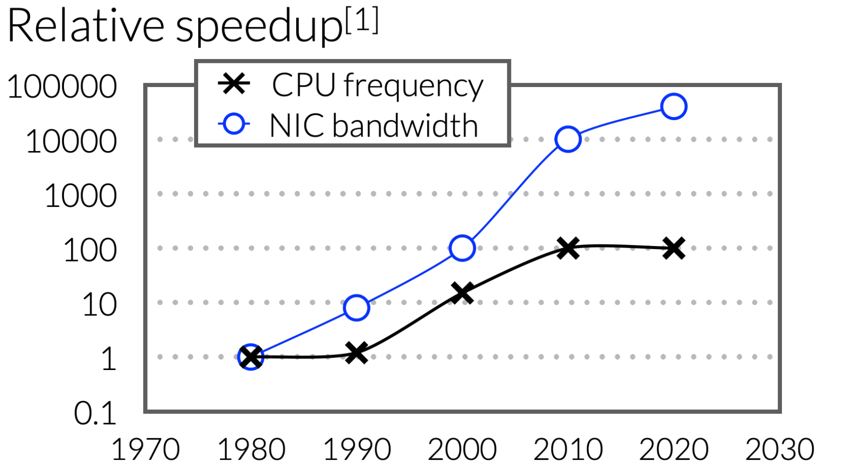
Using two-sided RDMA to optimize KVS

- Accelerate the network path with faster alternative
- Pros: the server CPU is left unoptimized/changed

Cons: server CPU may become the bottleneck, e.g., 150M reqs/sec(NIC) vs. 70M reqs/sec (CPU)



Setup: ConnectX-6 200Gbps RDMA NIC (RNIC), server: 24 cores Intel Xeon Gold 5316



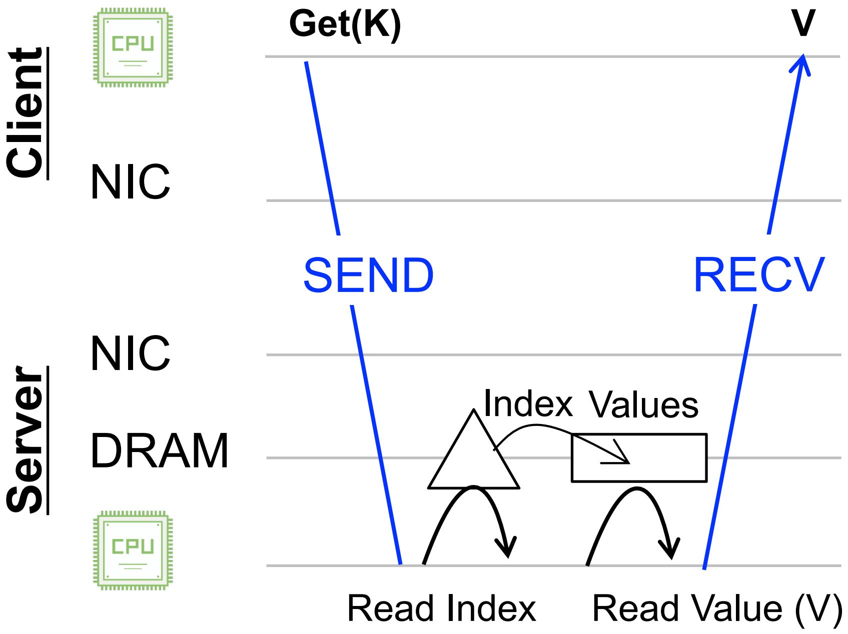
Year (StRoM: Smart Remote Memory, Eurosys'20)

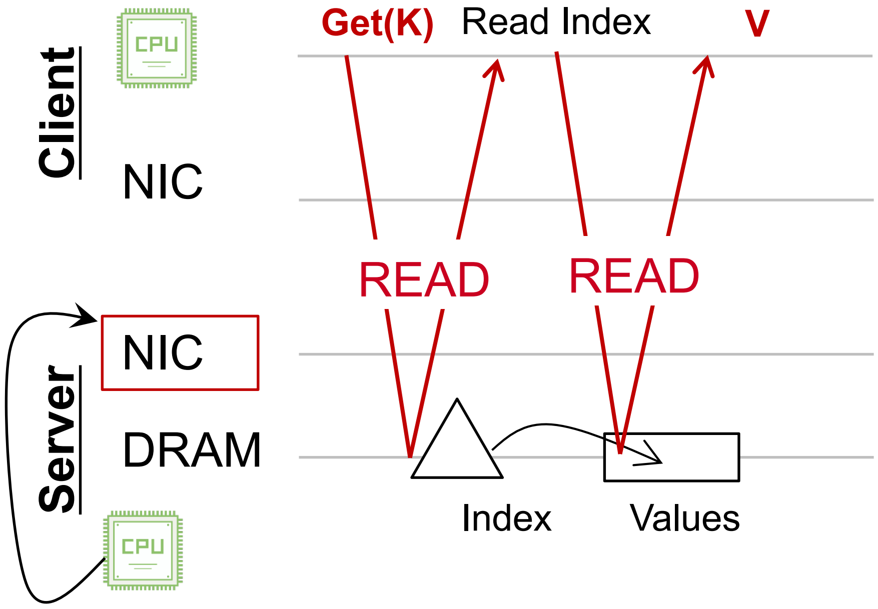
Optimizing systems w/ RDMA: basic approaches

Cons:
network amplification!

Using one-sided RDMA to optimize KVS

- Client directly execute the Get with the help of remote memory READ
- NIC can process READ much faster than server CPU





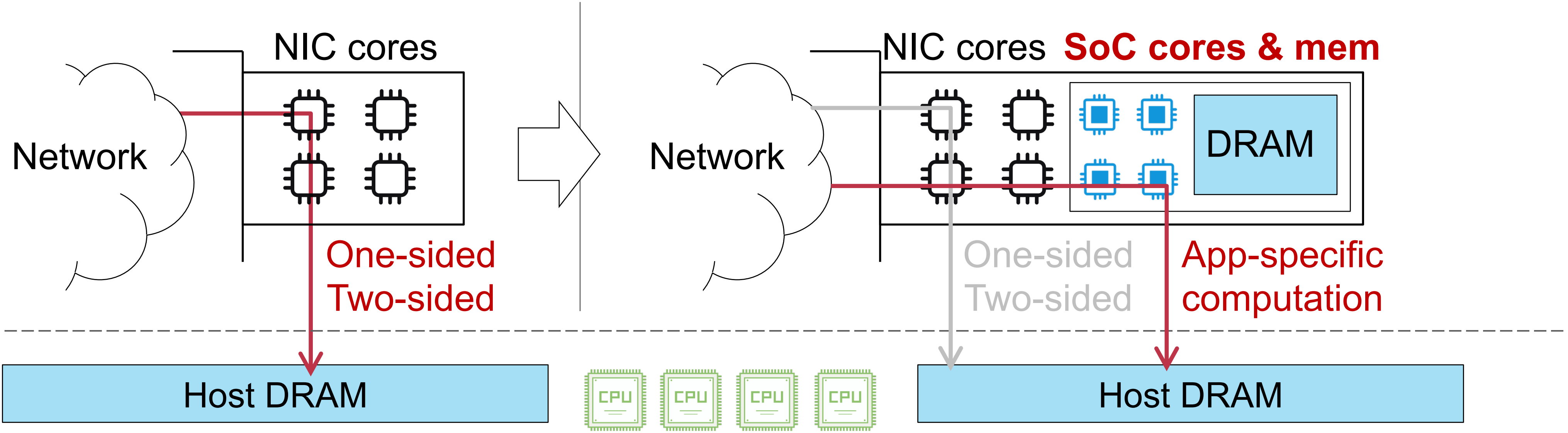
Offloaded computation

From RNIC ⇨ SmartNIC: larger offloading design spaces

The central processing unit RDMA-capable NIC (RNIC) are NIC cores

– ASIC that implements one-sided and two-sided operations (not programmable)

SmartNIC **extends RNIC** to support programmable capabilities

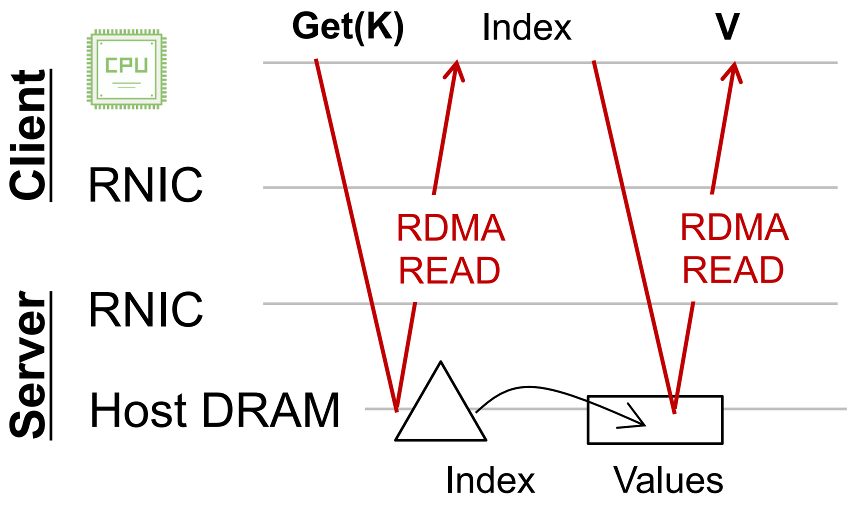


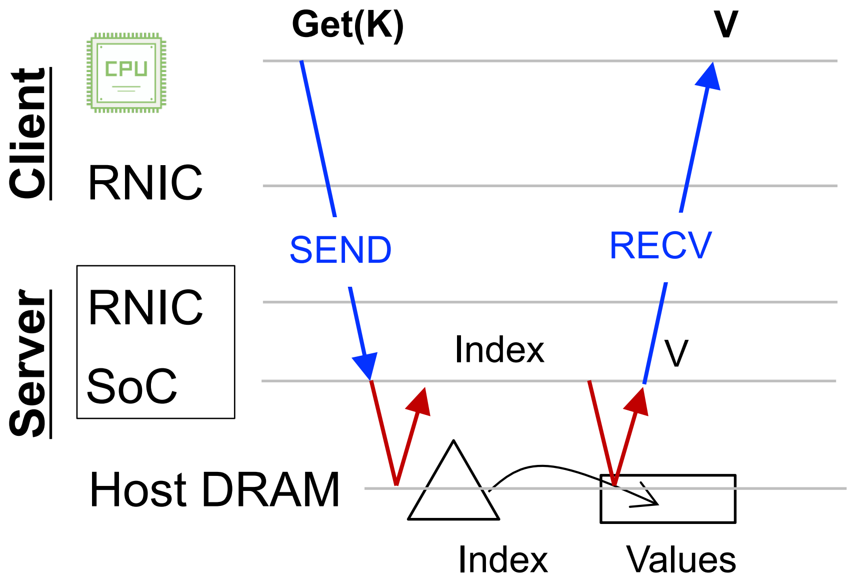
From one-sided RDMA to SmartNIC, does it help?

SmartNIC: RNIC equip with a programmable SoC (RNIC + SoC)

Back to our initial case study: `Get(K) -> V` **in key-value storage**

- We can use the programmability of SmartNIC to execute the Get() **w/o amplification**

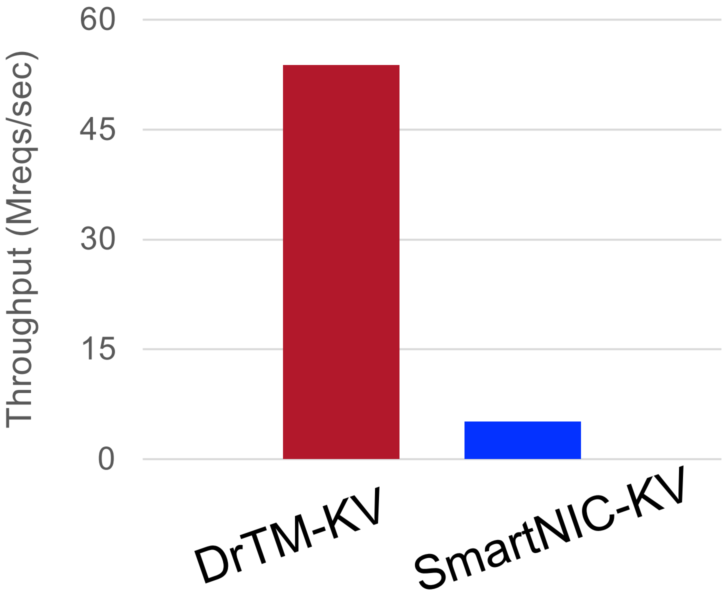


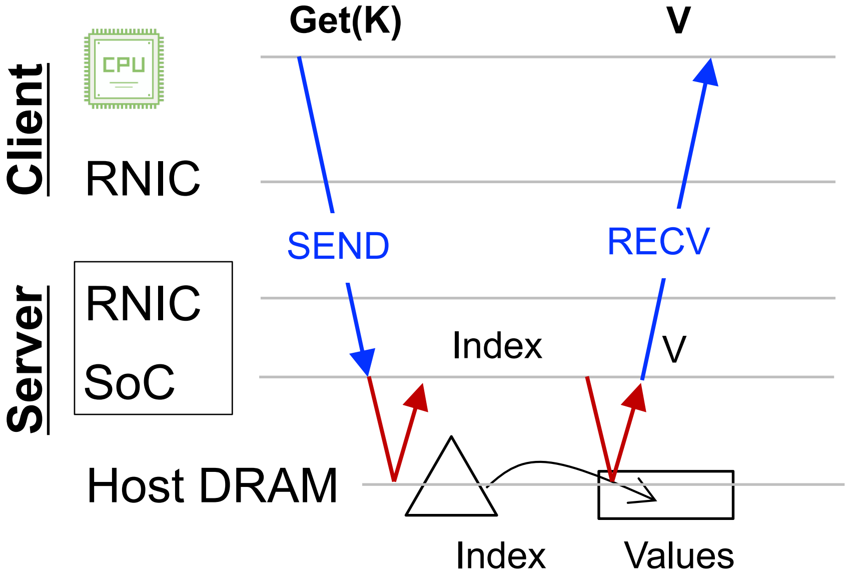


From one-sided RDMA to SmartNIC, does it help?

Our (naïve) SmartNIC-KVS is 14% of the RDMA-KVS !! (workload: YCSB-C (100% Get))

- RDMA-KVS: DrTM-KV [SOSP’15]
- SmartNIC-KVS: leverage SEND/RECV to offload Get to the NIC SoC





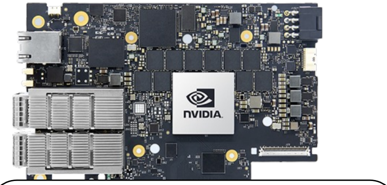
We decide to first characterize the SmartNIC before using it!

SmartNIC is more complex than we have thought

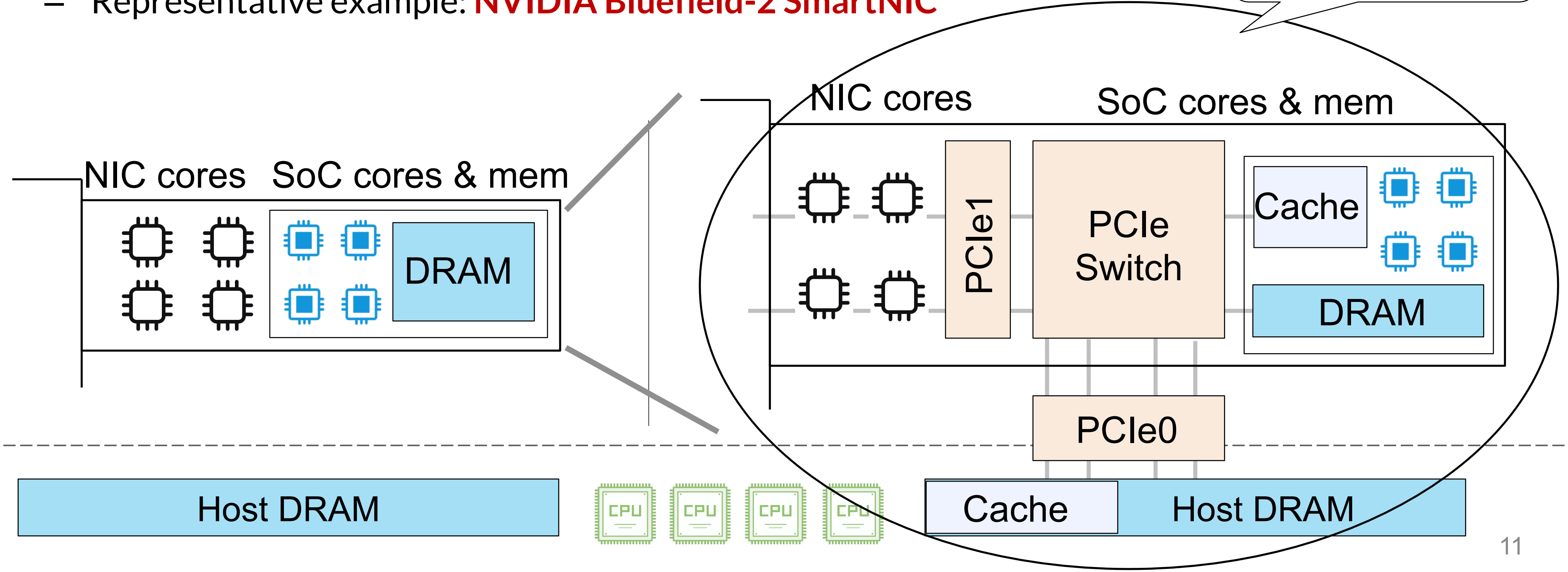
Many SmartNIC architectures exist (More complex than we thought)

We focus on off-path SmartNIC, a widely used SmartNIC architecture

- Representative example: **NVIDIA Bluefield-2 SmartNIC**



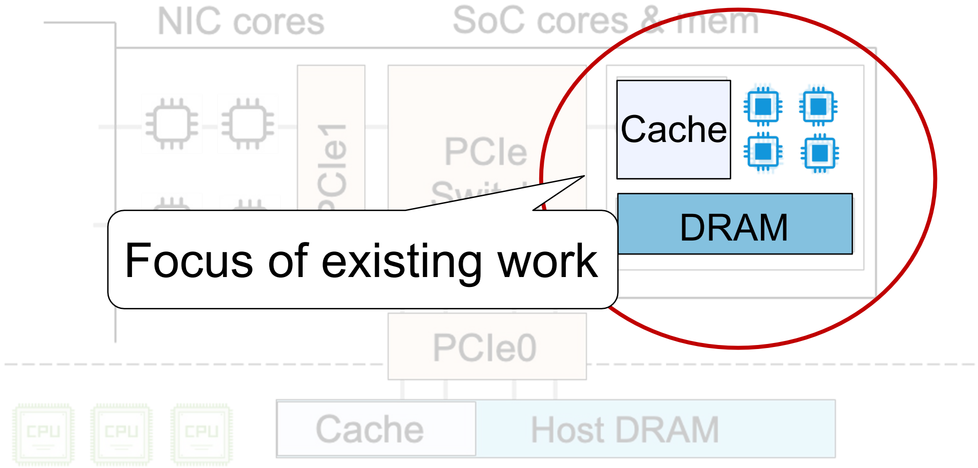
Discovered via exp + doc



Our work: the most comprehensive characterization on SmartNIC

Existing studies exist[1][2][3], which provides valuable insights

- The mostly focus on the offloading computation power of the SmartNIC
- A known takeaway is that: SmartNIC's SoC cores are *wimpier* than the host



	L1	L2	L3	DRAM
SoC	4x	4x	N/A	2x
Host	1x	1x	1x	1x

Memory access speed [1] (lower is better)

Benchmark	SoC	Host
Multi-core Coremark	0.2x	1x
Single-core Coremark	0.5x	1x
DPDK hash_perf	0.3x	1x
DPDK readwrite_lf_perf	0.3x	1x

CPU scores[1] (higher is better)

[1] Offloading distributed applications onto smartnics using ipipe. SIGCOMM'19

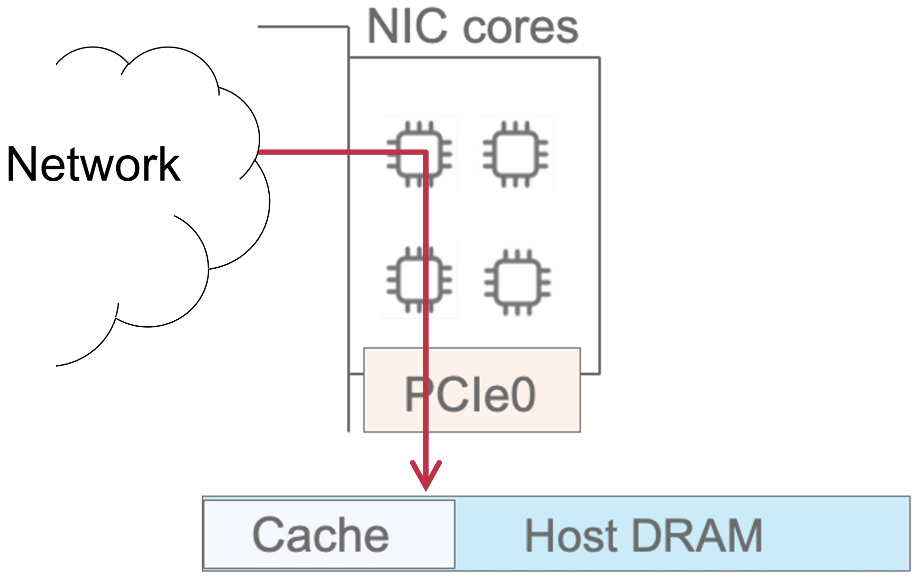
[2] Performance characteristics of the bluefield-2 smartnic, arXiv

[3] A dbms-centric evaluation of bluefield dpus on fast networks. ADMS'22

Our work: the most comprehensive characterization on SmartNIC

An important (and basic) component of NIC: communication, is not well explored

- The **communication paths** of SmartNIC are more complex than other NICs



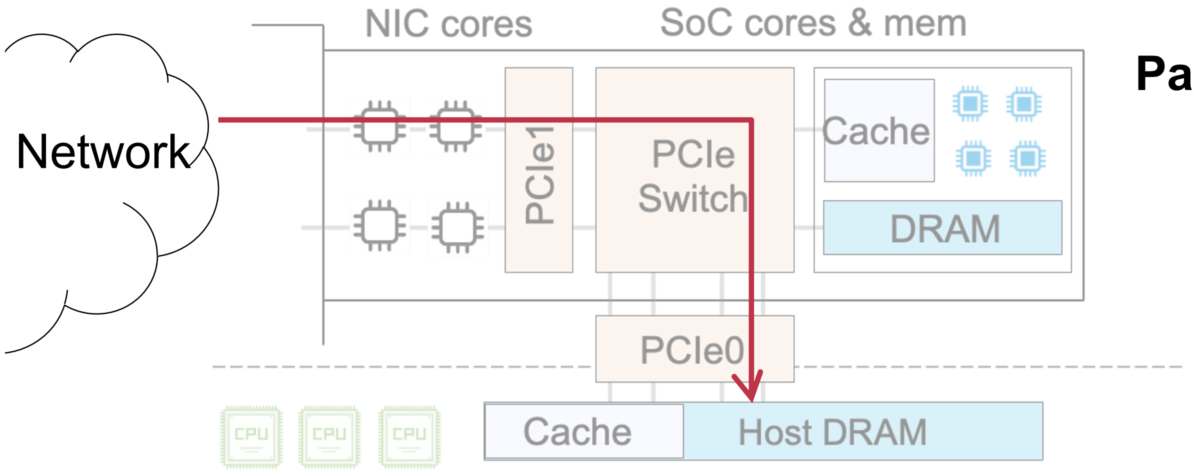
Traditional NICs (RDMA or non-RDMA)

Path #1: Client ⟶ NIC ⟶ Host memory

Our work: the most comprehensive characterization on SmartNIC

An important (and basic) component of NIC: communication, is not well explored

– The **communication paths** of SmartNIC are more complex than other NICs



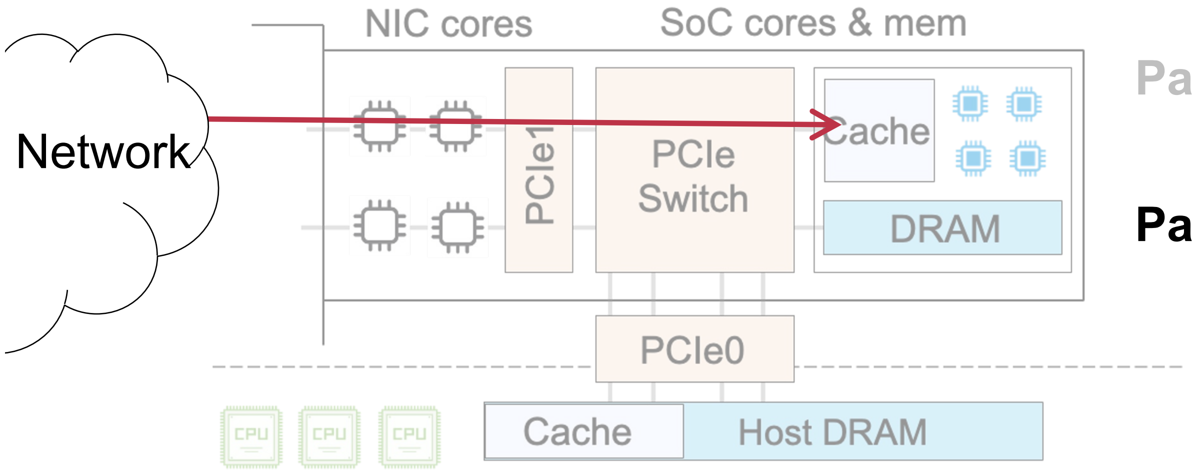
Path #1: Client ⟶ NIC ⟶ Host memory

SmartNIC

Our work: the most comprehensive characterization on SmartNIC

An important (and basic) component of NIC: communication, is not well explored

– The **communication paths** of SmartNIC are more complex than other NICs



SmartNIC

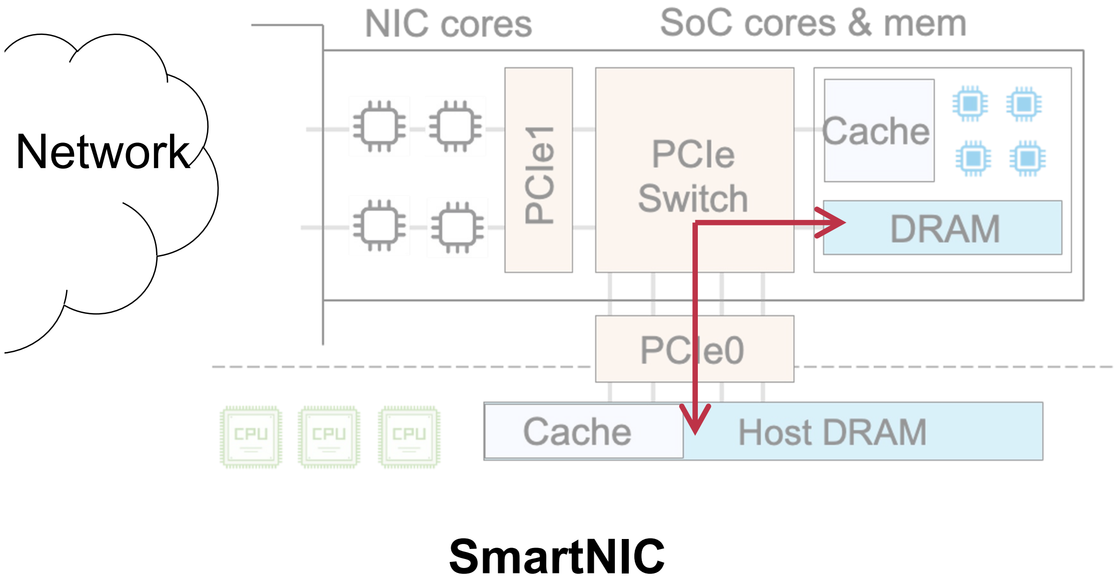
Path #1: Client ⟶ NIC ⟶ Host memory

Path #2: Client ⟶ NIC ⟶ SoC memory

Our work: the most comprehensive characterization on SmartNIC

An important (and basic) component of NIC: communication, is not well explored

- The **communication paths** of SmartNIC are more complex than other NICs



Path #1: Client ⟶ NIC ⟶ Host memory

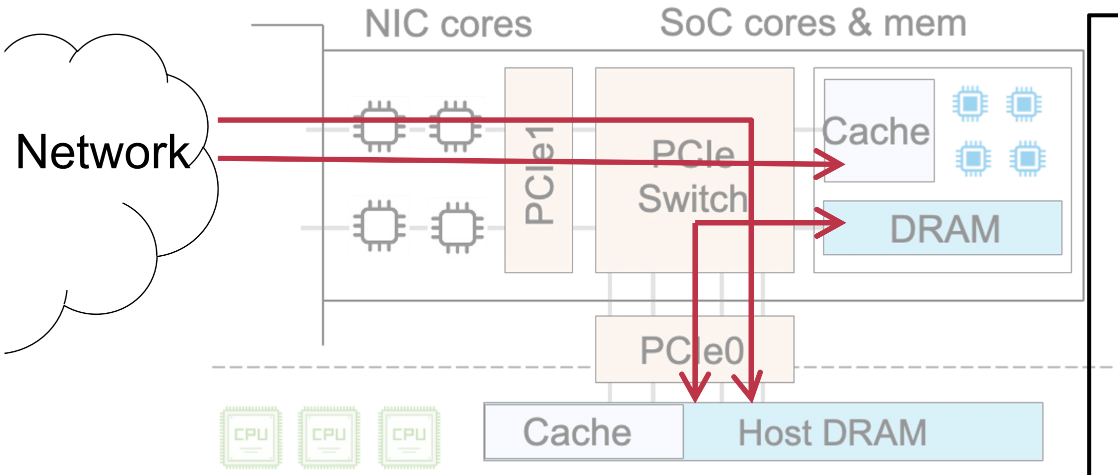
Path #2: Client ⟶ NIC ⟶ SoC memory

Path #3: SoC memory ⟷ host memory

Our work: the most comprehensive characterization on SmartNIC

An important (and basic) component of NIC: communication, is not well explored

– The **communication paths** of SmartNIC are more complex than other NICs



SmartNIC

Path #1: Client ⟶ NIC ⟶ Host memory

Path #2: Client ⟶ NIC ⟶ SoC memory

Path #3: SoC memory ⟷ host memory

Performance characterization on all them!

What do we characterize ?

1. SmartNIC hardware implication to the communication performance

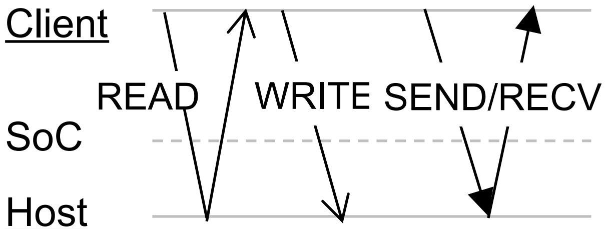
2. Design guideline on building systems with SmartNICs

Finding 1. SmartNIC < RNIC for path #1

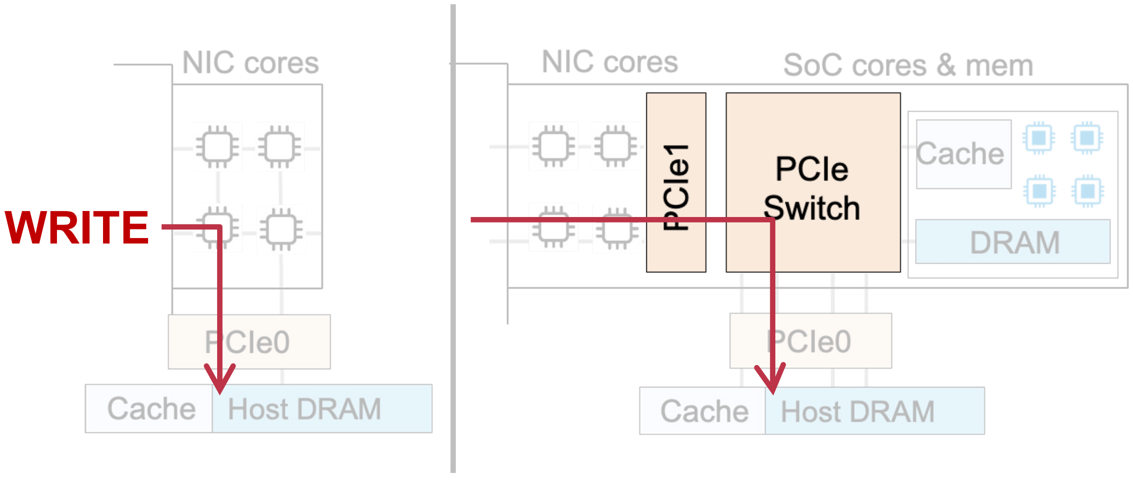
The path #1 is long on the SmartNIC

- Due to the intervention of PCIe switch
- The one-way switch pass latency (300ns) is non-trivial for microsecond-scale computing

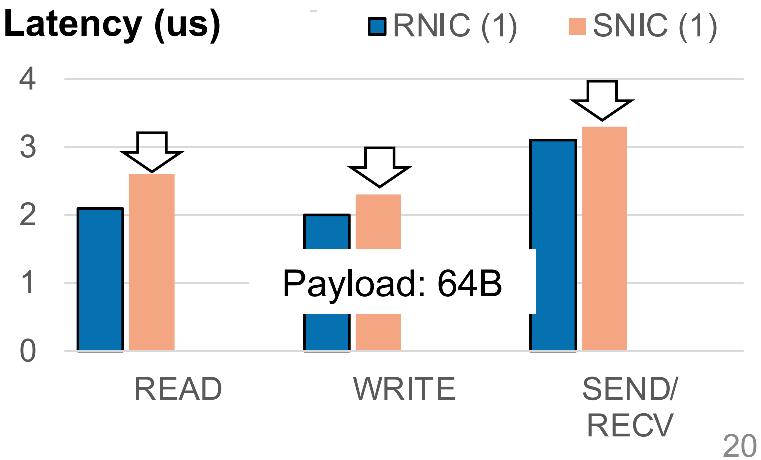
Primitives evaluated



Evaluation setup:
ConnectX-6 (RNIC) vs. Bluefield -2 (SmartNIC)
Both NICs use the same NIC cores



RNIC vs. SmartNIC

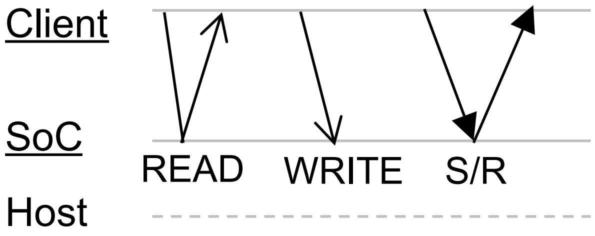


Finding 2. Path #2 is fast except for S/R

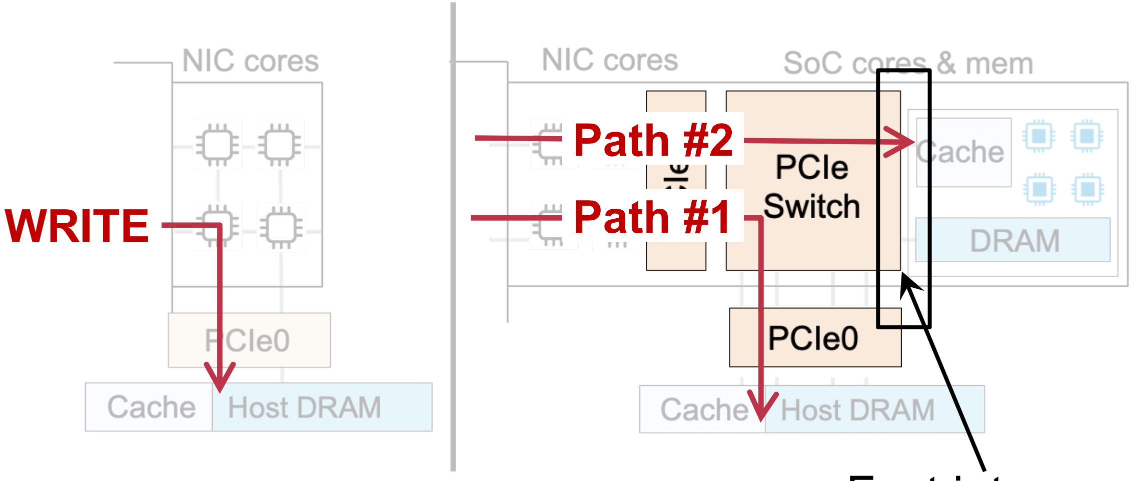
Communication with the SoC is faster except for the SEND/RECV (S/R)

- Due to the reduced PCIe pass (i.e., PCIe0)
- SEND/RECV is bottlenecked by the SoC cores

Primitives evaluated

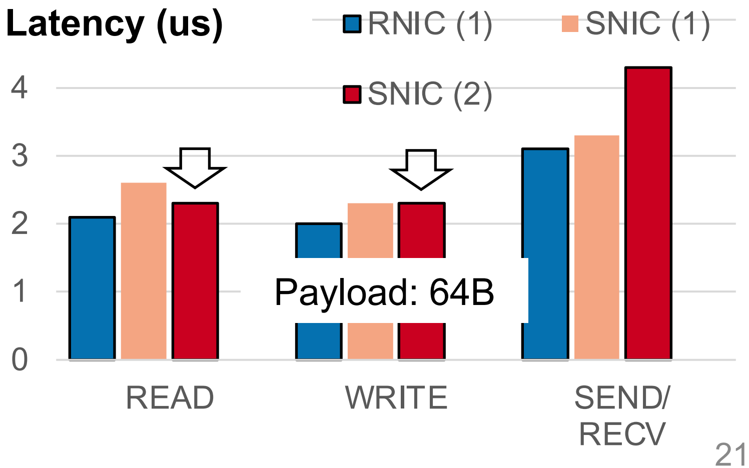


Evaluation setup:
ConnectX-6 (RNIC) vs. Bluefield -2 (SmartNIC)
Both NICs use the same NIC cores

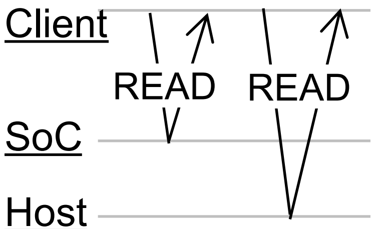


RNIC vs. SmartNIC

Fast interconnect other than PCIe



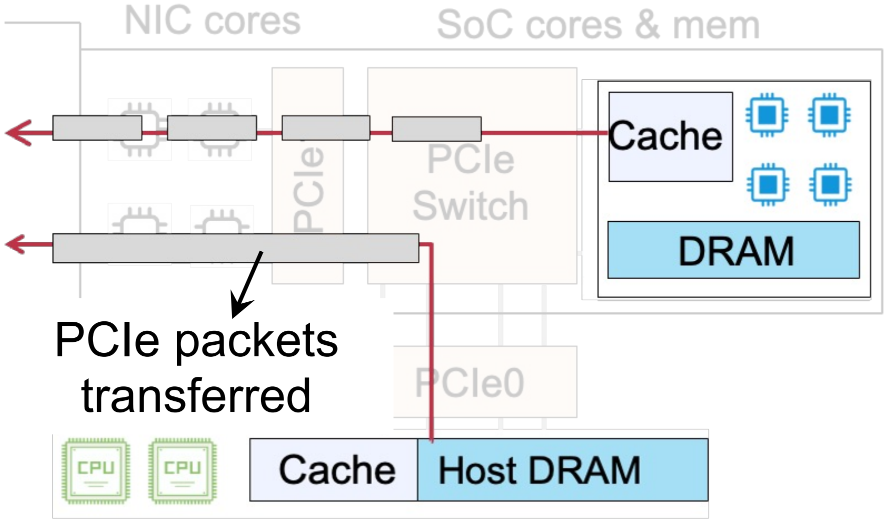
Finding 3. Anomalies exist paths involving SoC



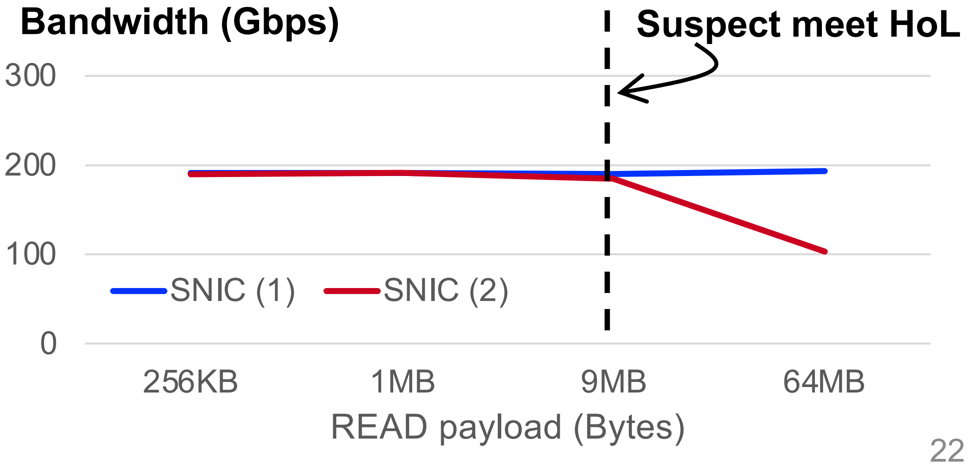
Example. Degraded bandwidth (for READ) with large data transfer

- Observation: SoC supports a smaller PCIe MTU than host
- Result: more PCIe packets processed, may cause HoL

Advice: proactively segmented large READ



	Host	SoC
PCIe MTU	512B	128B



Finding 4. Path #3 has trade-offs

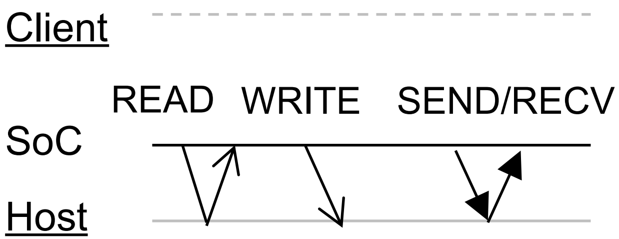
Many alternatives to implement Path #3

- The simplest (& easiest to use one): RDMA

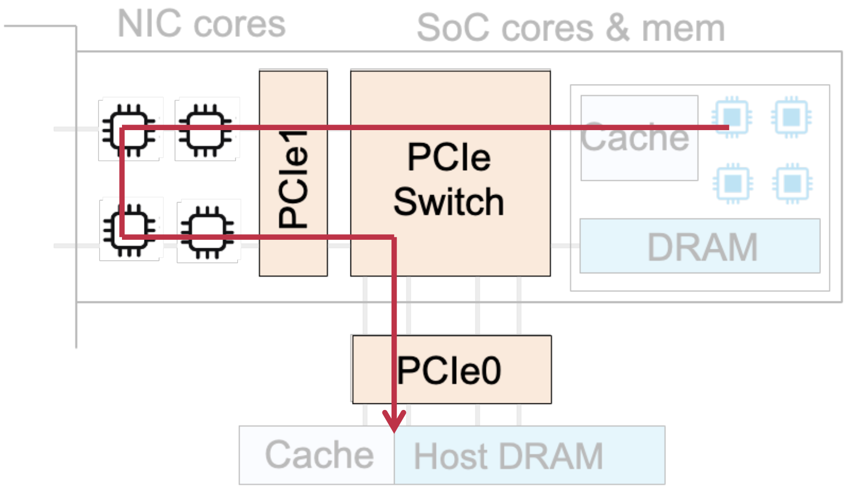
Yet, RDMA needs to pass RNICs & PCIes

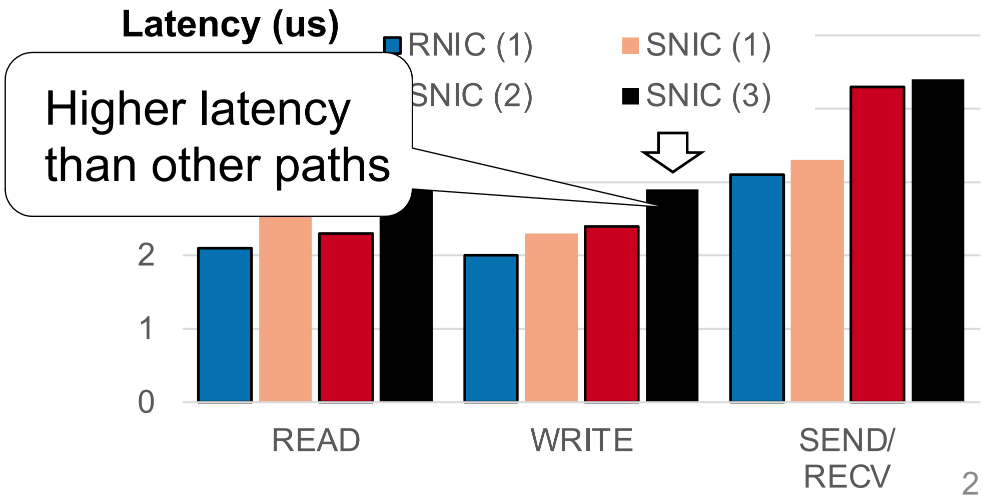
- For networking support

Primitives evaluated



Evaluation setup:
ConnectX-6 (RNIC) vs. Bluefield -2 (SmartNIC)
Both NICs use the same NIC cores





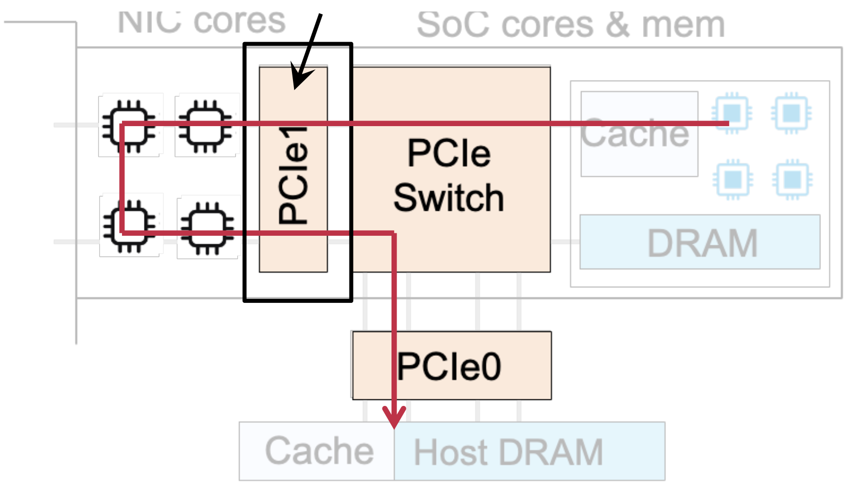
Higher latency than other paths

Finding 4. Path #3 has trade-offs

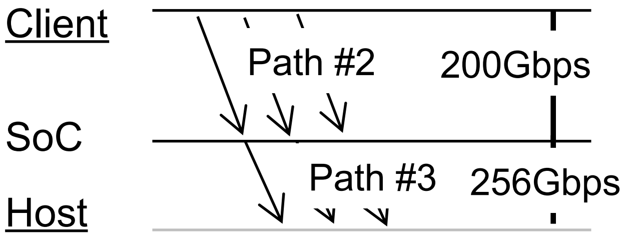
RDMA, though simple, has two problems for Path #3

- High latency due to additionally passes hardware units
- Bandwidth interference to the others

RDMA of path #3 overuses the PCIe bandwidth



Client → SoC → Host



Case study: file replication in LineFS [SOSP'21]

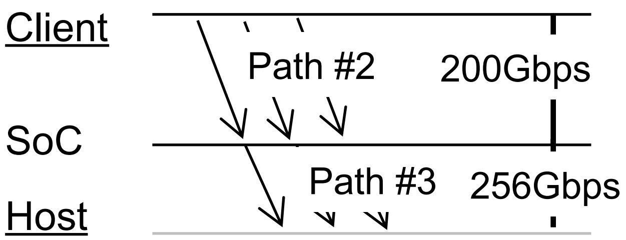
What is the performance of path#2 + path#3?

Finding 4. Path #3 has trade-offs

RDMA, though simple, has two problems for Path #3

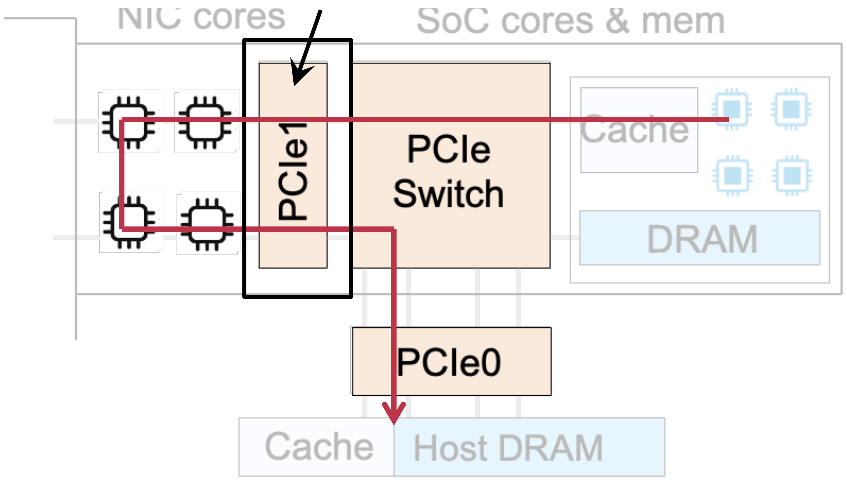
- High latency due to additionally passes hardware units
- Bandwidth interference to the others

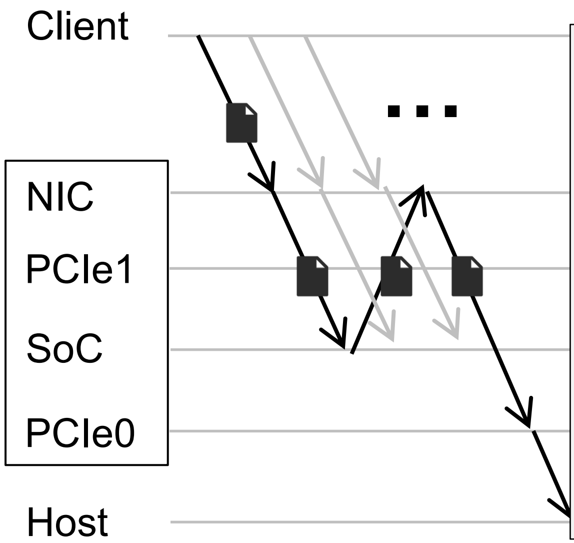
Client → SoC → Host



Case study: file replication in LineFS [SOSP'21]

RDMA of path #3 overuses the PCIe bandwidth





PCIe 1 & 2 bandwidth
Bi-directional: **256Gbps**

Data passes PCIe1 twice
1. NIC → SoC
2. SoC → NIC

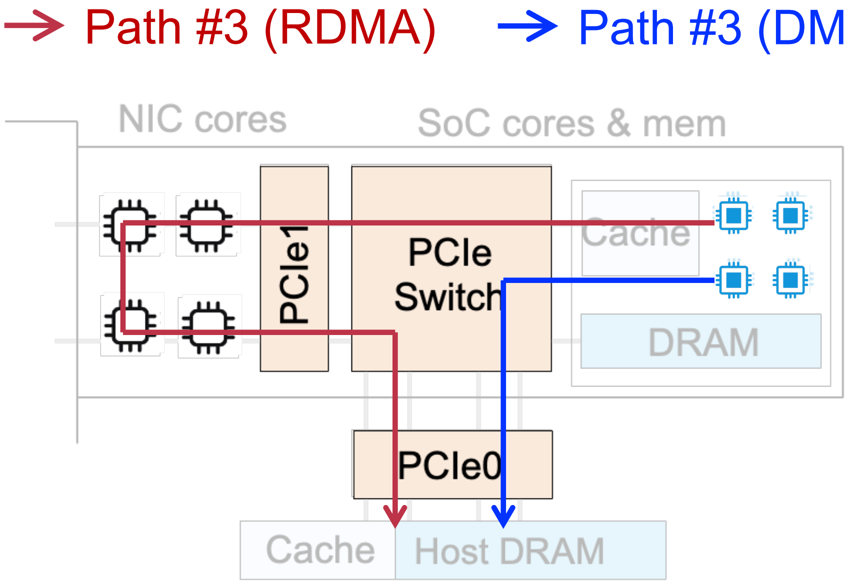
Path#2 + Path#3
Peak at **128Gbps**!

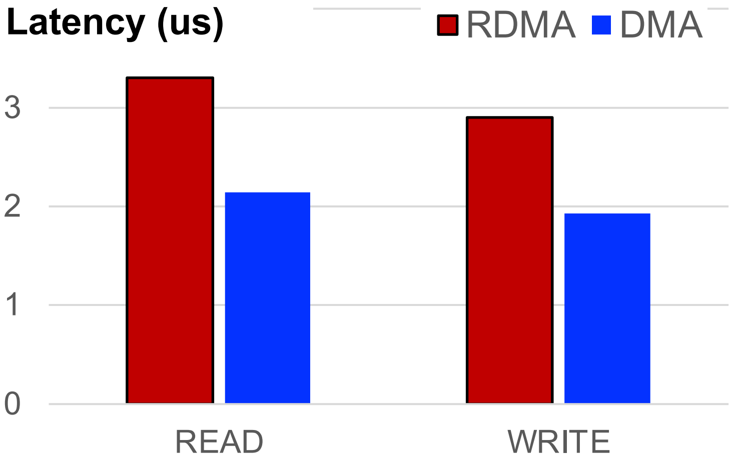
Finding 4. Path #3 has trade-offs

DMA always better?

DMA: another alternative for path #3

- Unlike RDMA, the SoC has a DMA engine for path #3 (i.e., DOCA DMA)
- DMA bypasses PCIe for communication between SoC and host



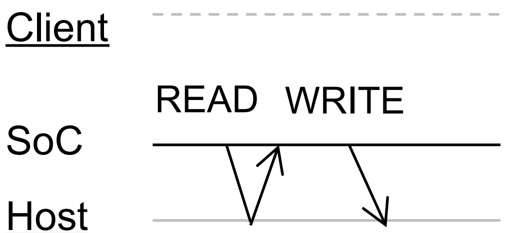


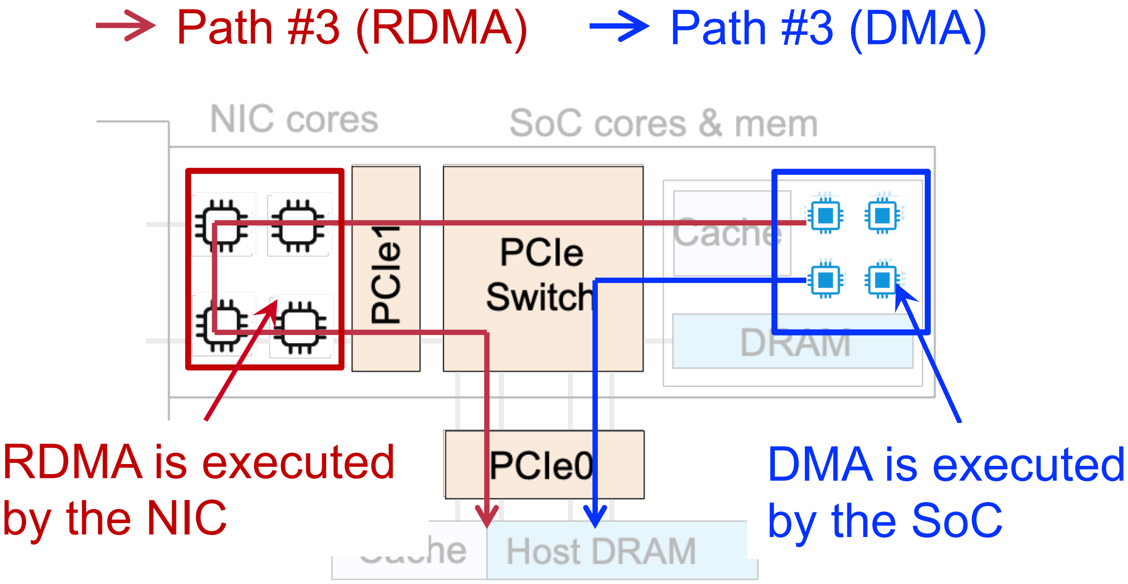
Finding 4. Path #3 has trade-offs

The engine capabilities of RDMA and DMA is different

- RDMA engine (NIC) is more powerful than DMA engine (SoC)
- So RDMA is faster for transferring small data

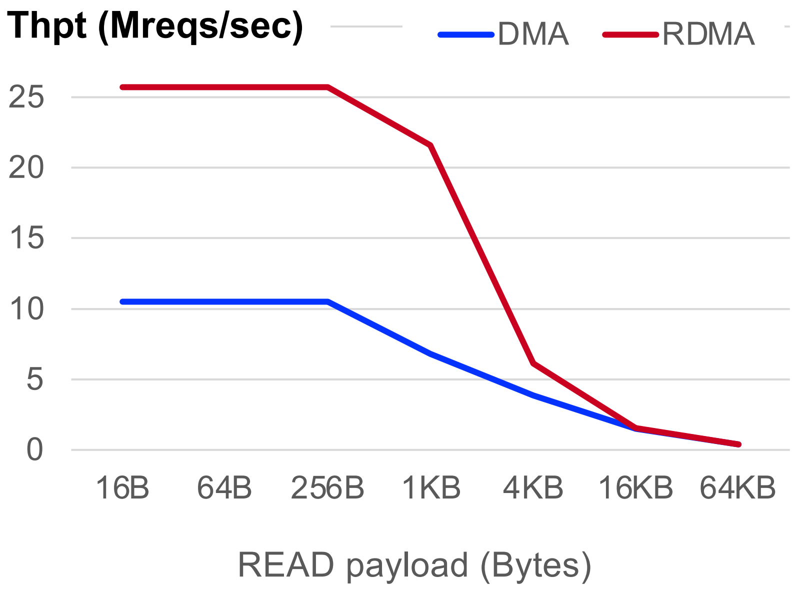
Primitives evaluated





RDMA is executed by the NIC

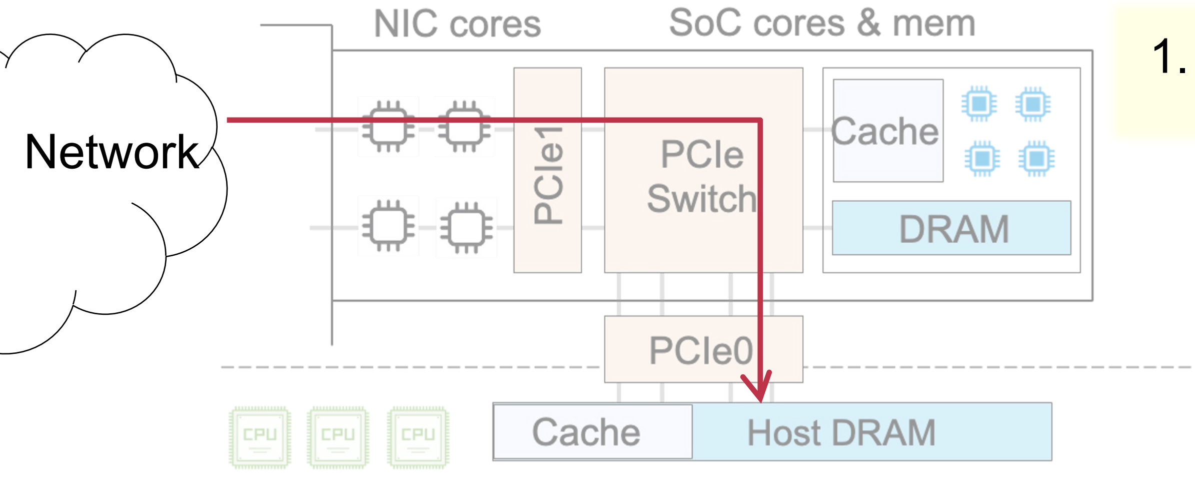
DMA is executed by the SoC



Key takeaway of the above findings

Each communication path of SmartNIC is not perfect

– Inferior performance or performance anomalies needs to take care



SmartNIC

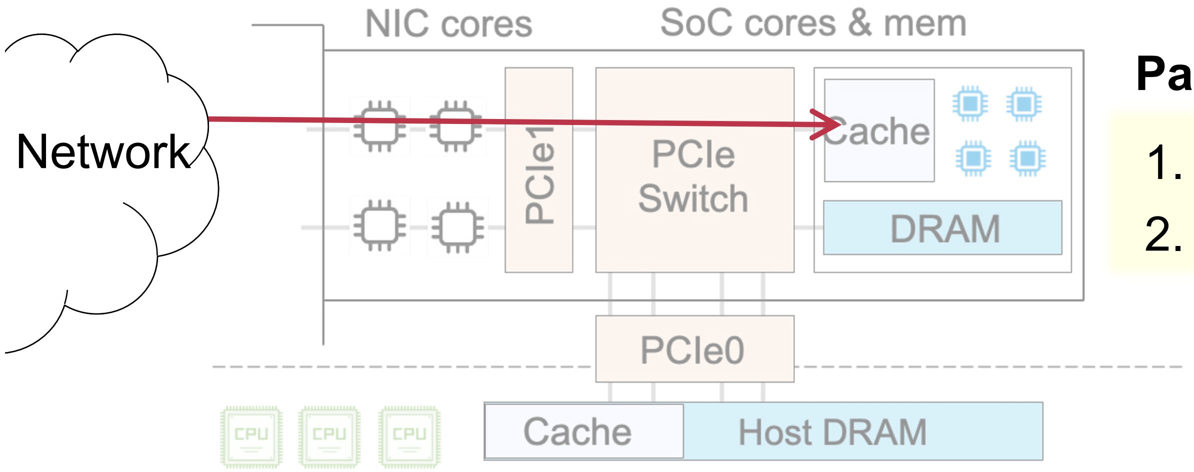
Path #1: Client ⟶ NIC ⟶ Host memory

1. Inferior performance vs. RNIC

Key takeaway of the above findings

Each communication path of SmartNIC is not perfect

– Inferior performance or performance anomalies needs to take care



SmartNIC

Path #1: Client ⟶ NIC ⟶ Host memory

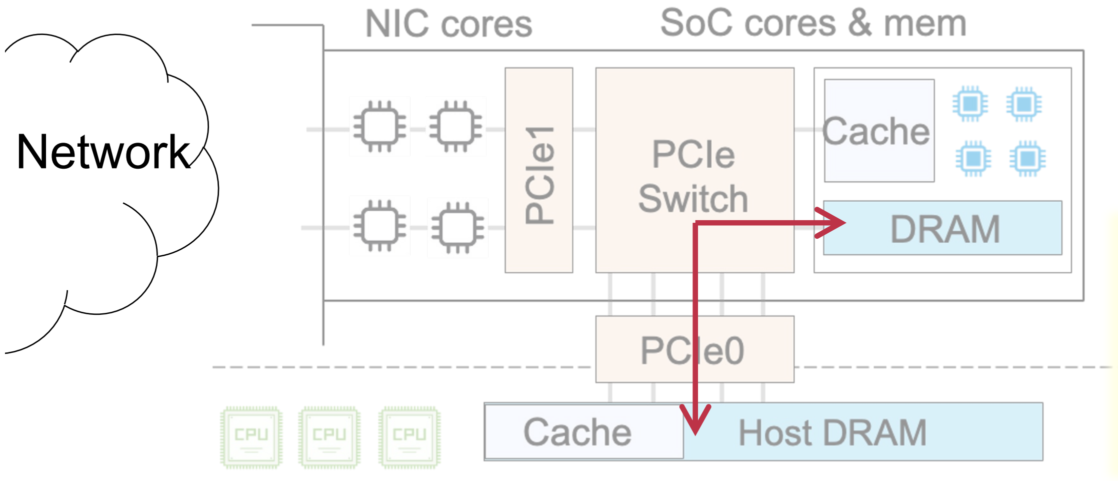
Path #2: Client ⟶ NIC ⟶ SoC memory

1. Faster access
2. Anomalies, e.g., more PCIe packets

Key takeaway of the above findings

Each communication path of SmartNIC is not perfect

– Inferior performance or performance anomalies needs to take care



SmartNIC

Path #1: Client ⟶ NIC ⟶ Host memory

Path #2: Client ⟶ NIC ⟶ SoC memory

Path #3: SoC memory ⟵⟶ host memory

1. RDMA: poor PCIe utilization + high latency
2. DMA: poor throughput

Back to our key-value store example

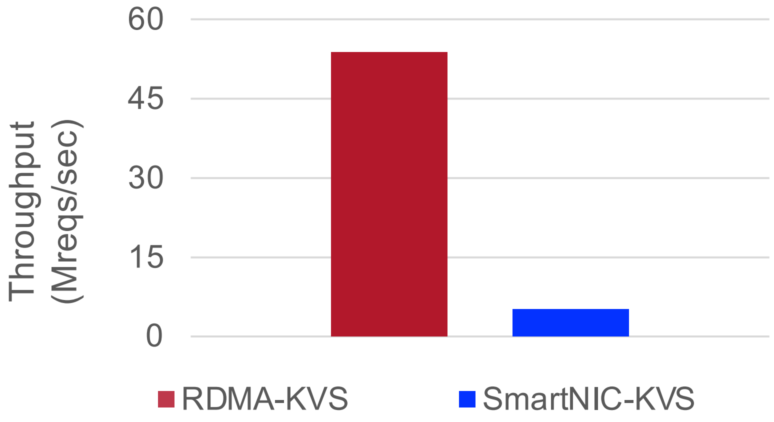
How to help?

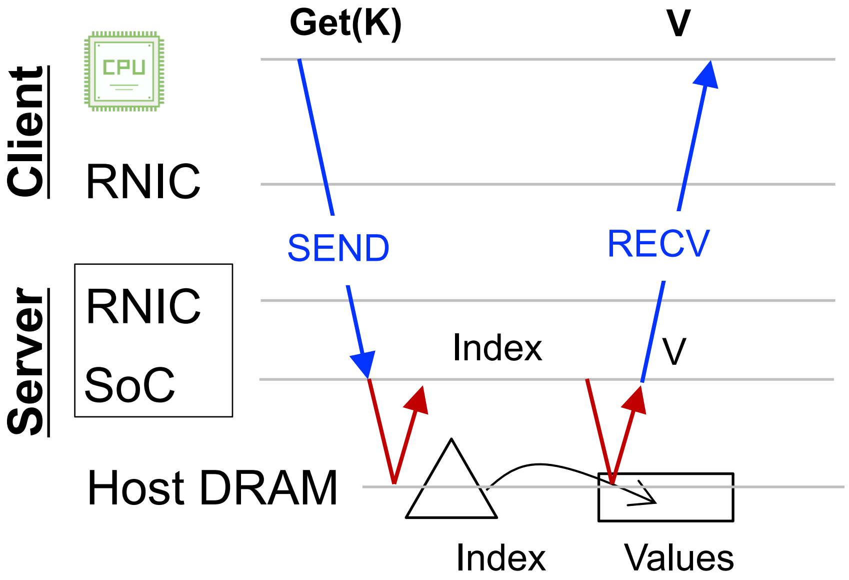
Our characterization explains why naïve KVS on SmartNIC is slow

1. SoC has wimpy cores (known)
2. Path #3 is slow in terms of latency (RDMA) and throughput (DMA)

NIC cores are under-utilized on the SmartNIC

Which are much faster than the SoC





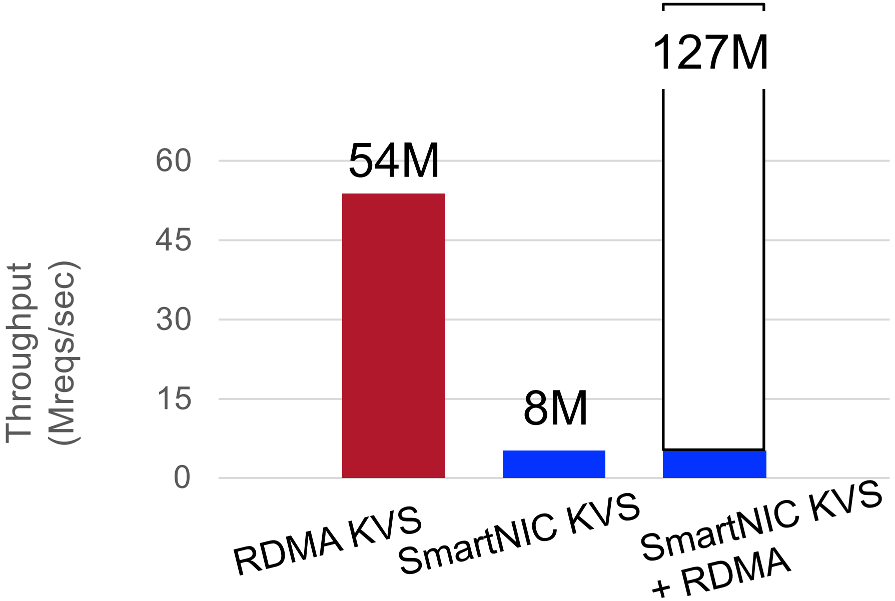
Observation: a single path is not optimal, but concurrency can help!

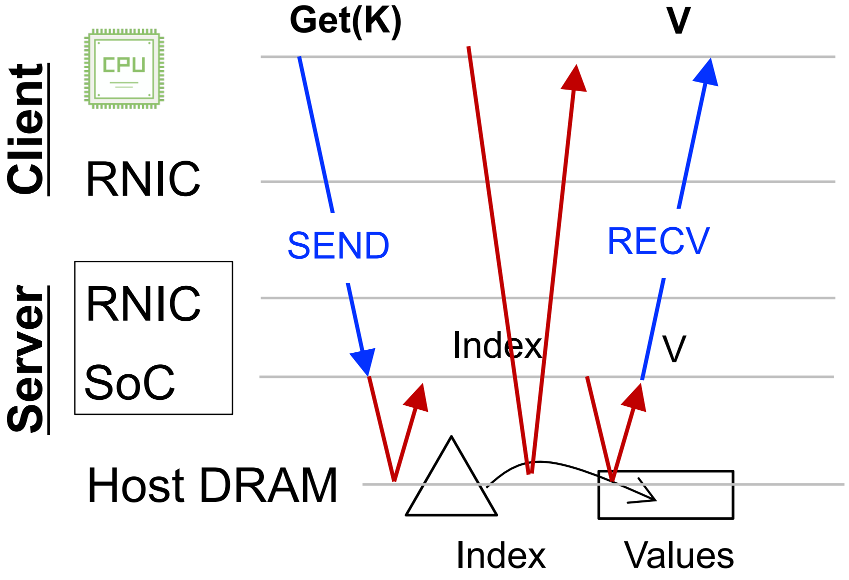
A single alternative does not utilize the full power of SmartNIC

– E.g., Bottlenecked by slow SoC–DMA and SoC in our naïve KVS design

Observation: concurrently utilize the SmartNIC power

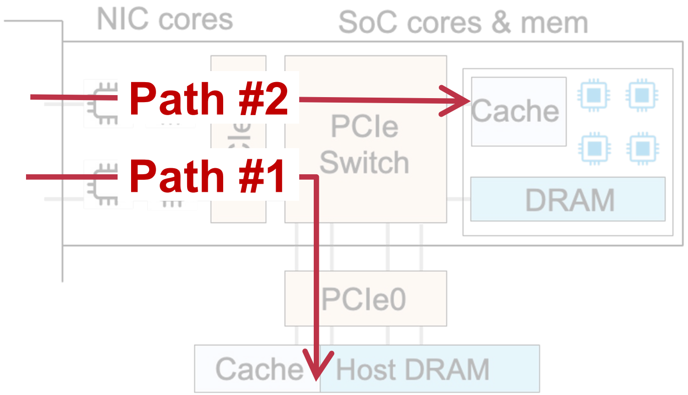
– E.g., we can utilize the unused RNIC!





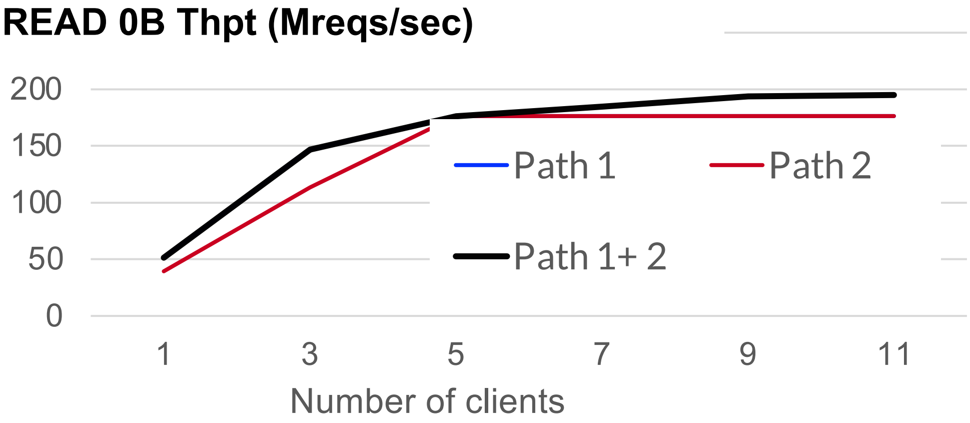
More findings: concurrent path can better utilize SmartNIC

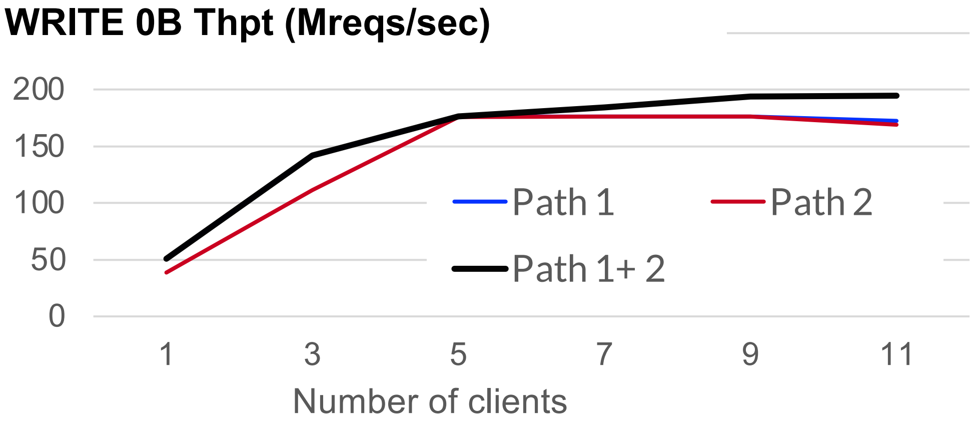
Concurrent = some clients issues Path #1 ops, other issues Path #2



Concurrent usage of Path #1 + Path #2

– Observation: SmartNIC seems to reserve NIC cores for different paths





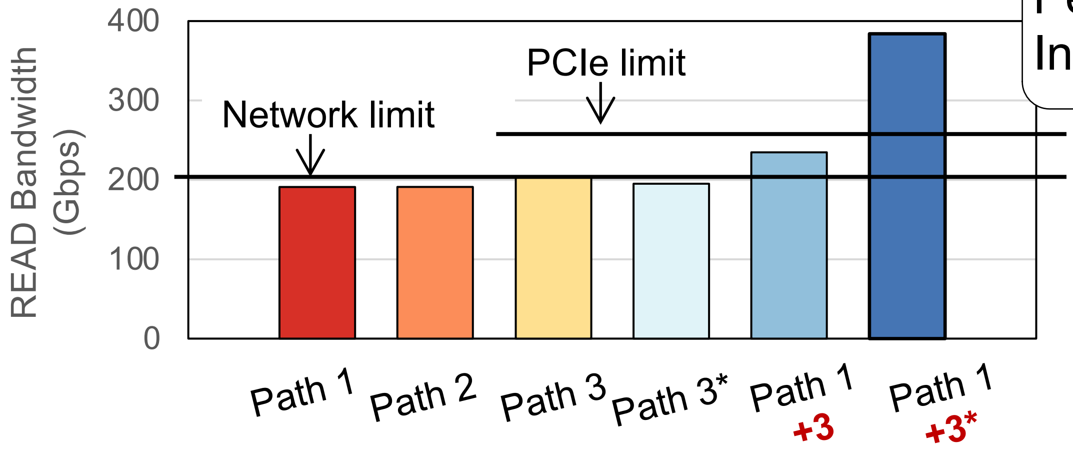
More findings: concurrent path can better utilize SmartNIC

Concurrent usage of Path #1 + Path #3

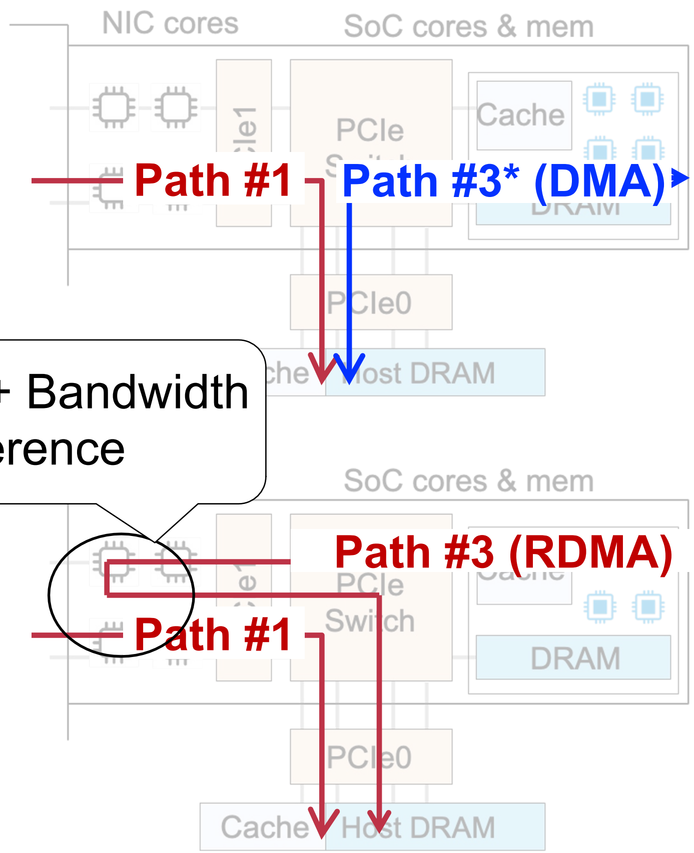
- Typically, can achieve a higher bandwidth

But, we should take care of interference !

- RDMA is not a good primitive for path #3



Perf. + Bandwidth Interference

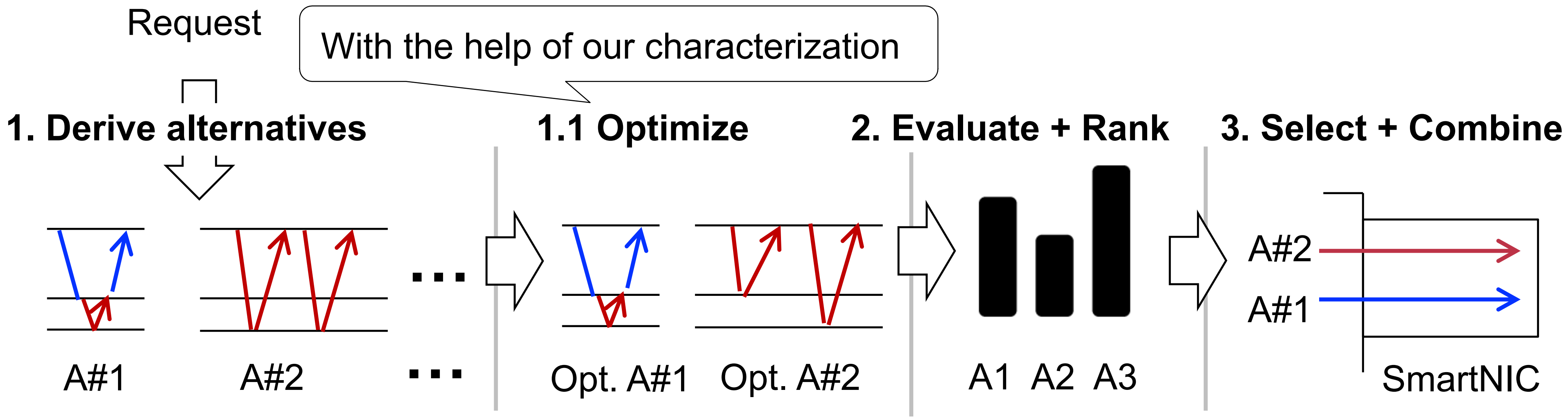


A guideline on building systems with SnartNIC

Recap the key takeaways from our characterization

- Single path: Inferior performance or performance anomalies
- Concurrent paths: better performance

Our suggested guideline when given a user networked request (e.g., KVS get())

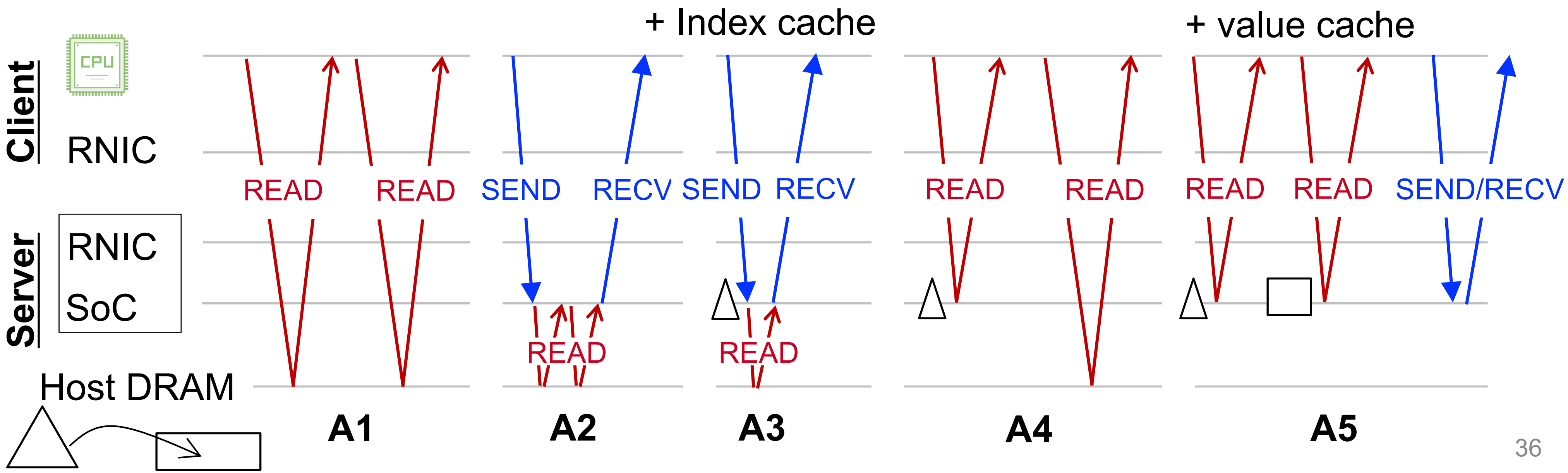


Distributed key-value storage Get() revisit

System requirements

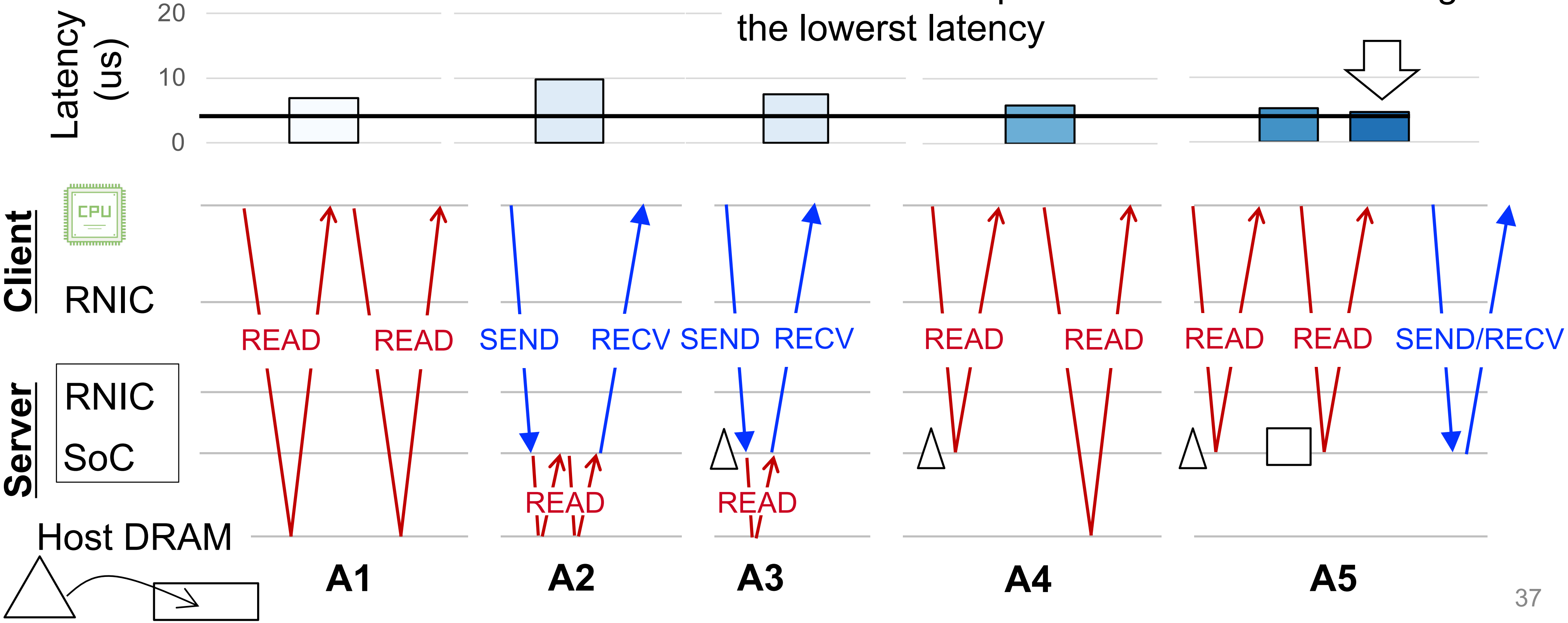
- Low host CPU usage, low latency & high throughput
- Low host CPU utilization

1 + 2. Design alternatives (A1—A5) & optimize! :



Evaluate different alternatives: throughput

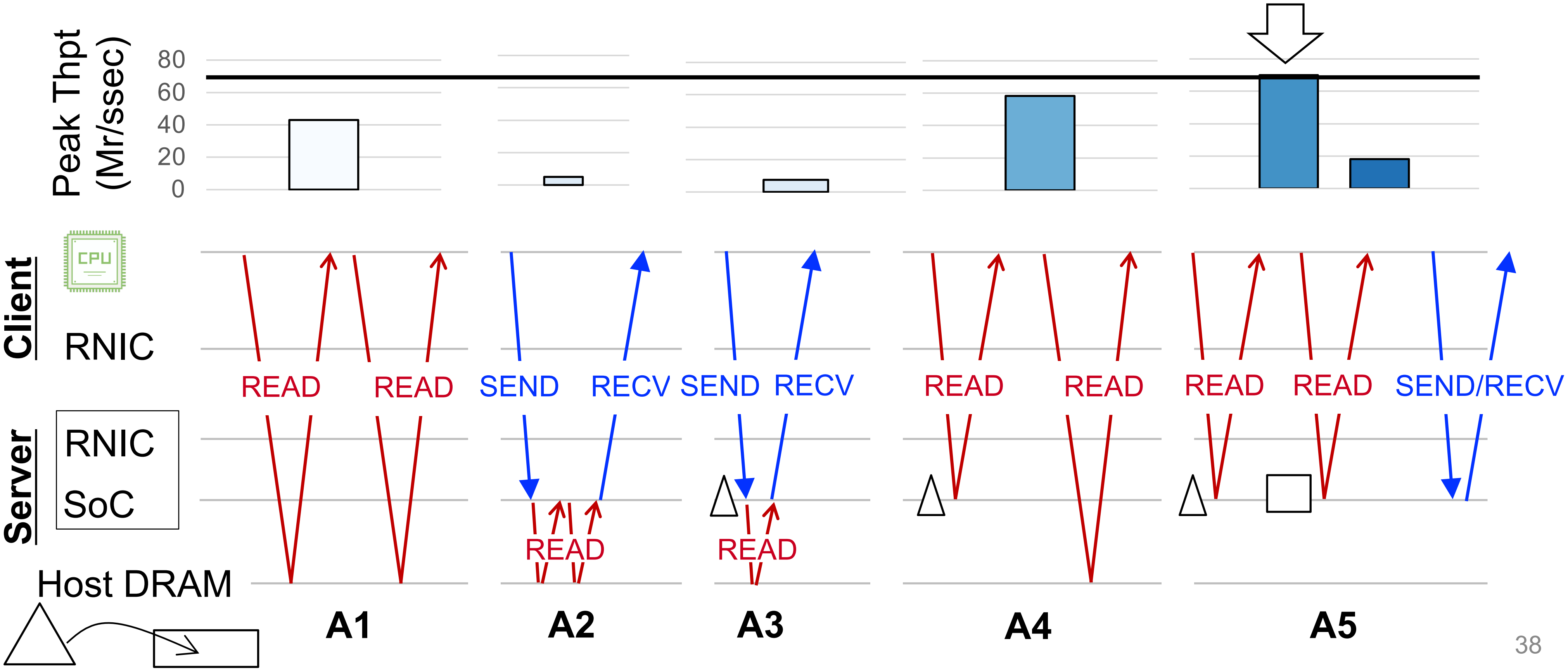
The goal: low latency (when there is not so much client)



Offload KVS request on the SoC w/ caching has the lowerst latency

Evaluate different alternatives: throughput

The goal: high throughput (for a single SmartNIC-powered key-value store)



Rank, select and then combine

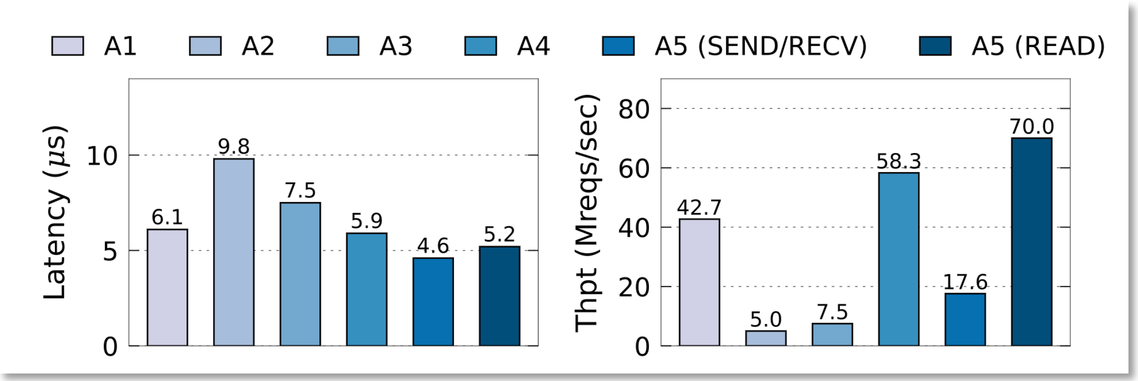
None of the approaches achieve both low latency & high throughput

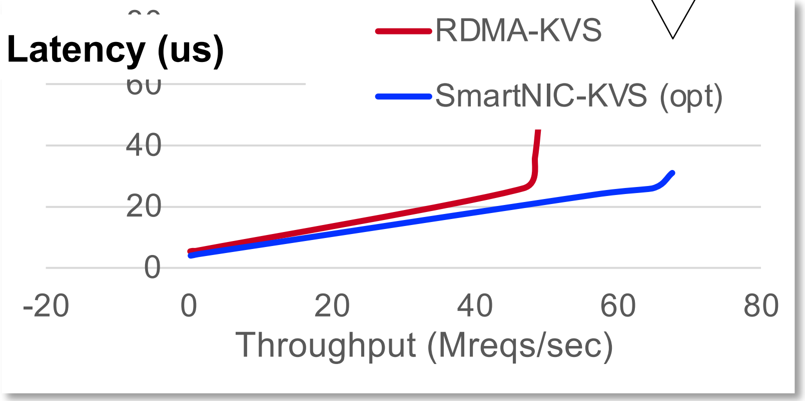
- A5(SEND/RECV) has the lowest latency, while A5(RDMA) has the highest throughput
- Note that A5 is not always possible due to memory constraint of SoC

Whenever possible, choose A5 (SEND/RECV) for the lowest latency

- If the SoC has been saturated, switch to A4 & A5 (RDMA)

Combine A4 + A5





Evaluation setup: YCSB-C 100% Get( )

More results & case studies in our paper

More findings and advices from our characterization

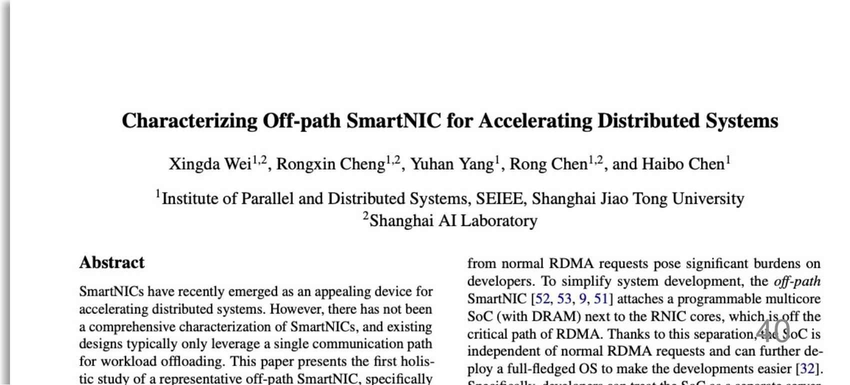
- e.g., Different communication path may access the cache or memory differently

More characterization on the concurrent combination of different paths

- A combination of different paths can yield better performance on microbenchmarks

More case studies

- How we improved LineFS [SOSP’21] by 1.3X with a combination of improved alternative design & optimization on each alternative



Characterizing Off-path SmartNIC for Accelerating Distributed Systems

Xingda Wei[1,2], Rongxin Cheng[1,2], Yuhan Yang[1], Rong Chen[1,2], and Haibo Chen[1]

[1]Institute of Parallel and Distributed Systems, SEIEE, Shanghai Jiao Tong University
[2]Shanghai AI Laboratory

Abstract

SmartNICs have recently emerged as an appealing device for accelerating distributed systems. However, there has not been a comprehensive characterization of SmartNICs, and existing designs typically only leverage a single communication path for workload offloading. This paper presents the first holistic study of a representative off-path SmartNIC, specifically

from normal RDMA requests pose significant burdens on developers. To simplify system development, the *off-path* SmartNIC [52, 53, 9, 51] attaches a programmable multicore SoC (with DRAM) next to the RNIC cores, which is off the critical path of RDMA. Thanks to this separation, the SoC is independent of normal RDMA requests and can further deploy a full-fledged OS to make the developments easier [32].

Conclusion

Before using the SmartNIC, we must first characterize it!

- More complicated than traditional network card
- Many design details need to take care

This work: a comprehensive study on off-path SmartNIC (i.e., Bluefield-2)

- Reveal anomalies (& solutions to them) + guidelines on how to better utilize it
- Our methodology may also apply to other NICs

Thanks and Q & A