

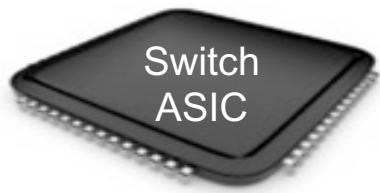
PISA-Based Application Acceleration for IO-Intensive Workloads

Xin Jin

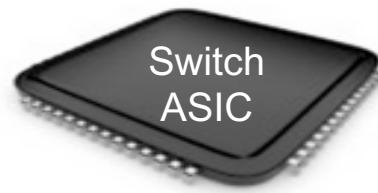


Joint work with Xiaozhou Li, Haoyu Zhang, Nate Foster, Jeongkeun Lee, Robert Soulé,
Changhoon Kim, and Ion Stoica

The revolution in networking



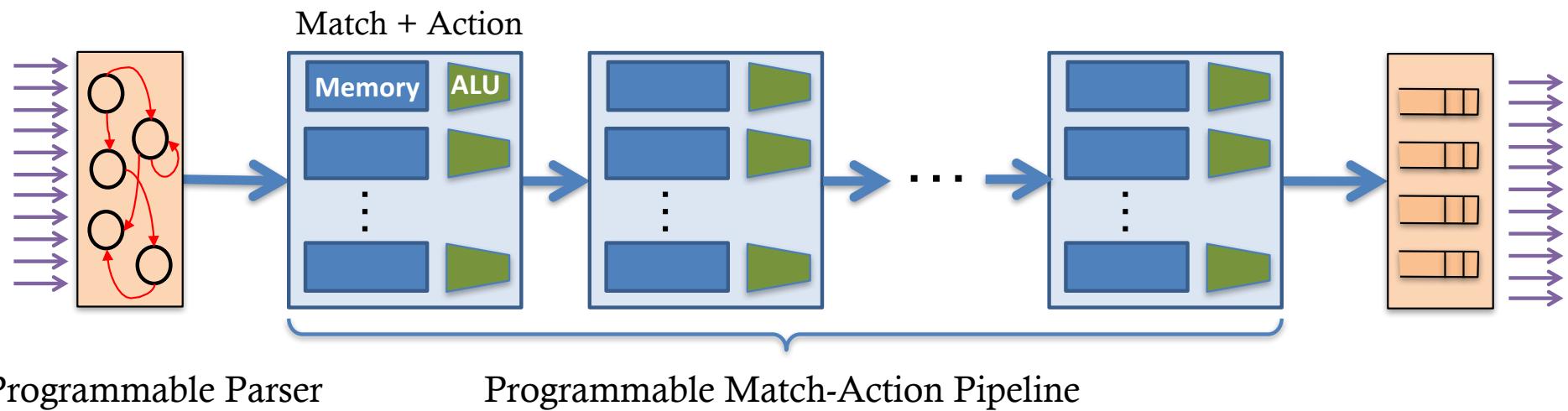
Fixed-function switch



Programmable switch

PISA: Protocol Independent Switch Architecture

- Programmable Parser
 - Convert packet data into metadata
- Programmable Match-Action Pipeline
 - Operate on metadata and update memory state



Programmable switch data planes enable many innovations

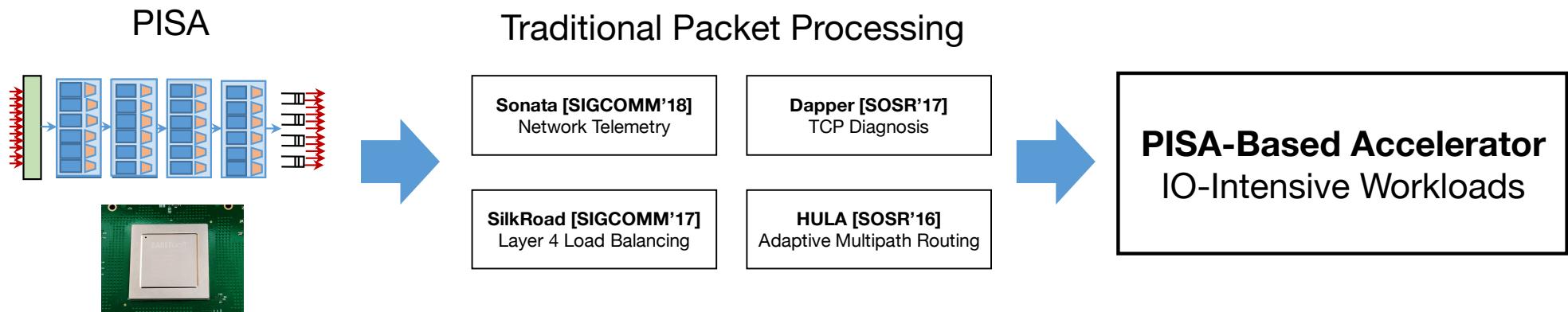
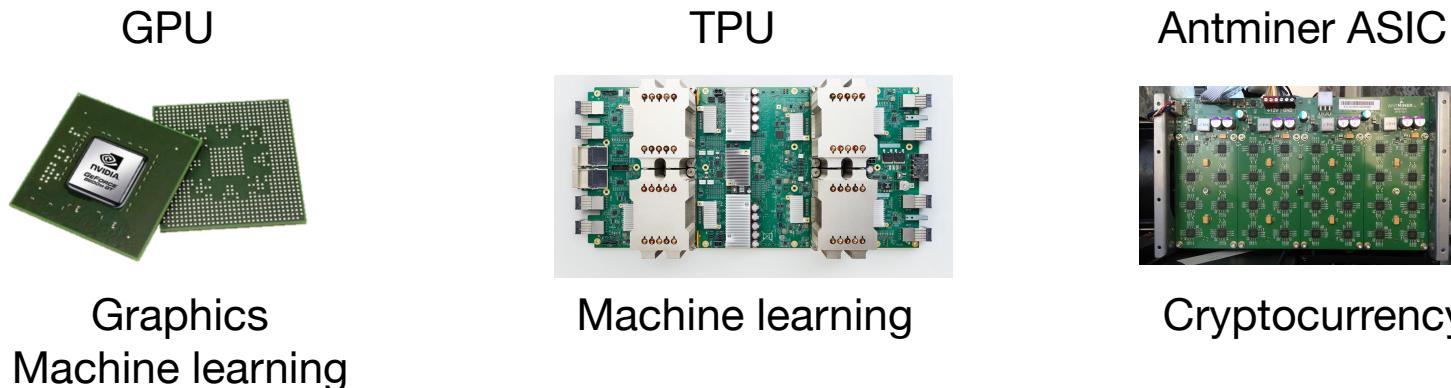
Sonata [SIGCOMM'18]
Network Telemetry

Dapper [SOSR'17]
TCP Diagnosis

SilkRoad [SIGCOMM'17]
Layer 4 Load Balancing

HULA [SOSR'16]
Adaptive Multipath Routing

The ending of the Moore's Law, and the rise of domain specific processors...



PISA switches as domain specific accelerators for IO-intensive workloads

- **NetCache** [SOSP'17]: balancing key-value stores with PISA-based caching
- **NetChain** [NSDI'18, best paper award]: fast coordination with PISA-based chain replication

Joint work with Xiaozhou Li, Haoyu Zhang, Nate Foster, Jeongkeun Lee, Robert Soulé, Changhoon Kim, and Ion Stoica

NetCache is a **rack-scale key-value store** that leverages
PISA-based caching to achieve

billions QPS throughput

&

~10 μ s latency

even under

highly-skewed

&

rapidly-changing

workloads.

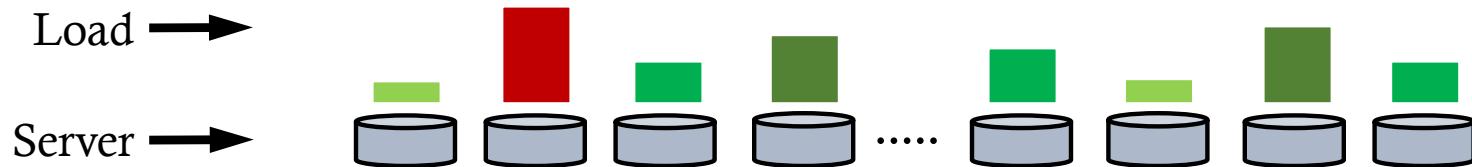
Goal: fast and cost-efficient rack-scale key-value storage

- Store, retrieve, manage key-value objects
 - Critical building block for large-scale cloud services
- Need to **meet aggressive latency and throughput objectives efficiently**
- **Target workloads**
 - Small objects
 - Read intensive
 - **Highly skewed and dynamic key popularity**

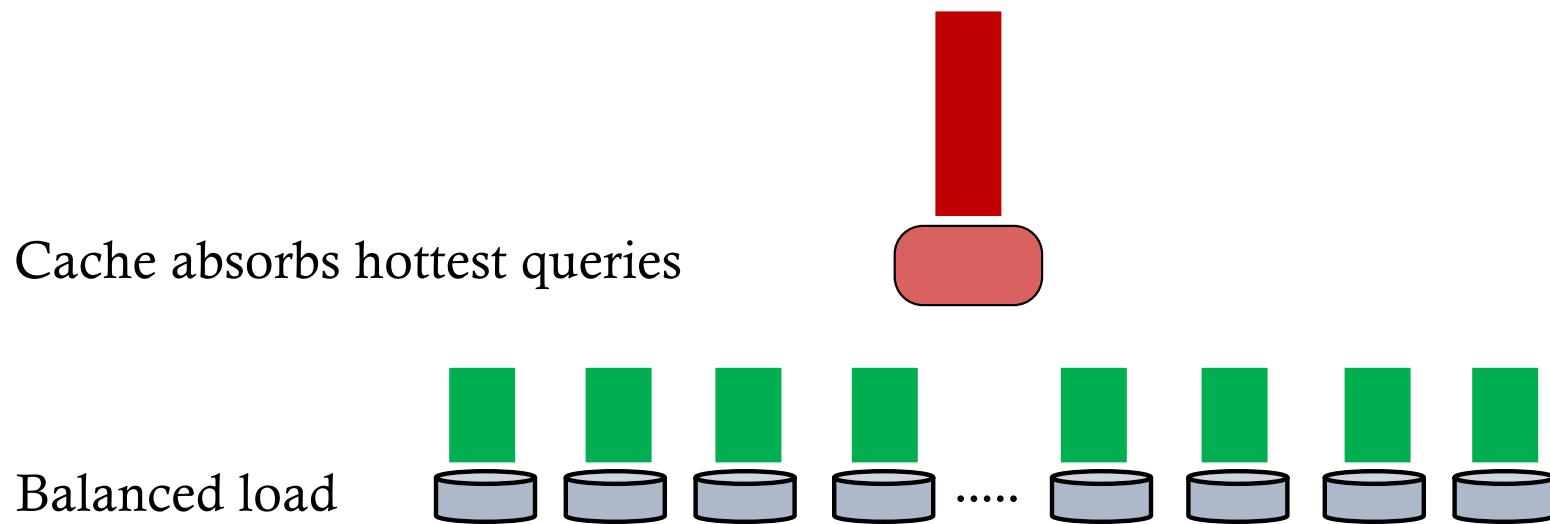


Key challenge: highly-skewed and rapidly-changing workloads

low throughput & high tail latency



Opportunity: fast, small cache for load balancing

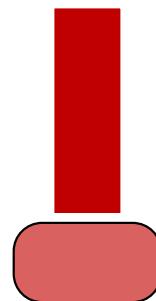


Opportunity: fast, small cache for load balancing

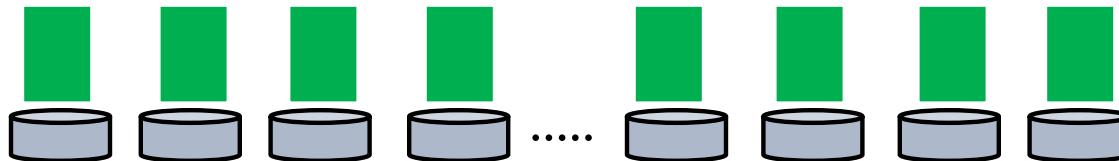
[B. Fan et al. **SoCC'11**, X. Li et al. **NSDI'16**]

Cache $O(N \log N)$ hottest items

E.g., 10,000 hot objects



N: # of servers

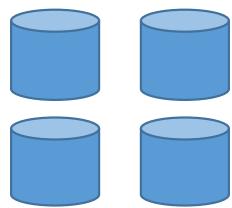


E.g., 100 backends with 100 billions items

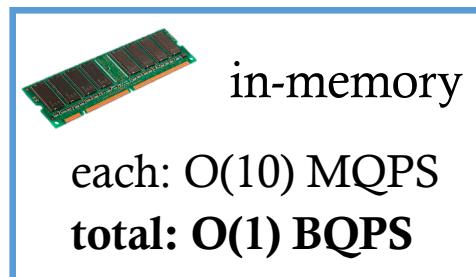
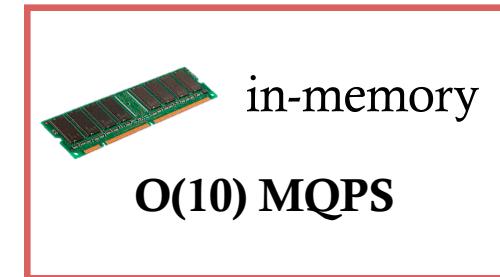
Requirement: cache throughput \geq backend aggregate throughput

NetCache: towards billions QPS key-value storage rack

Cache needs to provide the **aggregate** throughput of the storage layer



cache →

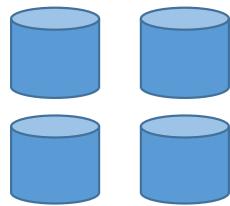


cache →



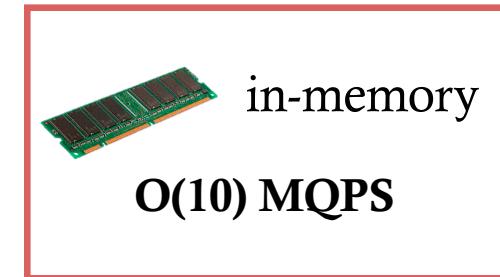
NetCache: towards billions QPS key-value storage rack

Cache needs to provide the **aggregate** throughput of the storage layer

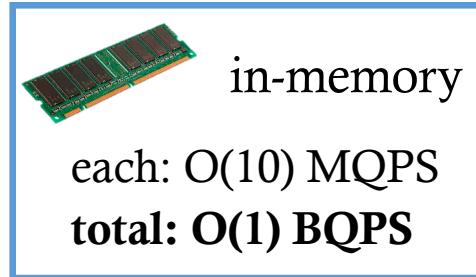


storage layer

cache →



cache layer



cache →



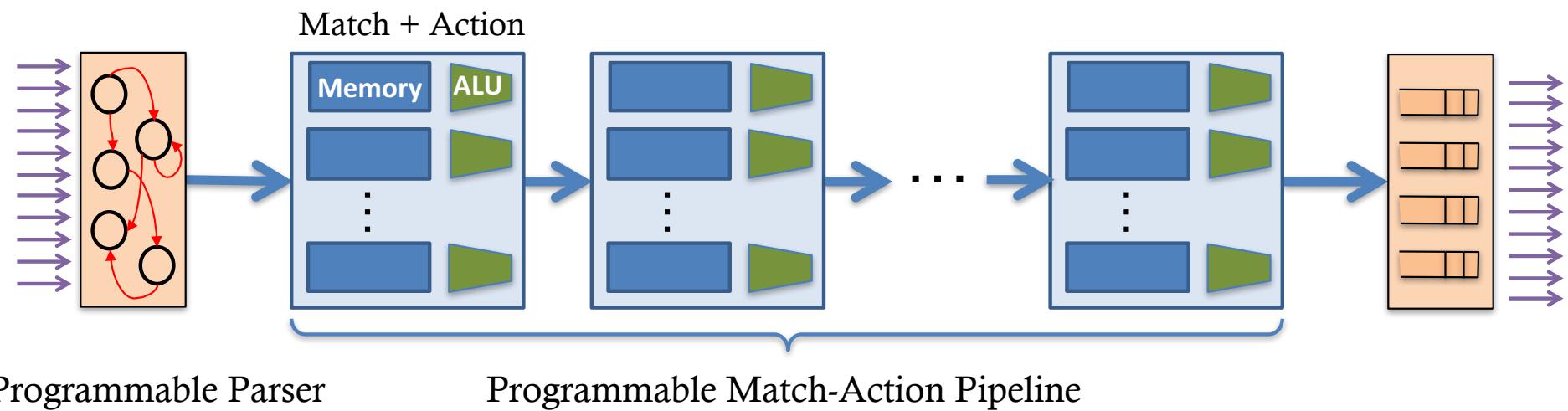
Small on-chip memory?
Only cache **$O(N \log N)$ small** items

Key-value caching in network ASIC at line rate ?!

- How to identify application-level packet fields ?
- How to store and serve variable-length data ?
- How to efficiently keep the cache up-to-date ?

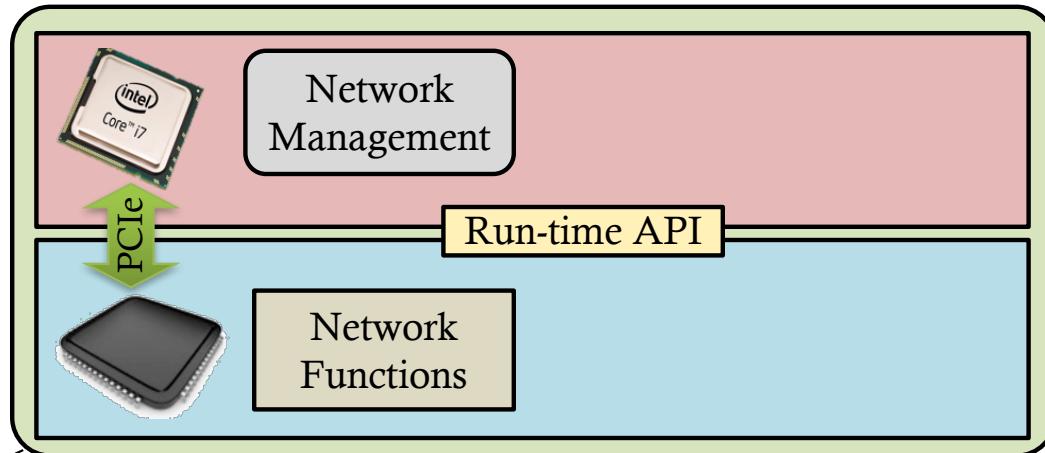
PISA: Protocol Independent Switch Architecture

- Programmable Parser
 - Parse custom key-value fields in the packet
- Programmable Match-Action Pipeline
 - Read and update key-value data
 - Provide query statistics for cache update

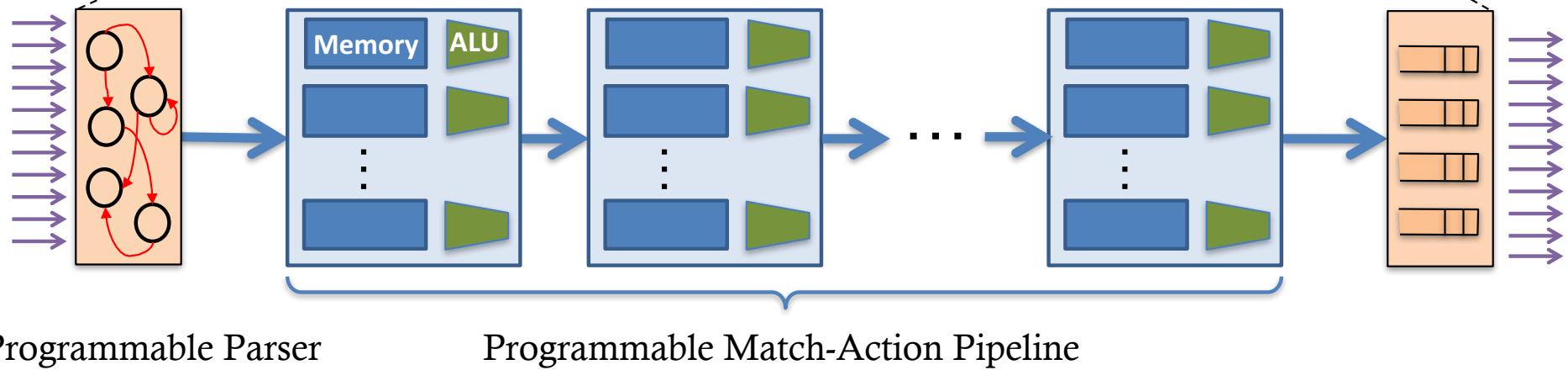


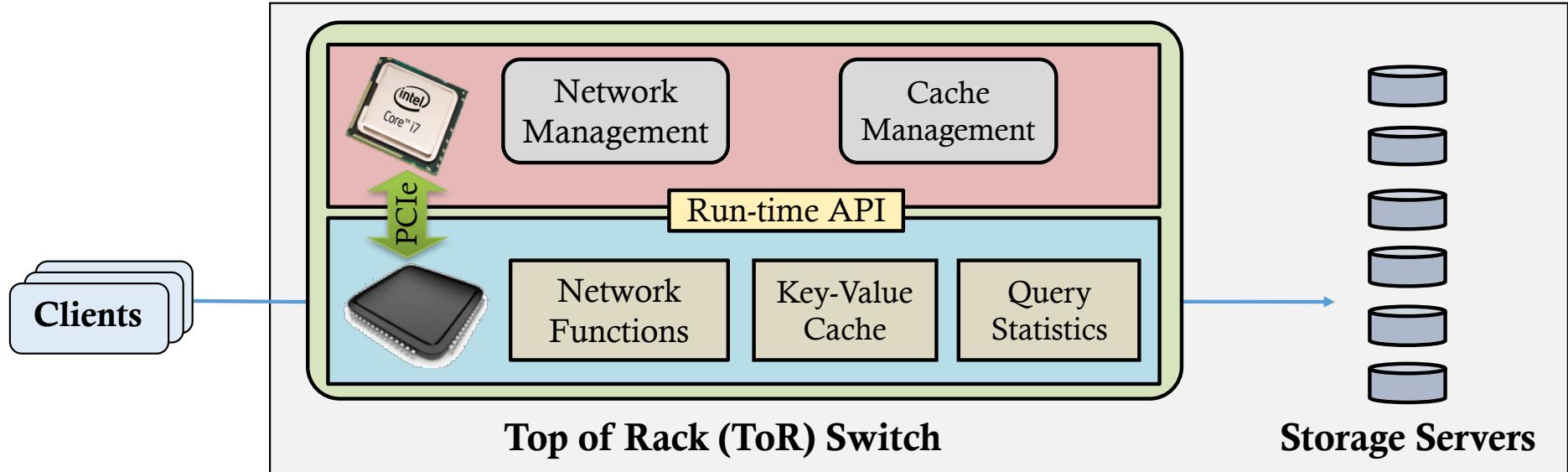
Control plane (CPU)

Data plane (ASIC)



Match + Action



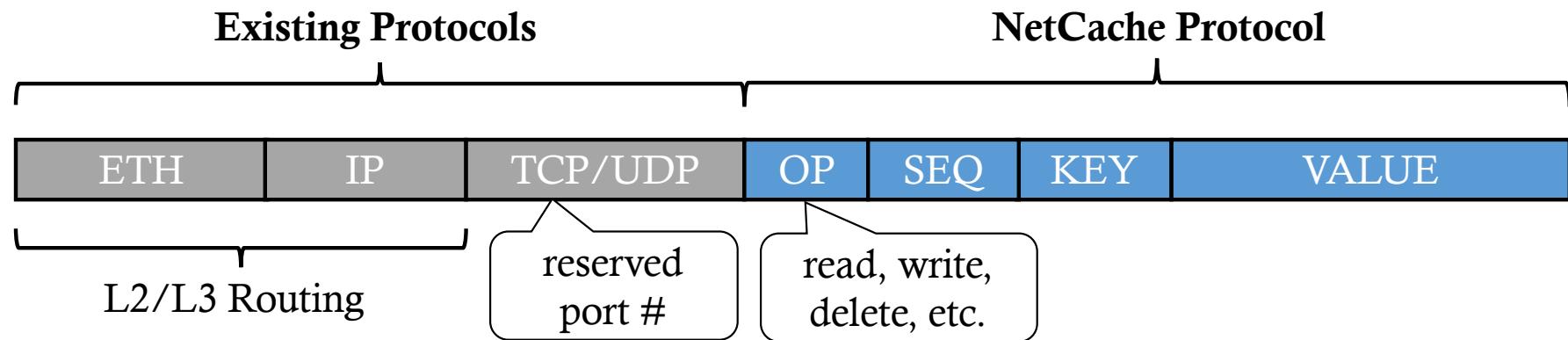


- **Switch data plane**
 - Key-value store to serve queries for cached keys
 - Query statistics to enable efficient cache updates
- **Switch control plane**
 - Insert hot items into the cache and evict less popular items
 - Manage memory allocation for on-chip key-value store

Key-value caching in network ASIC at line rate

- □ How to identify application-level packet fields ?
- How to store and serve variable-length data ?
- How to efficiently keep the cache up-to-date ?

NetCache Packet Format

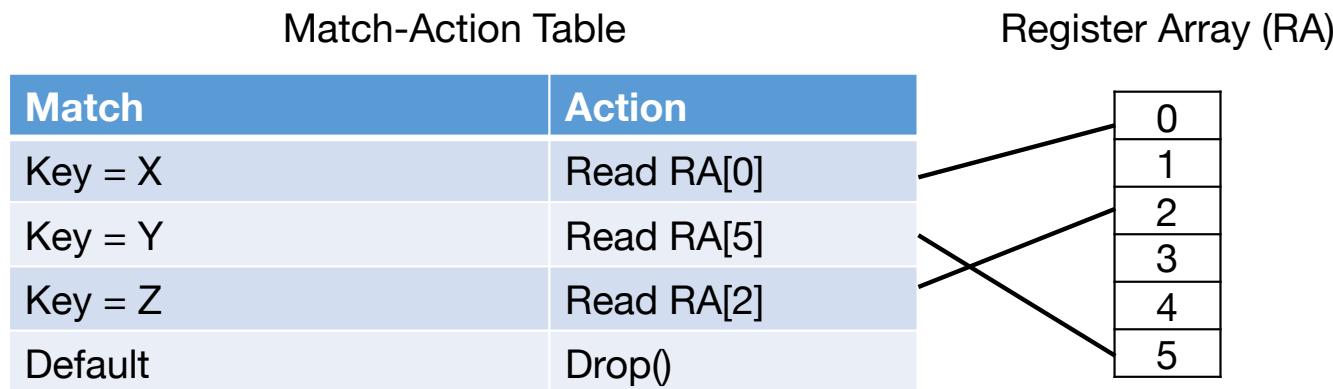


- Application-layer protocol: compatible with existing L2-L4 layers
- Only the top of rack switch needs to parse NetCache fields

Key-value caching in network ASIC at line rate

- How to identify application-level packet fields ?
- □ How to store and serve variable-length data ?
- How to efficiently keep the cache up-to-date ?

Key-value store using register arrays



Key Challenges:

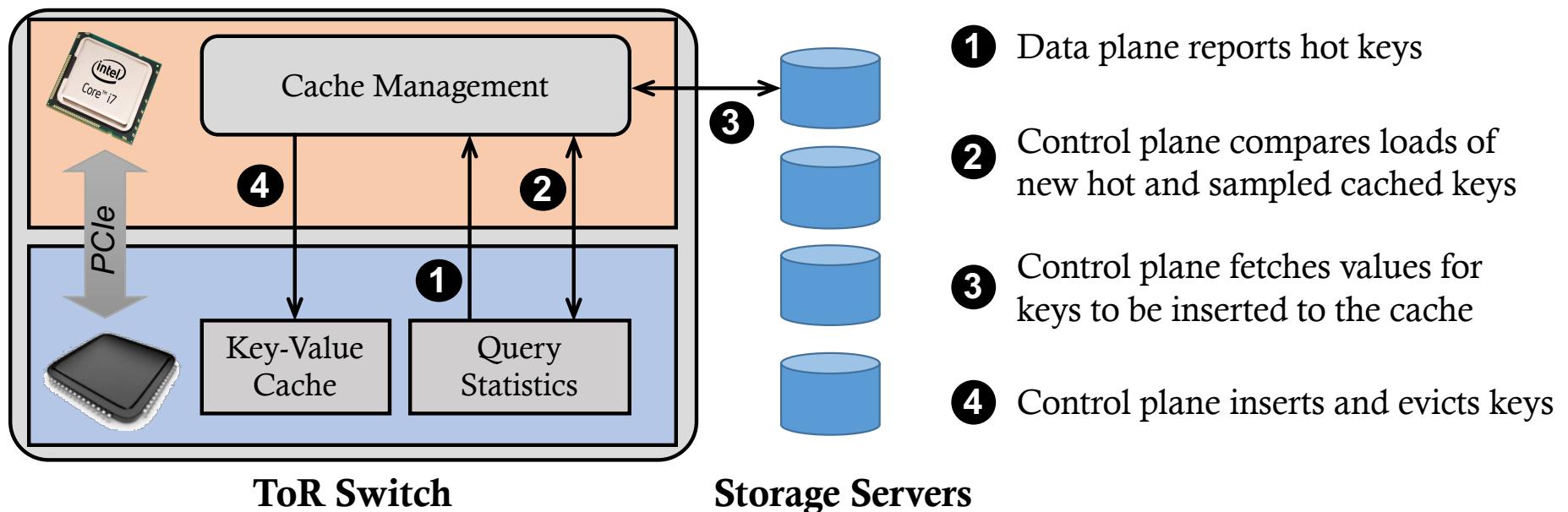
- ❑ No loop or string due to strict timing requirements
- ❑ Need to minimize hardware resources consumption
 - Number of table entries
 - Size of action data from each entry
 - Size of intermediate metadata across tables

Key-value caching in network ASIC at line rate

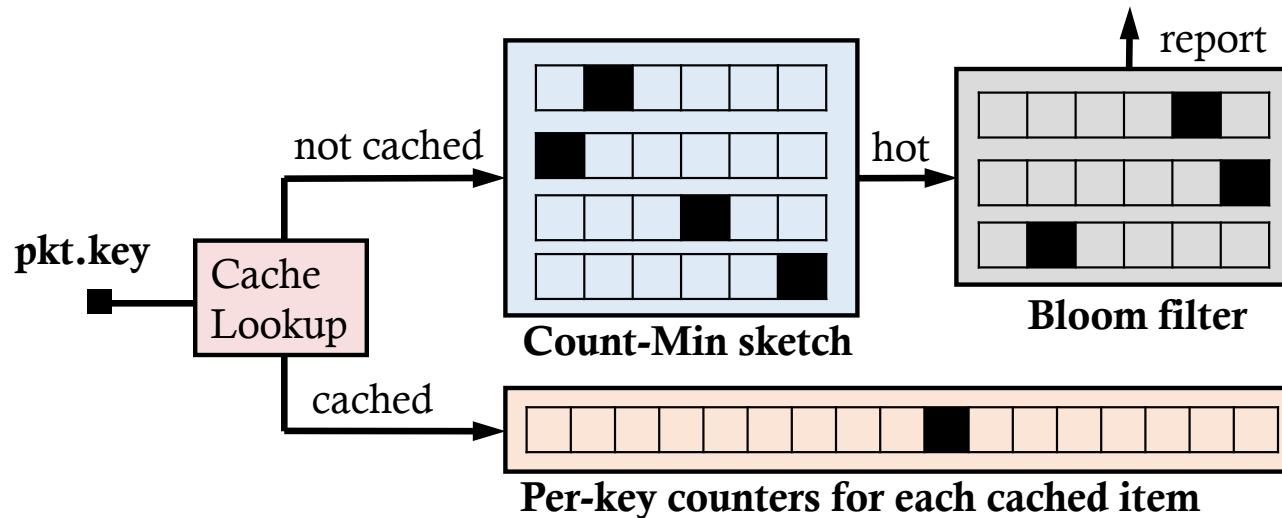
- How to identify application-level packet fields ?
- How to store and serve variable-length data ?
- □ How to efficiently keep the cache up-to-date ?

Cache insertion and eviction

- Challenge: cache the hottest $O(N \log N)$ items with **limited insertion rate**
- Goal: react quickly and effectively to workload changes with **minimal updates**



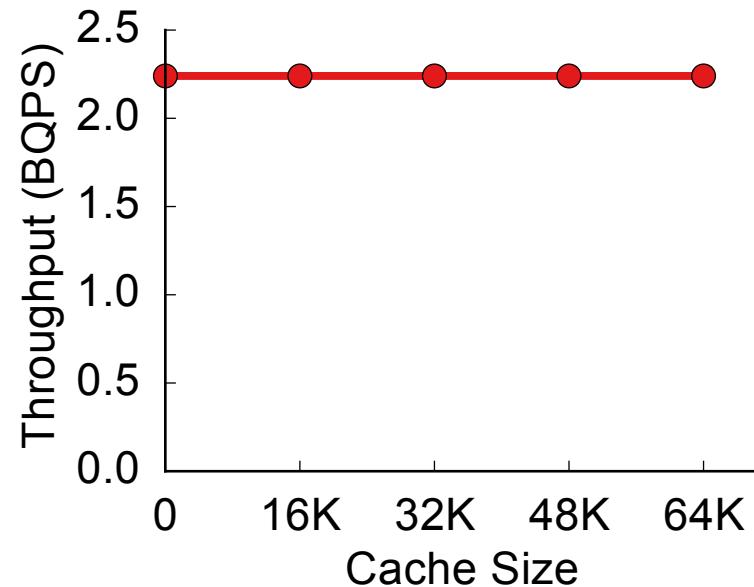
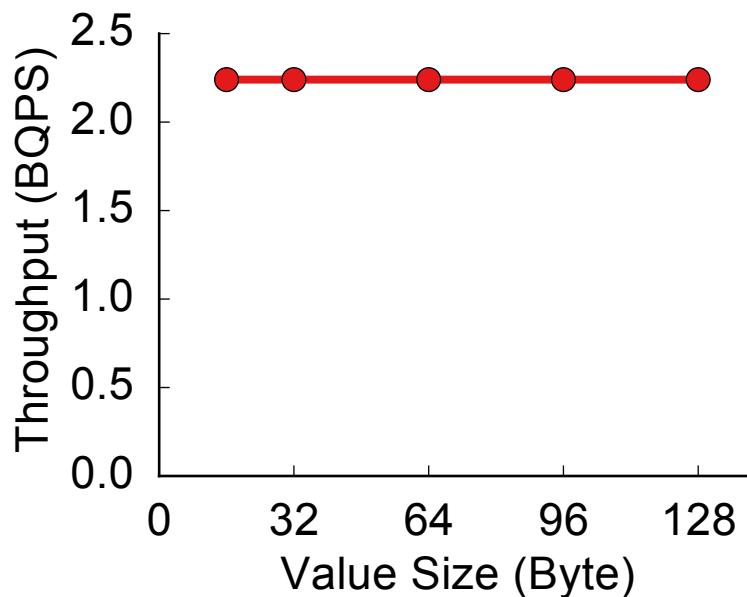
Query statistics in the data plane



- Cached key: per-key counter array
- Uncached key
 - Count-Min sketch: report new hot keys
 - Bloom filter: remove duplicated hot key reports

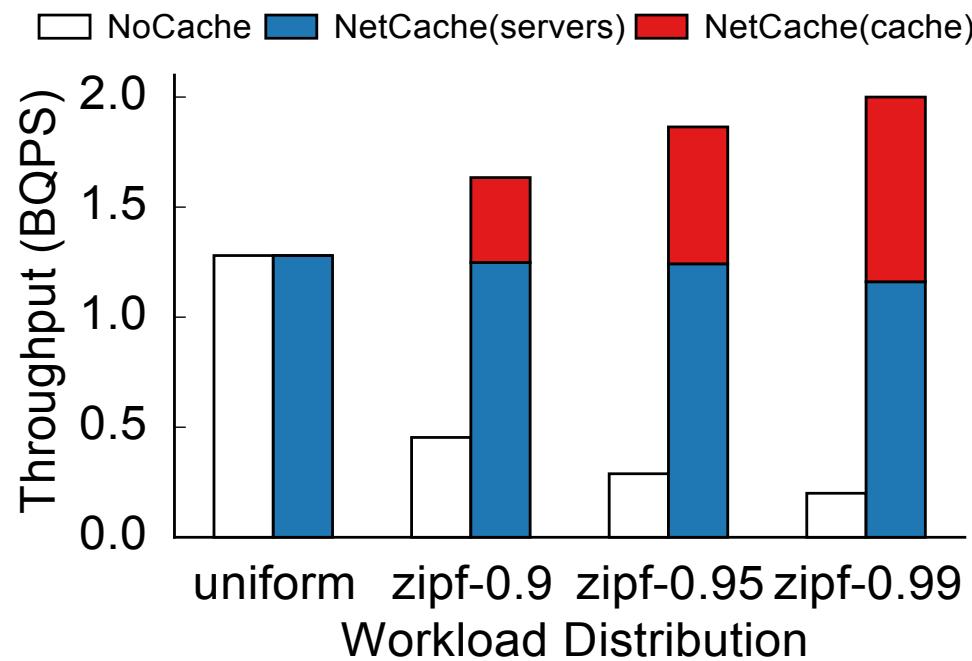
The “boring life” of a NetCache switch

Single switch benchmark



And its “not so boring” benefits

1 switch + 128 storage servers



3-10x throughput improvements

NetCache is a **rack-scale key-value store** that leverages
PISA-based caching to achieve

billions QPS throughput

&

~10 μ s latency

even under

highly-skewed

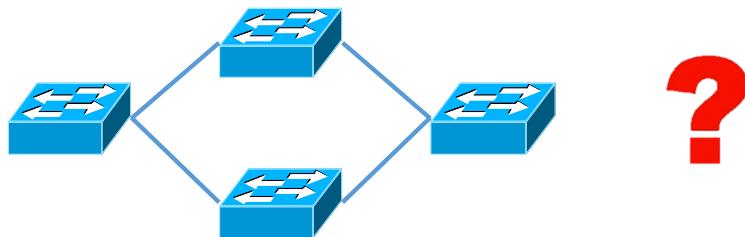
&

rapidly-changing

workloads.

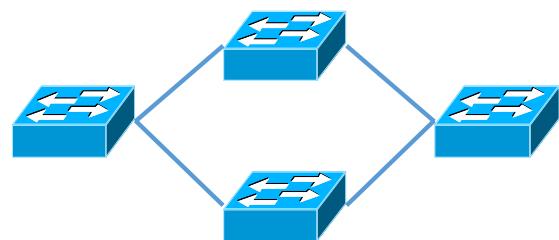


NetCache: lightning fast key-value **cache** enabled by PISA switches



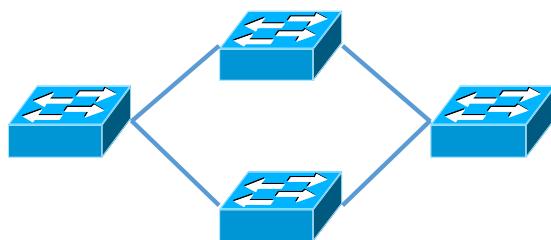


NetCache: lightning fast key-value **cache** enabled by PISA switches



NetChain: lightning fast **coordination** enabled by PISA switches

Conventional wisdom: avoid coordination



NetChain: lightning fast coordination
enabled by PISA switches

Open the door to rethink distributed systems design

Coordination services: fundamental building block of the cloud

Applications



Coordination Service



Provide critical coordination functionalities

Applications



Coordination
Service

Configuration
Management

Group
Membership

Distributed
Locking

Barrier

The core is a strongly-consistent, fault-tolerant key-value store

Applications



Configuration Management

Group Membership

Distributed Locking

Barrier

Coordination Service

Strongly-Consistent, Fault-Tolerant Key-Value Store



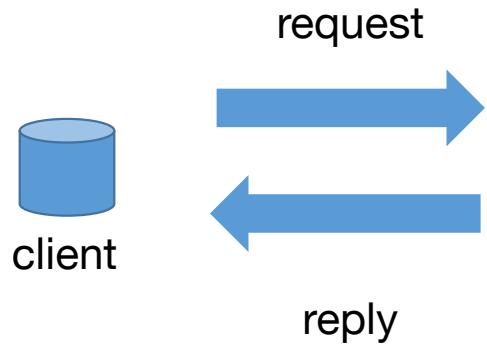
• • •



Servers

This Talk

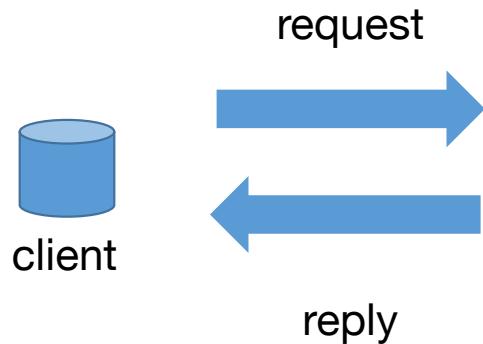
Workflow of coordination services



Can we do better?

- Throughput: at most server NIC throughput
- Latency: at least one RTT, typically a few RTTs

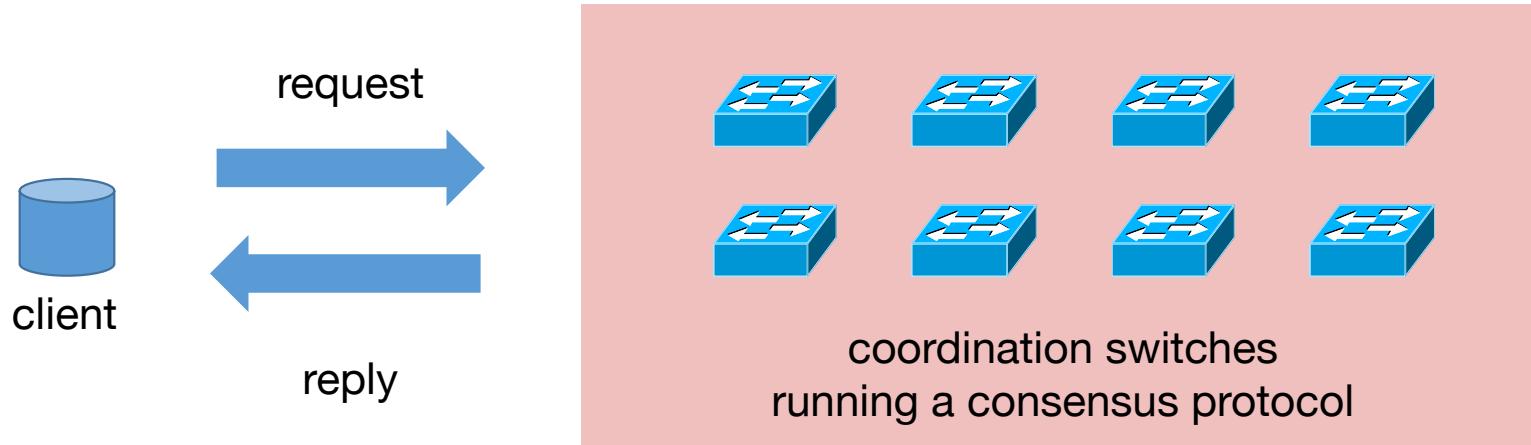
Opportunity: PISA-based coordination



Distributed coordination is
IO-intensive,
not computation-intensive.

| | Server | Switch |
|--------------------|----------------------|------------------------|
| Example | [NetBricks, OSDI'16] | Barefoot Tofino |
| Packets per second | 30 million | A few billion |
| Bandwidth | 10-100 Gbps | 6.5 Tbps |
| Processing delay | 10-100 us | < 1 us |

Opportunity: PISA-based coordination



- Throughput: switch throughput
- Latency: half of an RTT

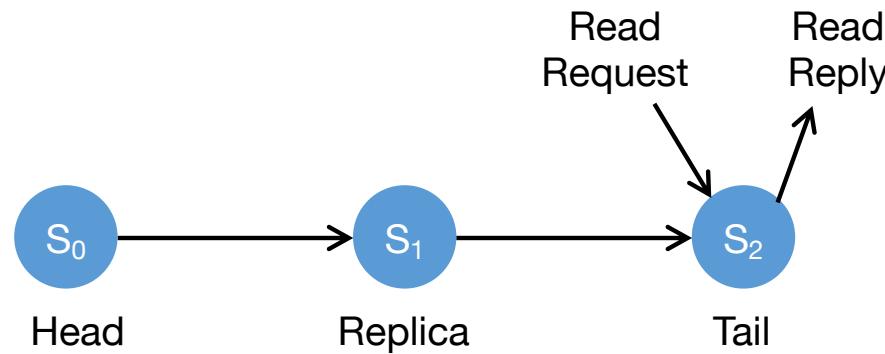
Design goals for coordination services

- High throughput
 - Low latency
 - Strong consistency
 - Fault tolerance
-
- The diagram illustrates the design goals for coordination services. It is divided into two main groups by curly braces. The first group, associated with a blue brace, contains the goals 'High throughput' and 'Low latency'. To its right, the text 'Directly from high-performance switches' is written in blue. The second group, associated with a red brace, contains the goals 'Strong consistency' and 'Fault tolerance'. To its right, the word 'How?' is written in red.
- Directly from
high-performance switches
- How?

Design goals for coordination services

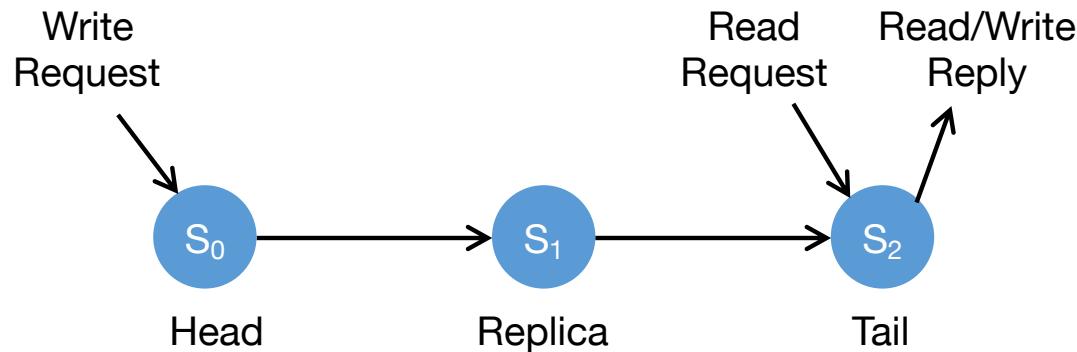
- High throughput
 - Low latency
 - Strong consistency
 - Fault tolerance
-
- The diagram illustrates the design goals for coordination services, grouped into two main categories. The first category, 'High throughput' and 'Low latency', is associated with 'high-performance switches' and is represented by a blue bracket on the right. The second category, 'Strong consistency' and 'Fault tolerance', is associated with 'PISA switches' and is represented by a red bracket on the right.
- Directly from
high-performance switches
- Chain replication with PISA switches

What is chain replication



- Storage nodes are organized in a **chain** structure
- Handle operations
 - **Read** from the **tail**

What is chain replication



- Storage nodes are organized in a **chain** structure
- Handle operations
 - **Read** from the **tail**
 - **Write** from **head** to **tail**
- Provide strong consistency and fault tolerance
 - Tolerate **f failures** with **f+1 nodes**

Division of labor in chain replication: a perfect match to network architecture

**Chain
Replication**

Storage Nodes

- Optimize for high-performance to handle read & write requests
- Provide strong consistency

Auxiliary Master

- Handle less frequent reconfiguration
- Provide fault tolerance

**Network
Architecture**

Network Data Plane

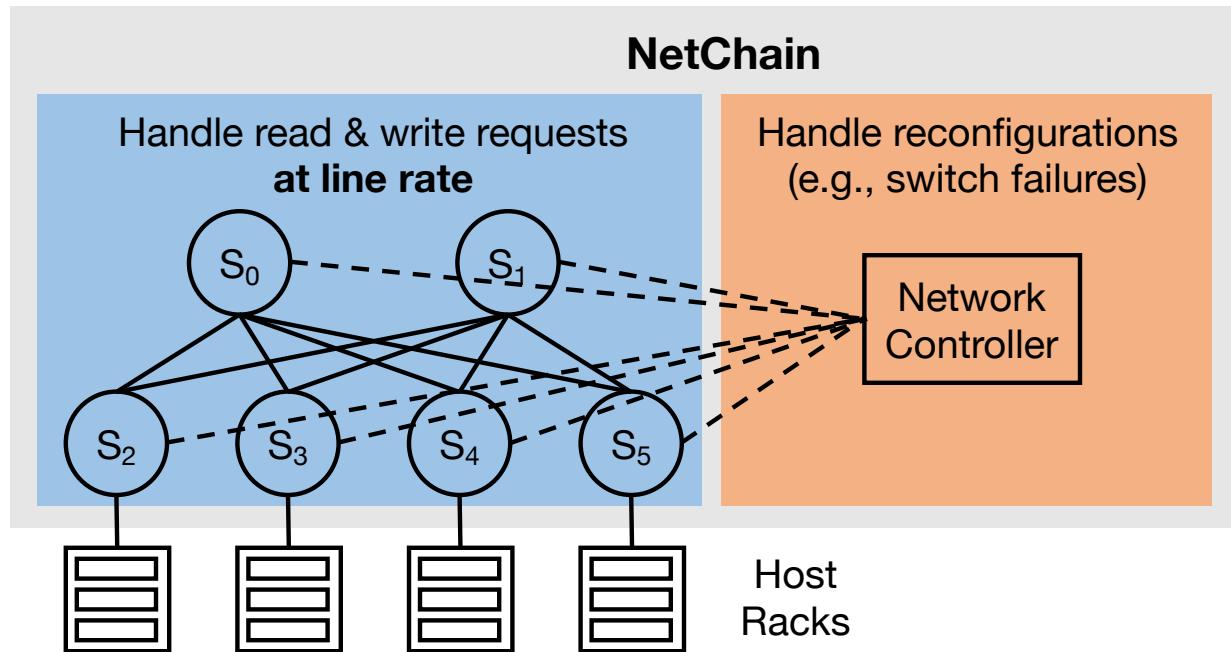
- Handle packets at line rate

Network Control Plane

- Handle network reconfiguration



NetChain overview



How to build a strongly-consistent, fault-tolerant, PISA-based key-value store

- How to store and serve key-value items?
 - How to route queries according to chain structure?
 - How to handle out-of-order delivery in network?
 - How to handle switch failures?
-
- The diagram consists of a vertical bracket on the right side of the slide, spanning the height of the first three bullet points. This bracket groups those three points together under the label "Data Plane". To the right of the fourth bullet point, there is a horizontal arrow pointing towards the right, indicating that this point belongs to the "Control Plane".
- Data
Plane
- Control
Plane

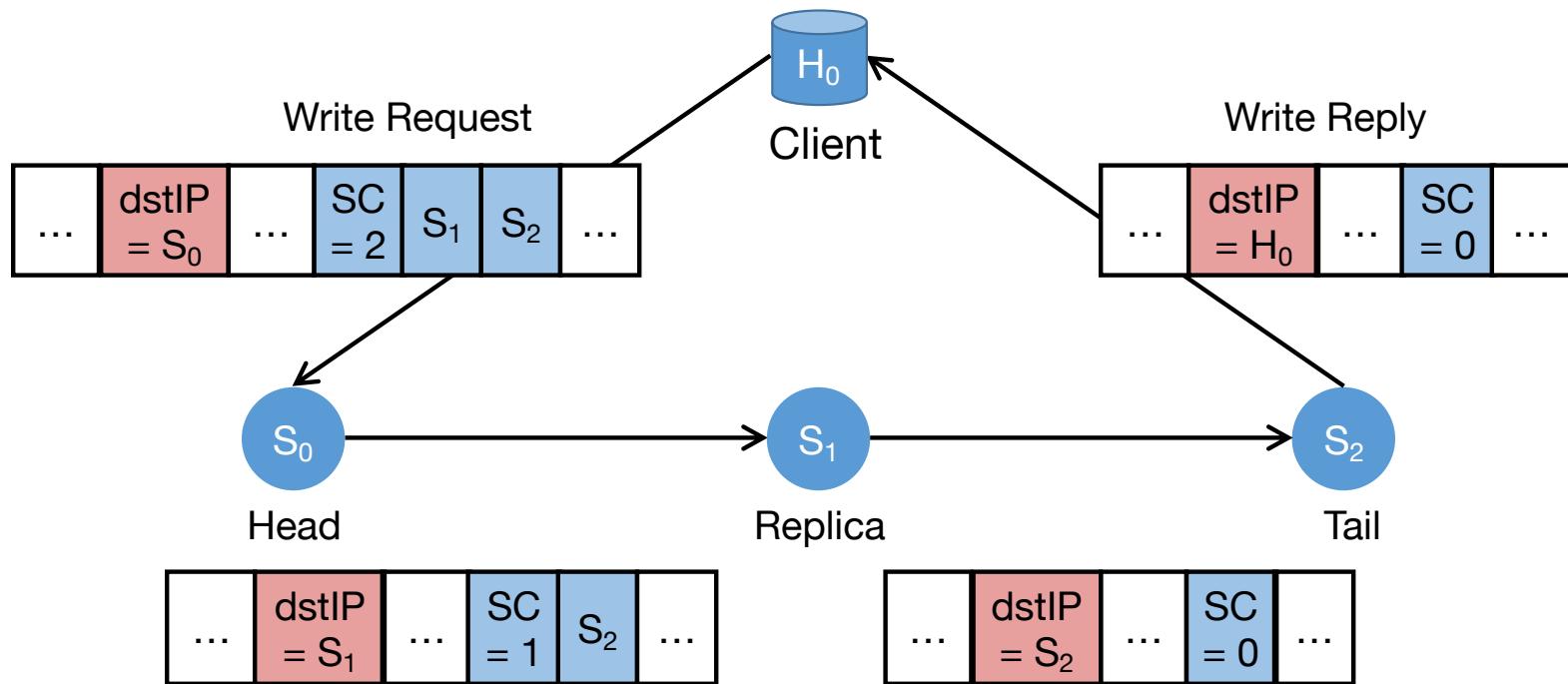
How to build a strongly-consistent, fault-tolerant, PISA-based key-value store

- How to store and serve key-value items?
NetCache 😊
 - How to route queries according to chain structure?
 - How to handle out-of-order delivery in network?
 - How to handle switch failures?
-
- The diagram consists of a vertical bracket on the right side of the slide, spanning from the middle of the first three list items up to the end of the fourth item. This bracket is labeled "Data Plane" to its right. Below the fourth item, there is a horizontal arrow pointing to the right, which is labeled "Control Plane" to its right.

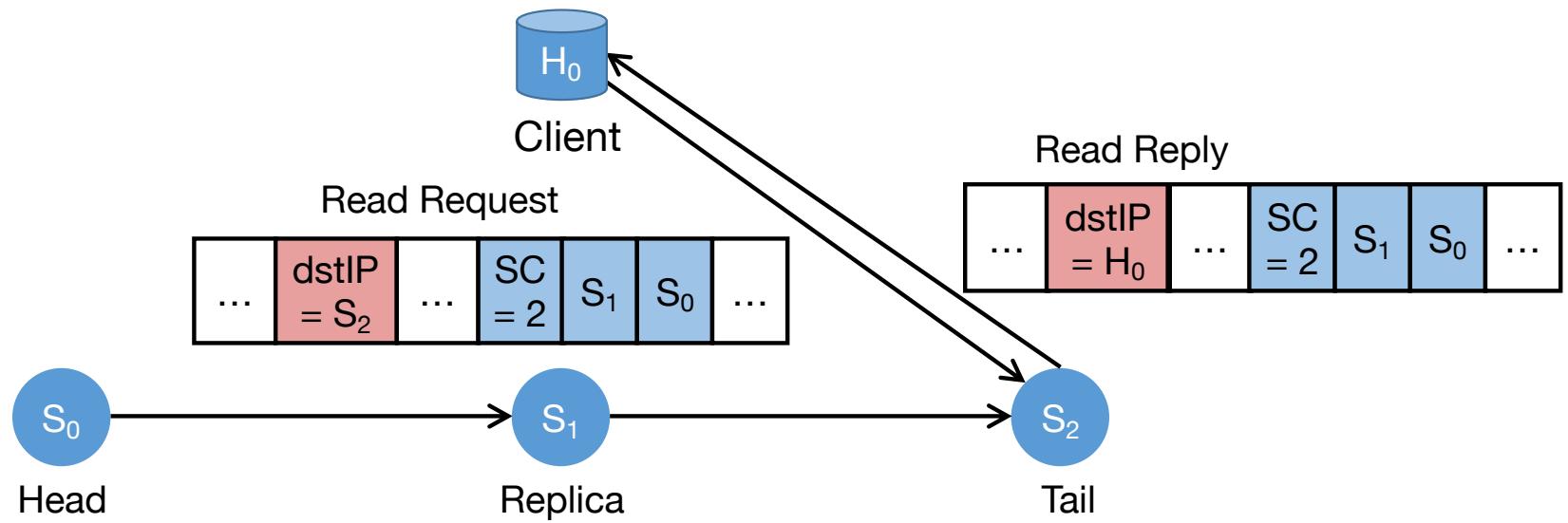
How to build a strongly-consistent, fault-tolerant, PISA-based key-value store

- How to store and serve key-value items?
 - How to route queries according to chain structure?
 - How to handle out-of-order delivery in network?
 - How to handle switch failures?
-
- The diagram consists of a vertical bracket on the right side of the slide, spanning the height of the first three bullet points. This bracket groups those three points together. To the right of the bracket, the text "Data Plane" is written vertically. Below the bracket, there is a horizontal arrow pointing to the right, positioned above the fourth bullet point. To the right of this arrow, the text "Control Plane" is written vertically.

NetChain routing: segment routing according to chain structure



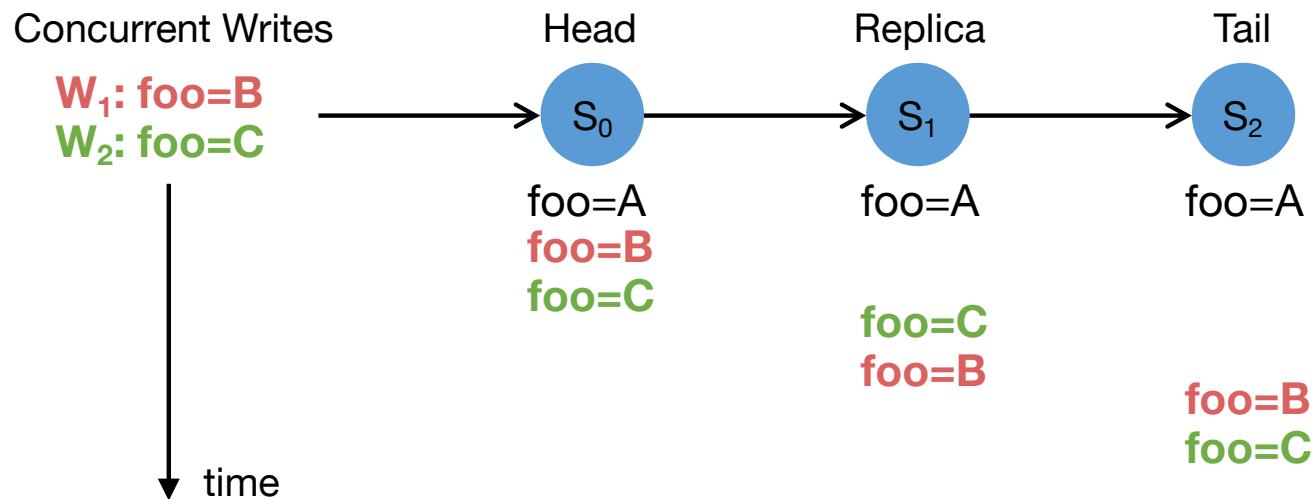
NetChain routing: segment routing according to chain structure



How to build a strongly-consistent, fault-tolerant, PISA-based key-value store

- How to store and serve key-value items?
 - How to route queries according to chain structure?
 - How to handle out-of-order delivery in network?
 - How to handle switch failures?
-
- The diagram consists of a vertical bracket on the right side of the slide, spanning from the middle of the second question down to the middle of the fourth question. This bracket groups the first three questions together and is labeled "Data Plane" to its right. Below the fourth question, there is a horizontal arrow pointing to the right, which is labeled "Control Plane" above it.

Problem of out-of-order delivery



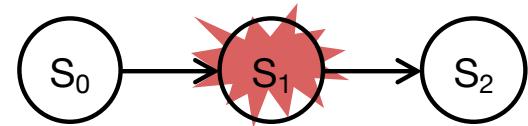
Serialization with sequence number

How to build a strongly-consistent, fault-tolerant, PISA-based key-value store

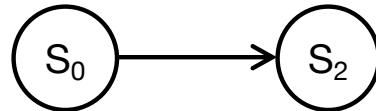
- How to store and serve key-value items?
 - How to route queries according to chain structure?
 - How to handle out-of-order delivery in network?
 - How to handle switch failures?
-
- The diagram consists of a vertical bracket on the right side of the slide, spanning the height of the first three bullet points. This bracket groups those three points together under the label "Data Plane". To the right of the fourth bullet point, there is a horizontal arrow pointing towards the right, indicating that this final point belongs to the "Control Plane".
- Data
Plane
- Control
Plane

Handle a switch failure

Before failure: tolerate f failures with $f+1$ nodes

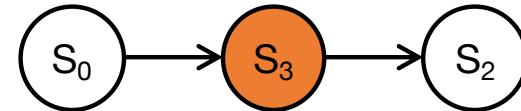


Fast Failover



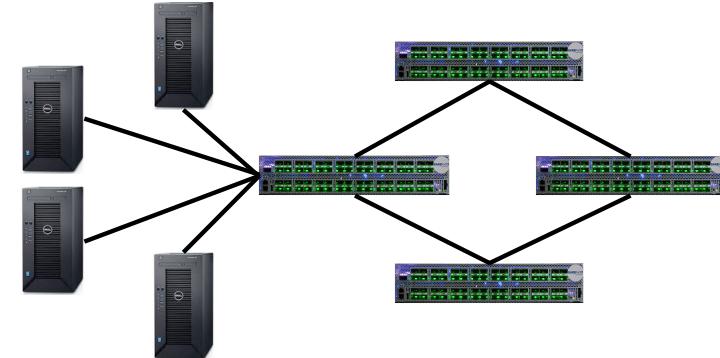
- Failover to remaining f nodes
- Tolerate $f-1$ failures
- Efficiency: only need to update **neighbor** switches of failed switch

Failure Recovery



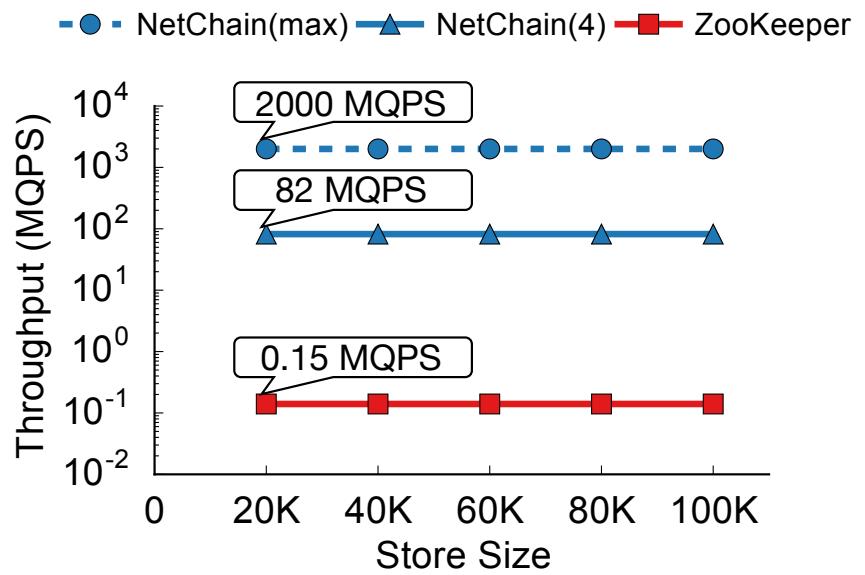
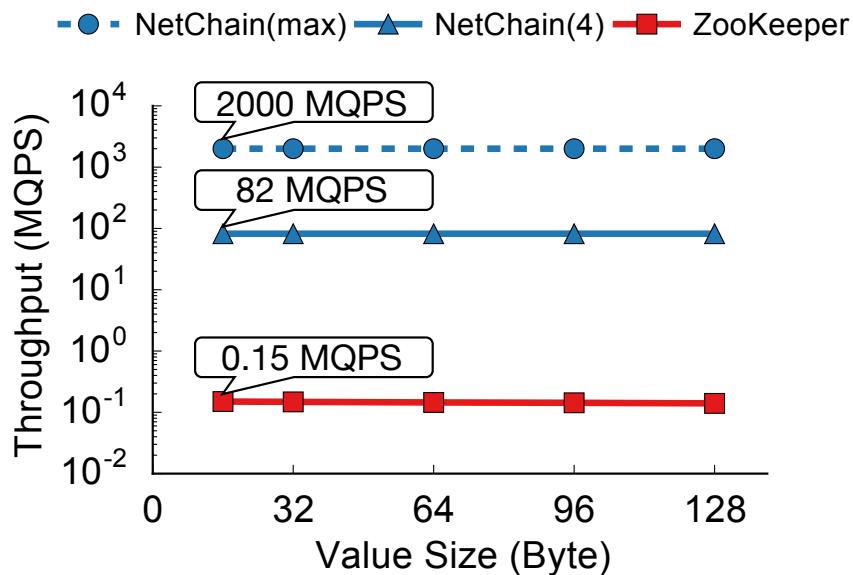
- Add another switch
- Tolerate f failures again
- Consistency: **two-phase atomic switching**
- Minimize disruption: **virtual groups**

Implementation

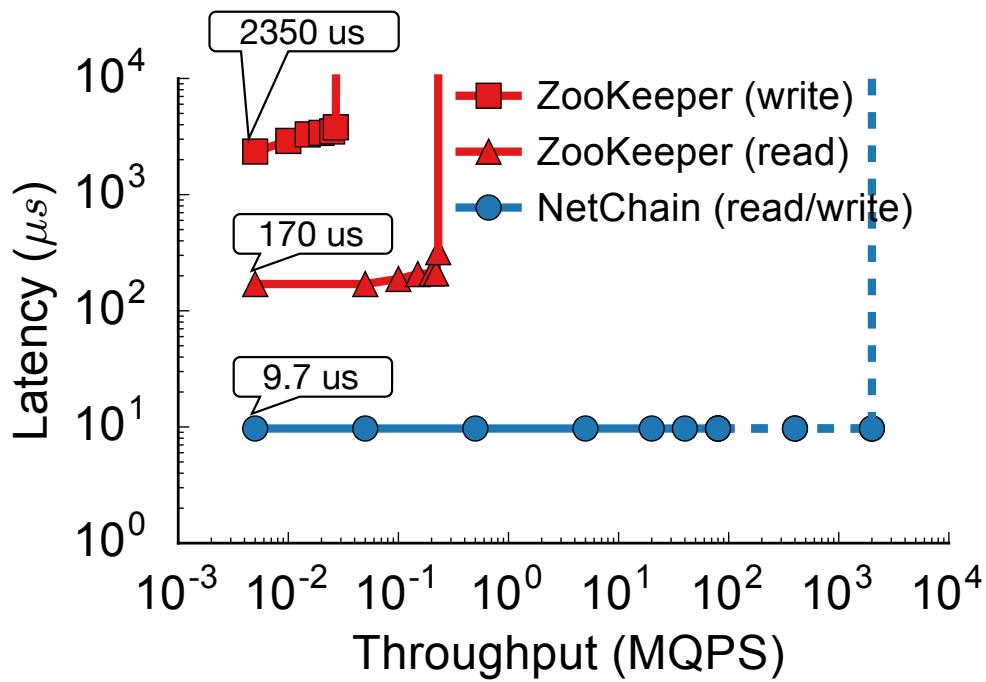


- **Testbed**
 - 4 Barefoot Tofino switches and 4 commodity servers
- **Switch**
 - P4 program on 6.5 Tbps Barefoot Tofino
 - Routing: basic L2/L3 routing
 - Key-value store: up to 100K items, up to 128-byte values
- **Server**
 - 16-core Intel Xeon E5-2630, 128 GB memory, 25/40 Gbps Intel NICs
 - Intel DPDK to generate query traffic: up to 20.5 MQPS per server

Orders of magnitude higher throughput



Orders of magnitude lower latency



Conclusion

- **Moore's law** is ending...
 - **Specialized** processors for **domain-specific** workloads: GPU servers, FPGA servers, TPU servers...
- **PISA servers**: new generation of ultra-high performance systems for **IO-intensive workloads** enabled by PISA switches
 - NetCache: fast key-value caching built with PISA switches
 - NetChain: fast coordination built with PISA switches

Thanks!