Pegasus: Tolerating Skewed Workloads in Distributed Storage with In-Network Coherence Directories

Jialin Li, Jacob Nelson, Ellis Michael, Xin Jin, Dan R. K. Ports

NUS National University of Singapore

Microsoft Research

W UNIVERSITY of WASHINGTON

JOHNS HOPKINS UNIVERSITY

Many real-workloads are **skewed** and **dynamic**

Skewed workloads lead to
load imbalance

popular objects can **overload** storage servers

Skewed workloads are **diverse**

- read-heavy
- write-heavy
- read-write mixed

OSDI 20 ATTEND PROGRAM PARTICIPATE SPONSORS ABOUT

A large scale analysis of hundreds of in-memory cache clusters at Twitter

Workload Analysis of a Large-Scale Key-Value Store

- small objects
- large objects
- combination of both

Google

Fast key-value stores: An idea whose time has come and gone

Two approaches to deal with **highly skewed workloads**

Caching

- Cache popular objects in a faster tier
- Caching tier *absorbs* traffic to popular objects
- More **uniform** load on backend storage servers

Selective Replication

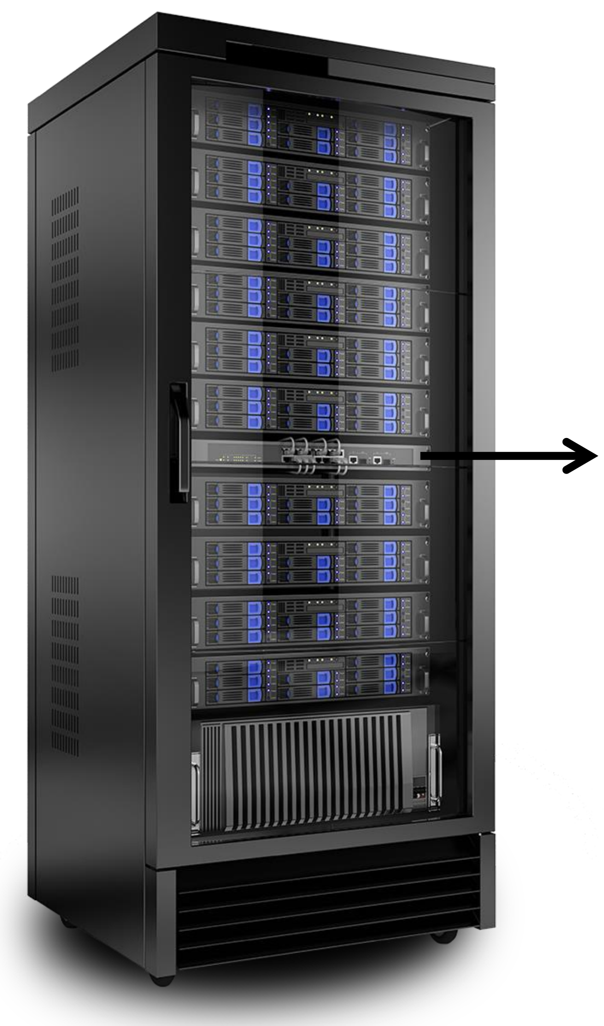
- Replicate popular objects on multiple servers
- Requests to replicated objects can be forwarded to *any* replica
- **Distribute** load across servers

Existing solutions have **limitations**

Caching	Selective Replication
non read-heavy workloads ✗	dynamic replicated objects & locations ✗
fast in-memory storage systems ✗	strong consistency ✗

Our system: **Pegasus**

rack-scale
storage system



programmable
top-of-rack
switch

highly skewed and
dynamic workloads

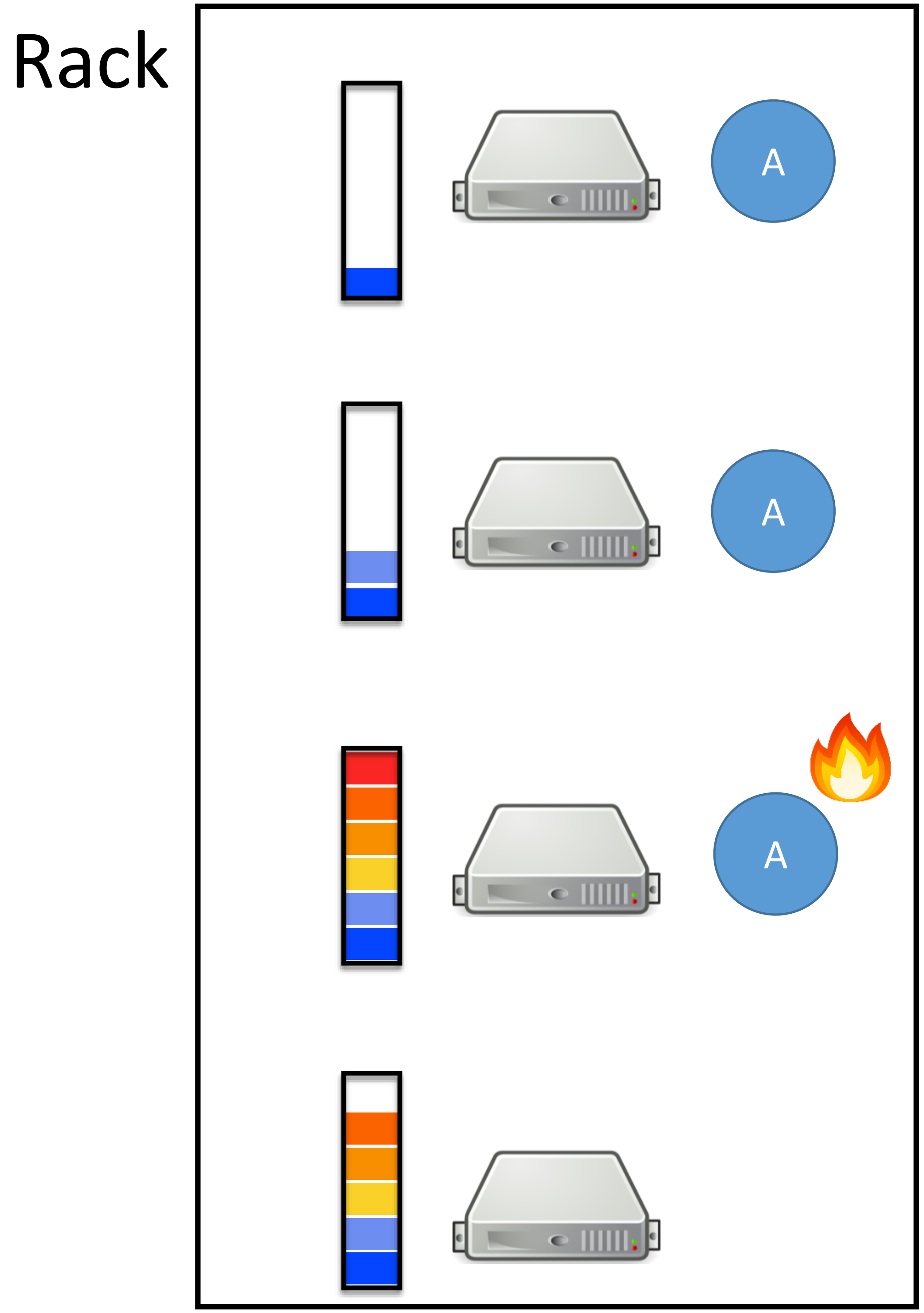
fast in-memory
storage systems

any object sizes

all read-write ratios

strong consistency

Observation: rack as a whole has **spare processing capacity**



How to route requests to the right server?

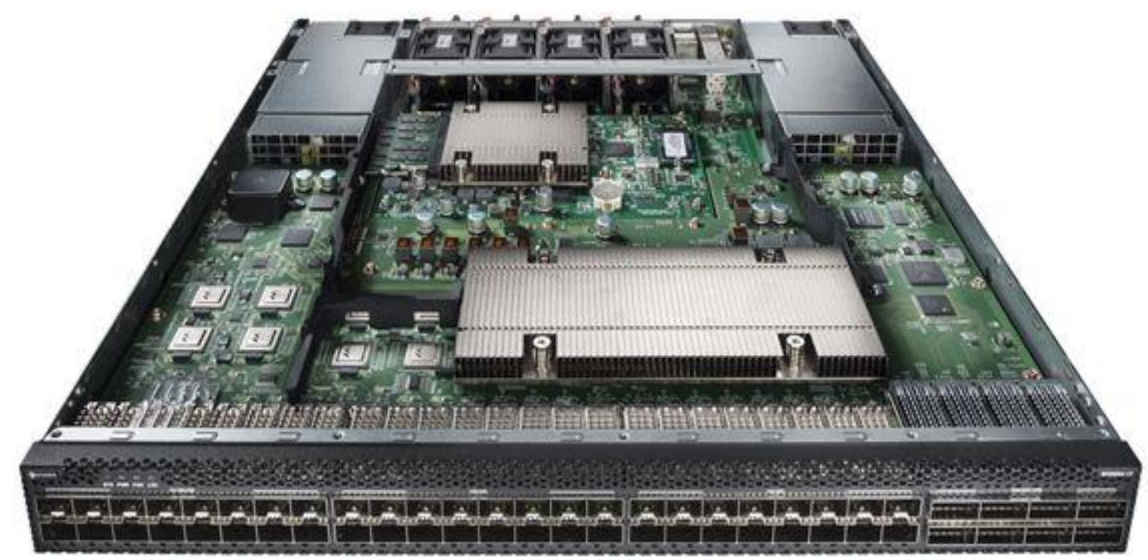
How to ensure consistency?

Pegasus' approach

selective replication

+

in-network **coherence directory**

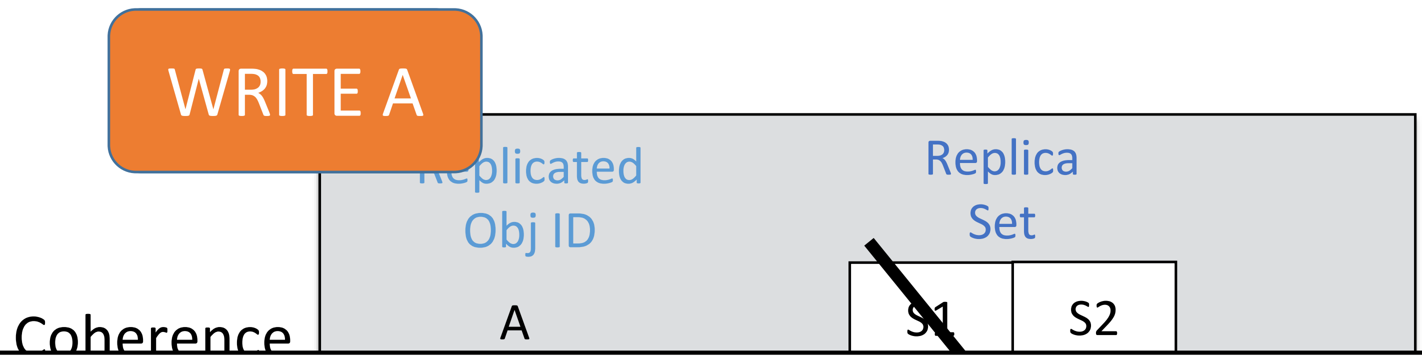


Barefoot P4 switch

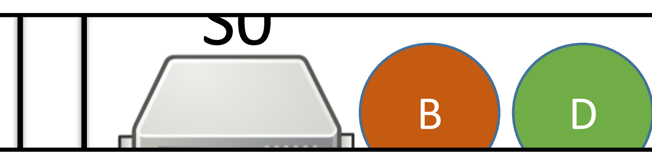
Coherence directory for replicated data

- Inspired by CPU cache coherence protocols
- Centralized directory that **tracks:**
  - Which objects are replicated
  - Location of replicated objects
- **Forwards** requests to server with spare capacity
- Ensures **strong consistency**

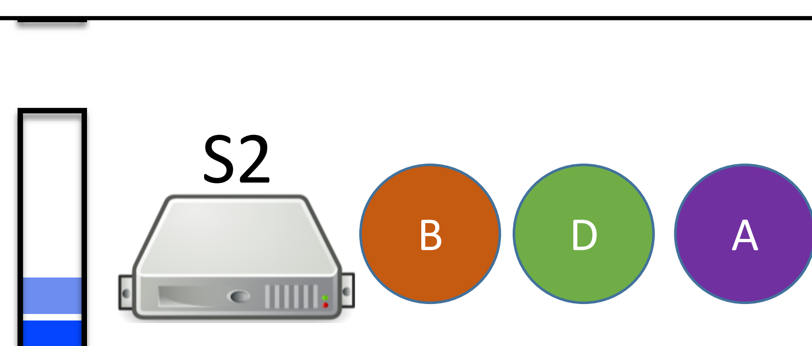
Coherence directory illustrated



How to build a directory that can handle **all** the traffic?



How to ensure consistency with a **lightweight** protocol?



Implementing coherence directory **in the network**

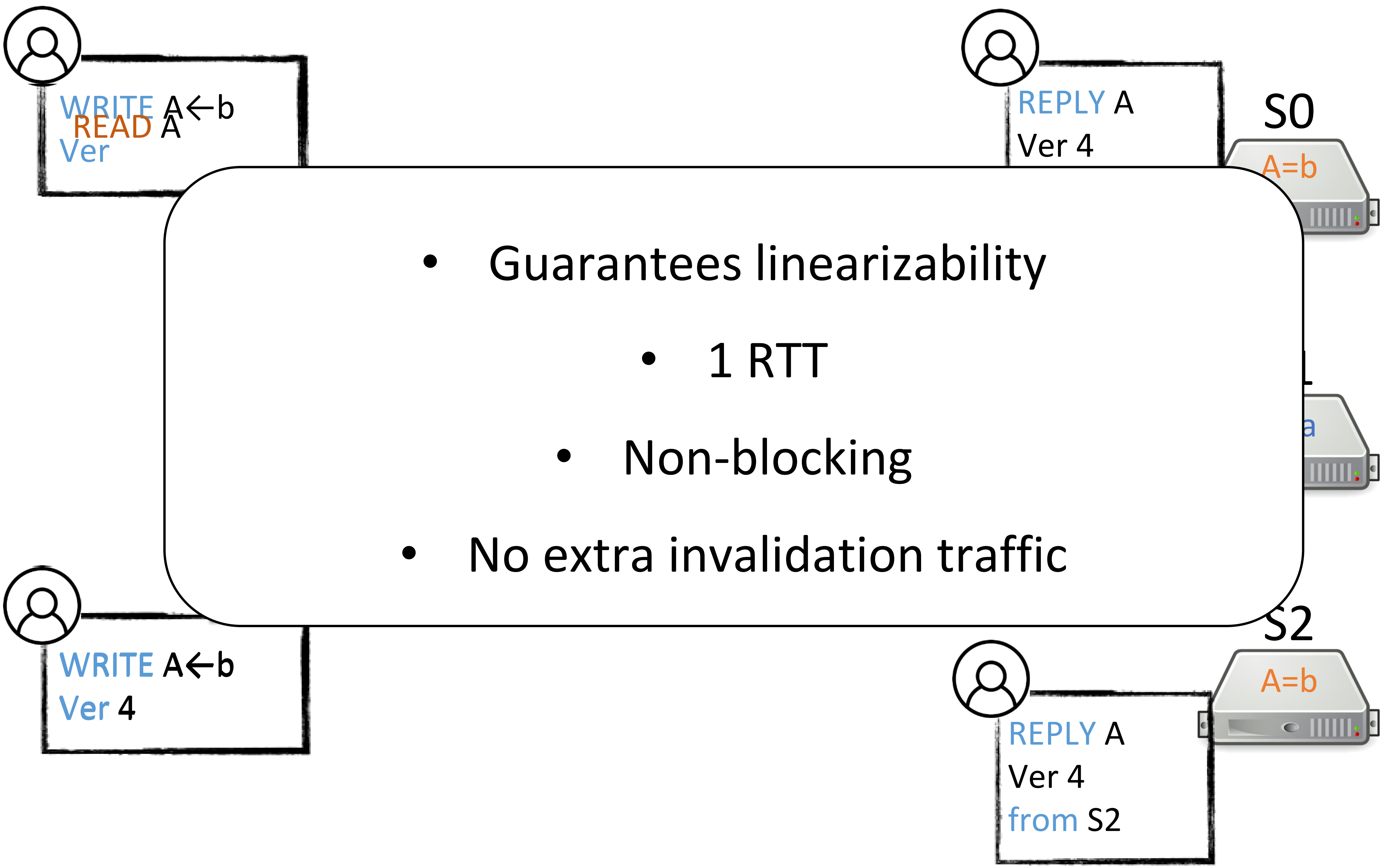
rack-scale
storage system

- All requests and replies traverse the ToR switch
- ToR serves as a **central point**
- **Line-rate** packet processing
  - No throughput bottleneck
  - Zero latency overhead

Pegasus **version-based** coherence protocol

- Switch processes **all** requests
- Switch tracks which servers have the **latest** copy
- Updates the directory when receiving replies
- How to deal with network asynchrony?
  - Use **version numbers**!

Pegasus’ coherence protocol in action

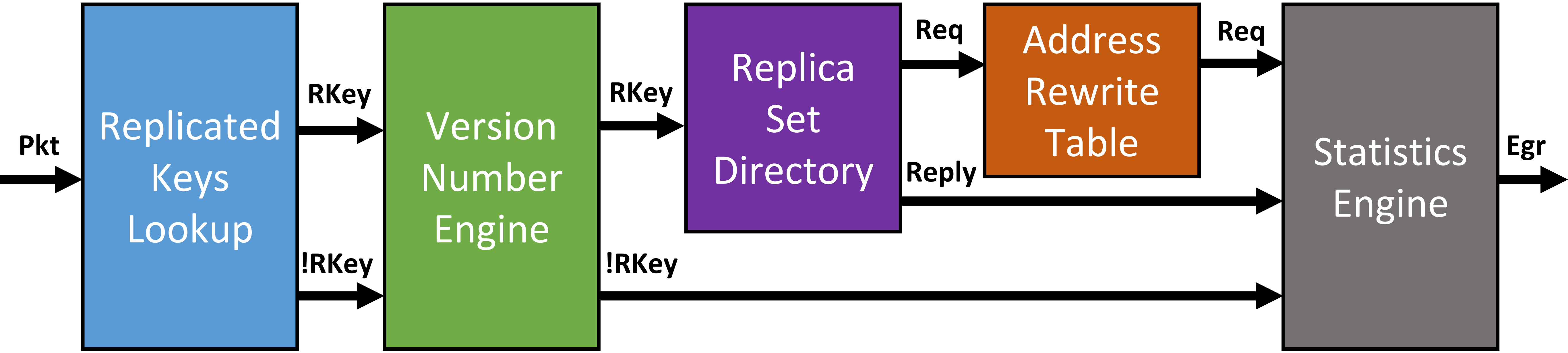


- Guarantees linearizability
- 1 RTT
- Non-blocking
- No extra invalidation traffic

Other protocol details

- Adding and removing replicated objects
  - Pegasus controller monitors object access frequencies
  - Updates coherence directory with most popular objects
- Avoiding duplicate requests
  - Server maintains a client table
  - Retried write requests forward to the same server
- Server selection policy
  - Random
  - Weighted round-robin
- Handling server and rack failure
  - Multi-rack deployment
  - Each rack runs a Pegasus instance
  - Chain replication across racks

Coherence directory
switch implementation



Efficient switch implementation

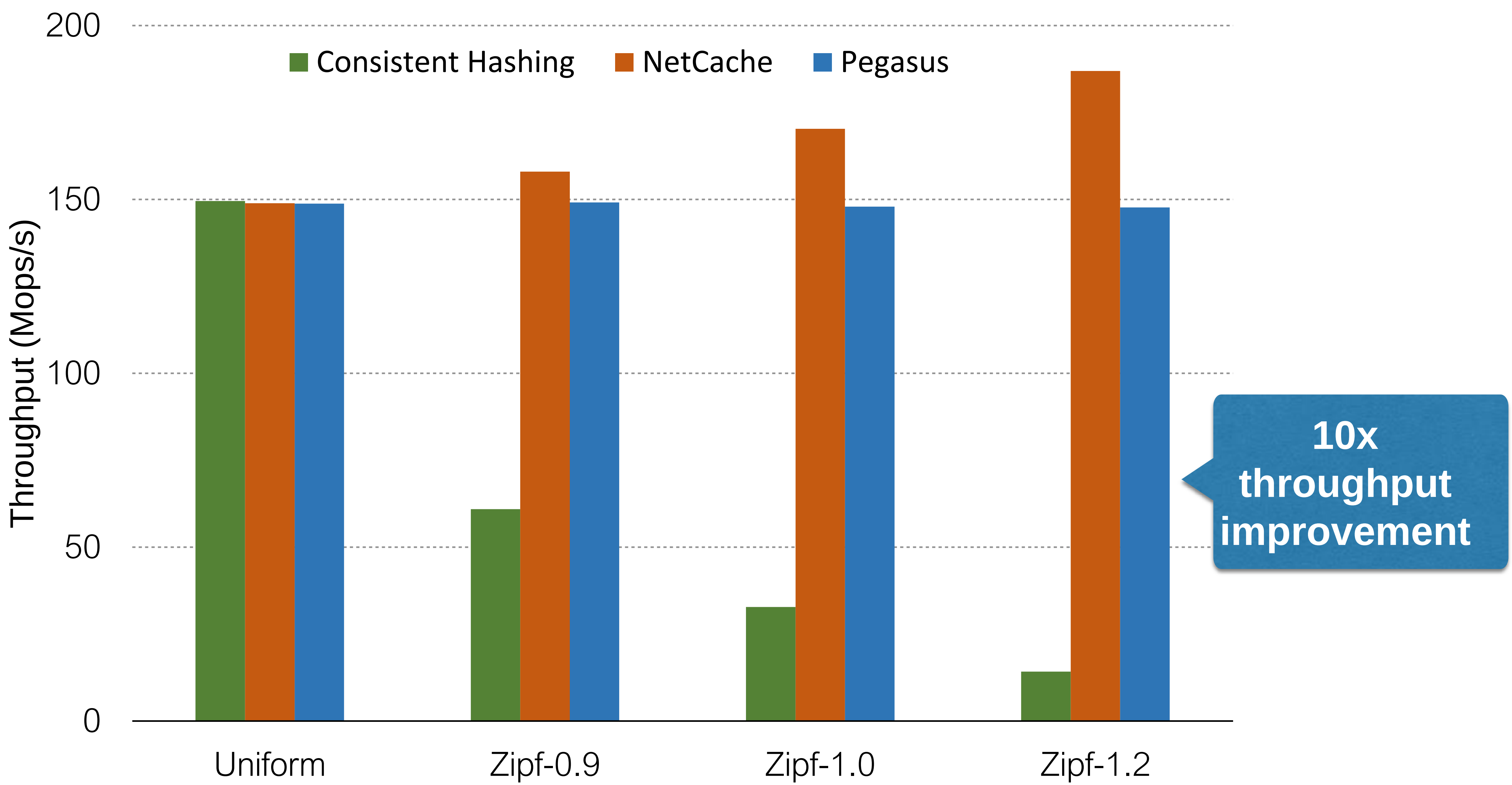
- Switch only stores small **metadata**
- Only needs to replicate the most popular **O(nlogn)** objects, where n is the **number of servers** (extension of [1])
- Consumes less than **3.5%** of switch SRAM

[1] Small Cache Big Effect: Provable Load Balancing for Randomly Partitioned Cluster Services. Bin Fan et al., 2011

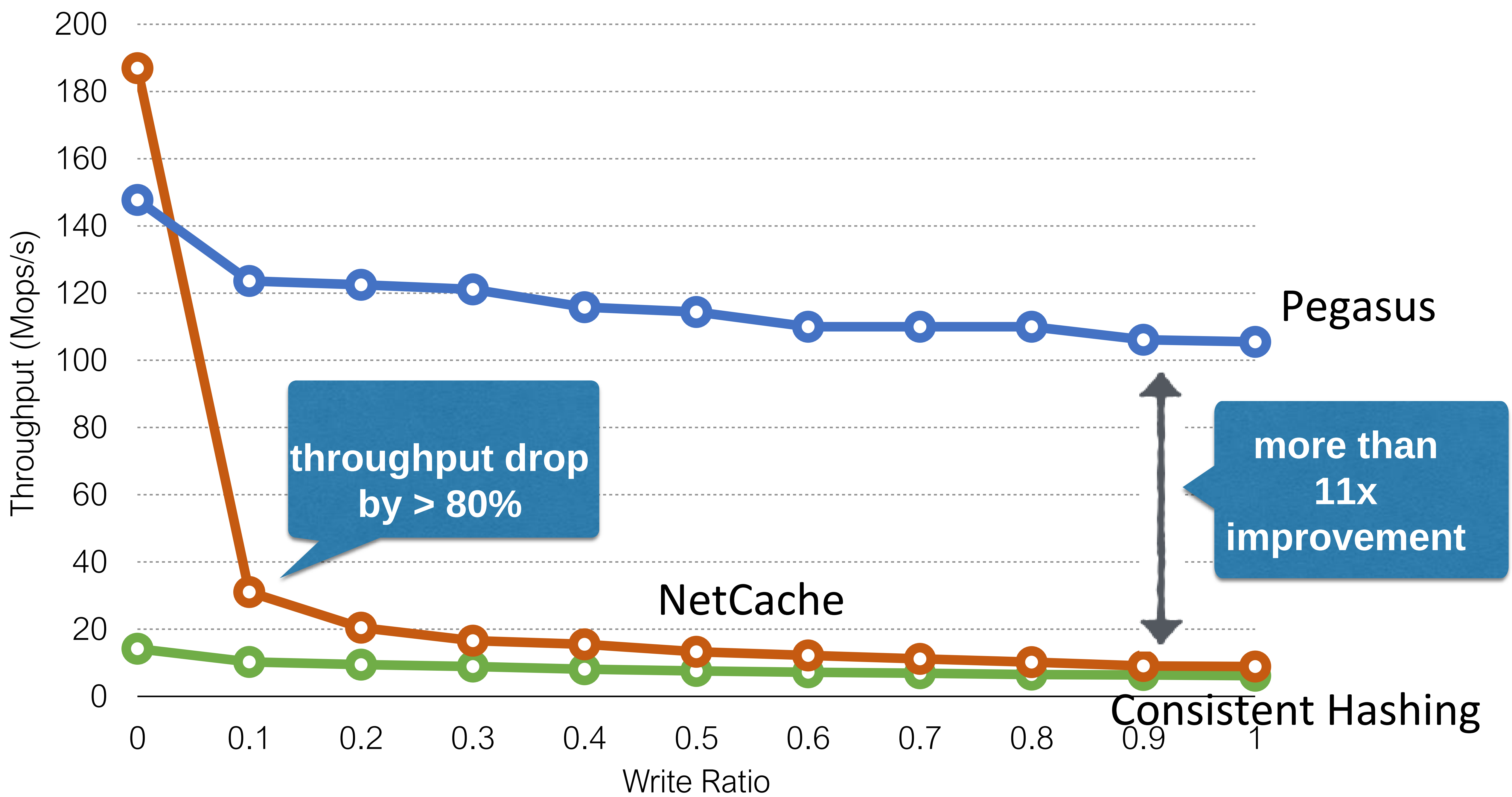
Evaluation

- 28 nodes with dual socket Xeon Silver 4114, 48 GB RAM per socket
- Mellanox ConnectX-4 25Gb NICs
- Connected to an Arista 7170-64S (Barefoot Tofino-based) switch
- Open-loop clients, Poisson inter-arrival distribution
- Client requests choose keys following:
  - Uniform distribution
  - Zipf distribution (skewed workload)
- Measured max throughput subject to a 99%-latency SLO
- Comparison systems:
  - Consistent hashing with 16 virtual nodes
  - NetCache

Pegasus is effective under **highly skewed** workloads



Pegasus equally effective under **different** read/write ratios



Conclusion

- New approach to distributed storage load balancing
  - Build a coherence directory directly in ToR switch
  - Tracks location and forwards requests for popular objects
  - Guarantees strong consistency
- Resulting system: Pegasus
  - > 10x throughput improvement compared to existing approaches
  - Equally effective under a variety of workloads
    - Read-heavy, write-heavy, read-write mixed
    - Small and large objects