

Pegasus: Load-Aware Selective Replication with an In-Network Coherence Directory



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

W UNIVERSITY *of* WASHINGTON

Microsoft
Research


JOHNS HOPKINS
UNIVERSITY

Many workloads are skewed and dynamic

Profile snippet for user 'Jl @j'. A post titled 'Happ' is shown with a timestamp of '12:09 PM', '95,015 Re', and '16K' comments.

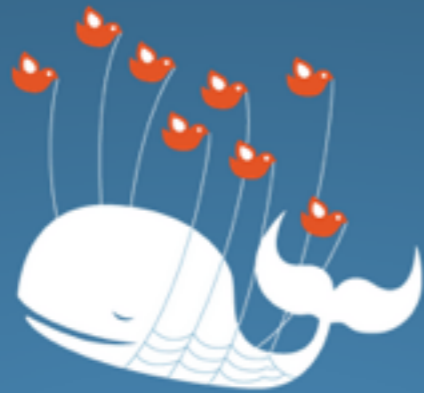


Tall
Head

Twitter post from Super Bowl (@SuperBowl) with the text 'SUPER. BOWL. WEEK. #SBLIII'. The image shows Tom Brady in a Patriots jersey (number 12) and Matt Ryan in a Rams jersey (number 16). The caption reads 'PATRIOTS NE LIII LA RAMS SUPER BOWL ATLANTA 02 03 19'. The post is dated '2:13 AM - 28 Jan 2019' and has '2,782 Retweets' and '11,296 Likes'.

Profile snippet for user 'alin Li alinLi14'. A comment 'cool 😊' is visible. An orange arrow points from the Super Bowl tweet to this profile.

Skewed workloads lead to **load imbalance**



Twitter is over capacity.

Please wait a moment and try again. For more information, check out **Twitter Status**.

Bahasa Indonesia Bahasa Melayu Deutsch English Español Filipino Français Italiano Nederlands Português Türkçe
Русский हिन्दी 日本語 简体中文 繁體中文 한국어

© 2012 Twitter About Help Status



One approach: use caching to handle skewed workloads



Caching Layer



Limitations of Caching

- Caching layer needs to be **magnitude-faster** [Fan et al. '11]
 - Building such a layer for fast in-memory storage systems is challenging!
- Only effective for **read-heavy** workloads

Pegasus' Approach

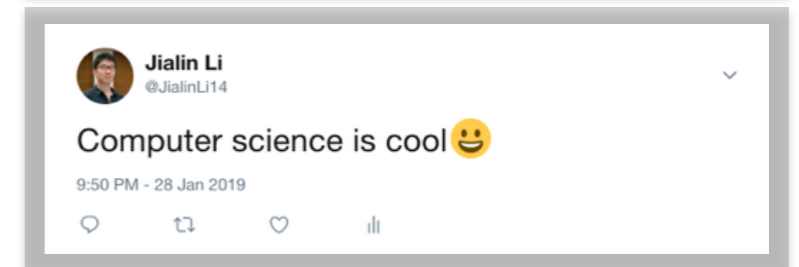


provable load balancing for both **read-heavy**
and **write-heavy** workloads

**Selective
Replication**

**In-Network
Coherence Directory**

Selective Replication



Q: How many objects should we replicate?

Storage Overhead **Low**

Load Balancing **Weak**

Storage Overhead **High**

Load Balancing **Strong**



Replicate Fewer Objects

Replicate More Objects

We only need to replicate the most popular
 $O(n \log n)$ objects

n is the number of **storage servers**

Forward request to the **least-loaded server**

(generalization of previous result [Fan et al. '11])

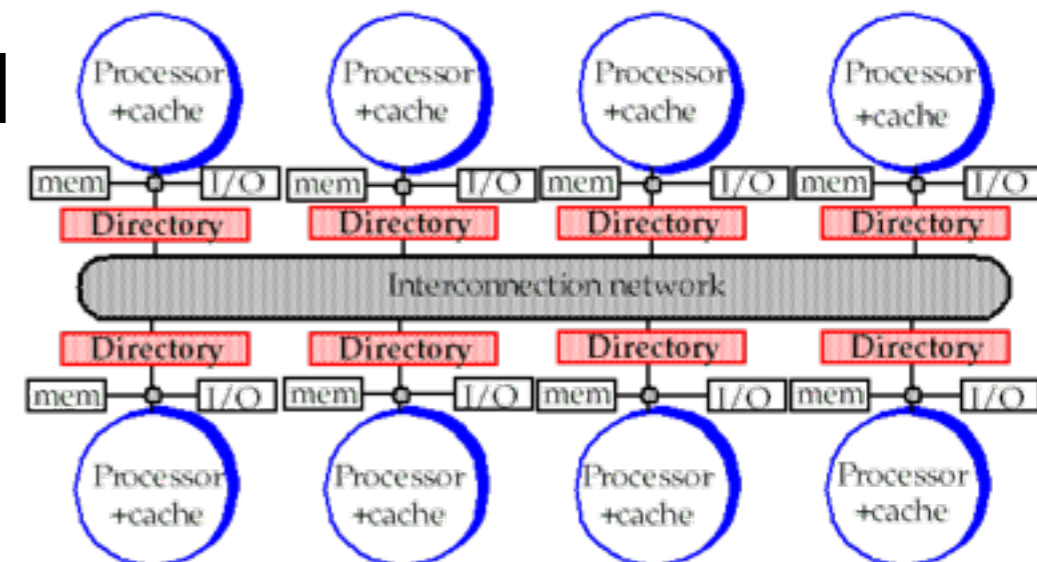
Challenges of Selective Replication

- How to track the most popular $O(n \log n)$ objects?
 - Object popularity changes constantly
- How to manage the replica set?
- How to route requests to the least loaded server?
- How to ensure consistency?

Our Solution:
In-Network Coherence Directory

What is a coherence directory?

- Widely used in multi-processor architectures and distributed shared memory systems
- Tracks state of each cache block and location of shared copies
- Coordinates coherence protocol



Coherence directory applies nicely to selective replication



READ D2

Coherence Directory

Obj ID

Replica Set

can be implemented **efficiently** on programmable switches

D2

S0

S2

ard to
S2

Implementing coherence directory in the network



READ D2

Lookup
Table

Match	pkt.obj = A1	pkt.obj = B4	pkt.obj = D2
Action	index = 0	index = 1	index = 2

0 1 2

S1	S2	S0
----	----	----

Reg Array 0

Replica
Set

	S1	S2
--	----	----

Reg Array 1

	S0	
--	----	--

Reg Array 2

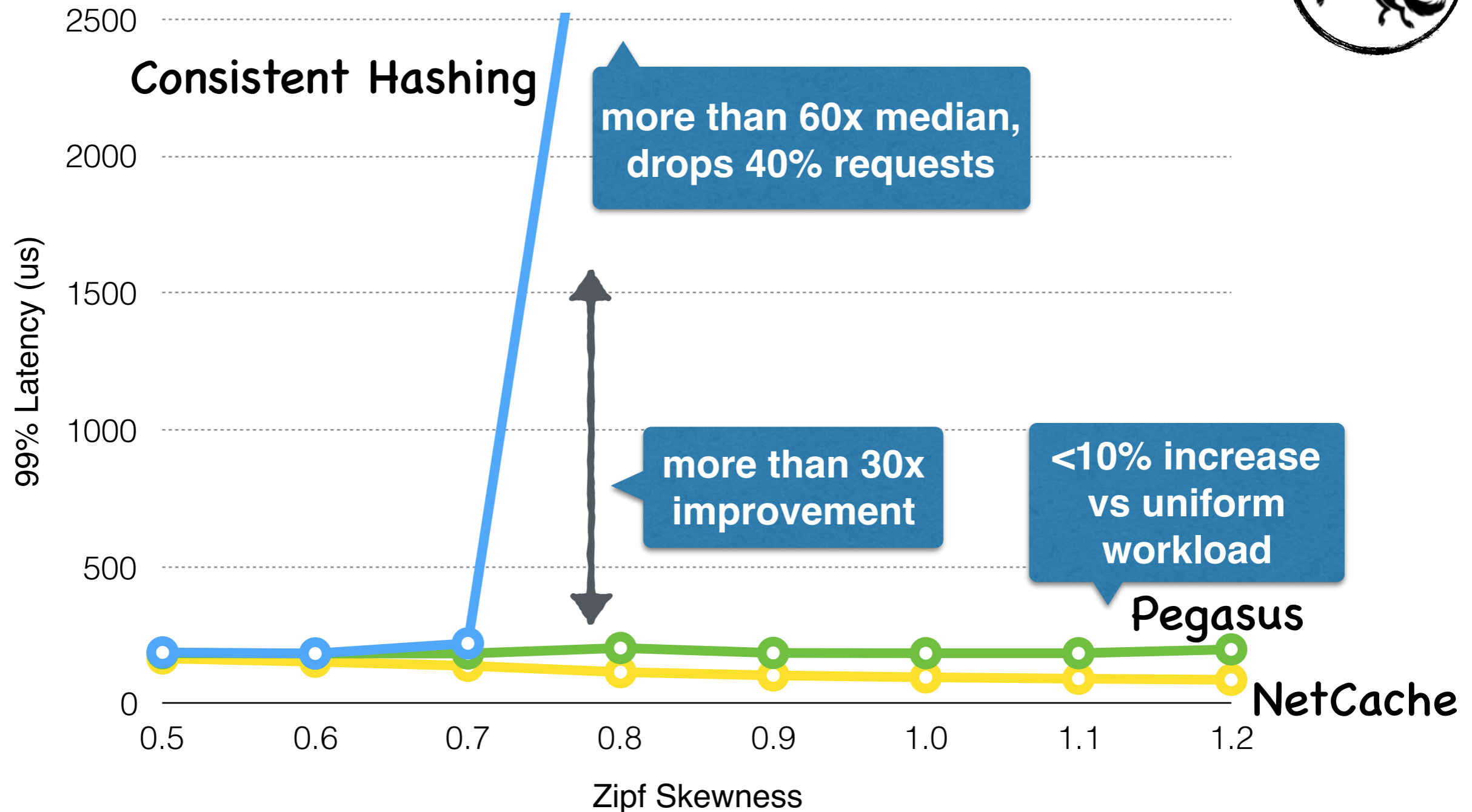
Forwarding
Table

Match	server ID = S0	server ID = S1	server ID = S2
Action	pkt.dst = 10.0.0.1	pkt.dst = 10.0.0.5	pkt.dst = 10.0.0.7

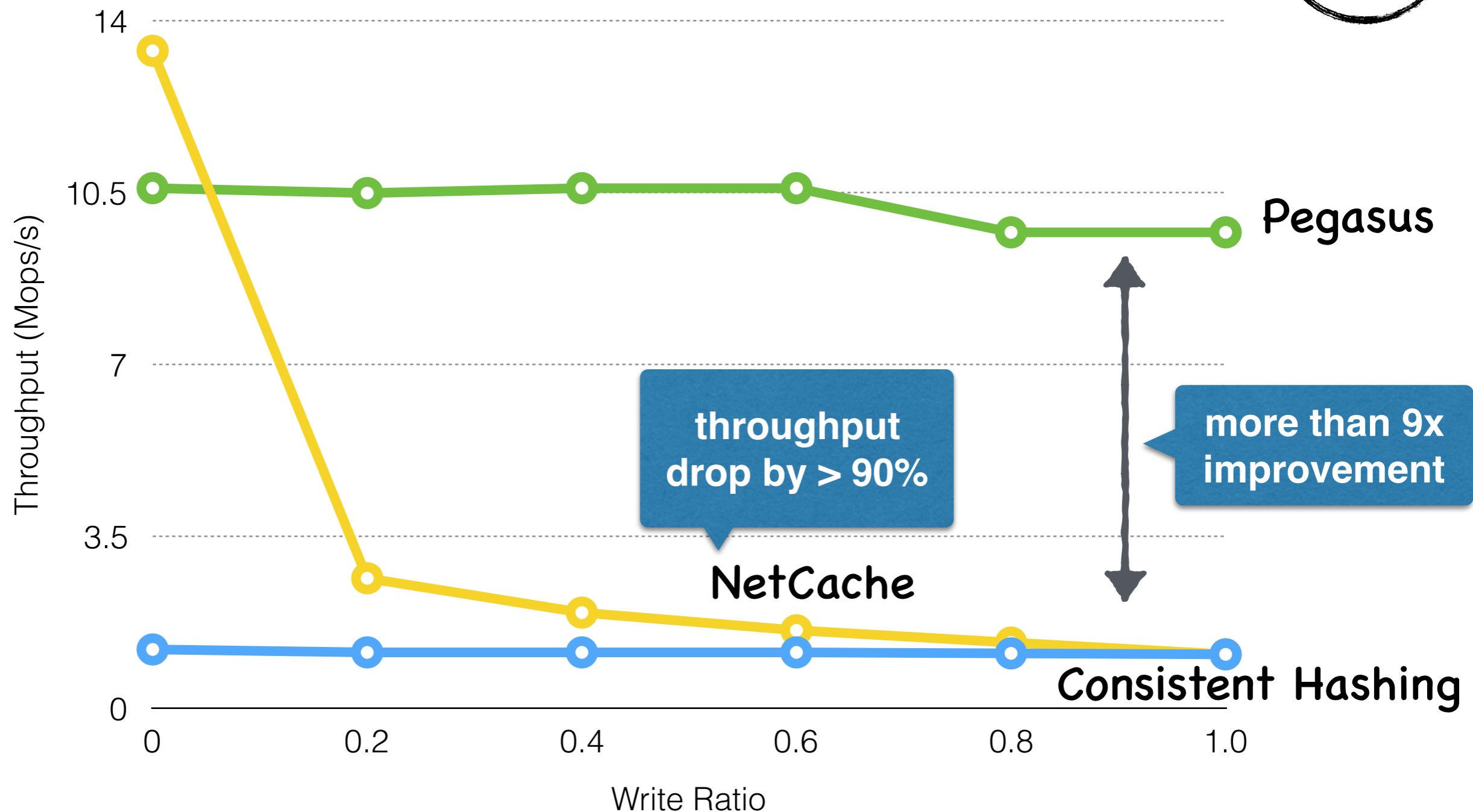
Challenges of In-Network Coherence Directory

- Track which server has the **minimum load**
 - Switch-based load prediction
- Track most popular $O(n \log n)$ objects
- Balance load for **writes**
 - Dynamic replica set
- Ensure **strong consistency**
 - Version-based coherence protocol

Load balancing under highly skewed workloads



Load balancing under different read/write ratios



Summary

- Use programmable switch to improve load balancing of storage systems
- Selectively replicate the most popular objects
- Build in-network coherence directories to manage replicated objects
- 9x throughput improvement compared to consistent hashing