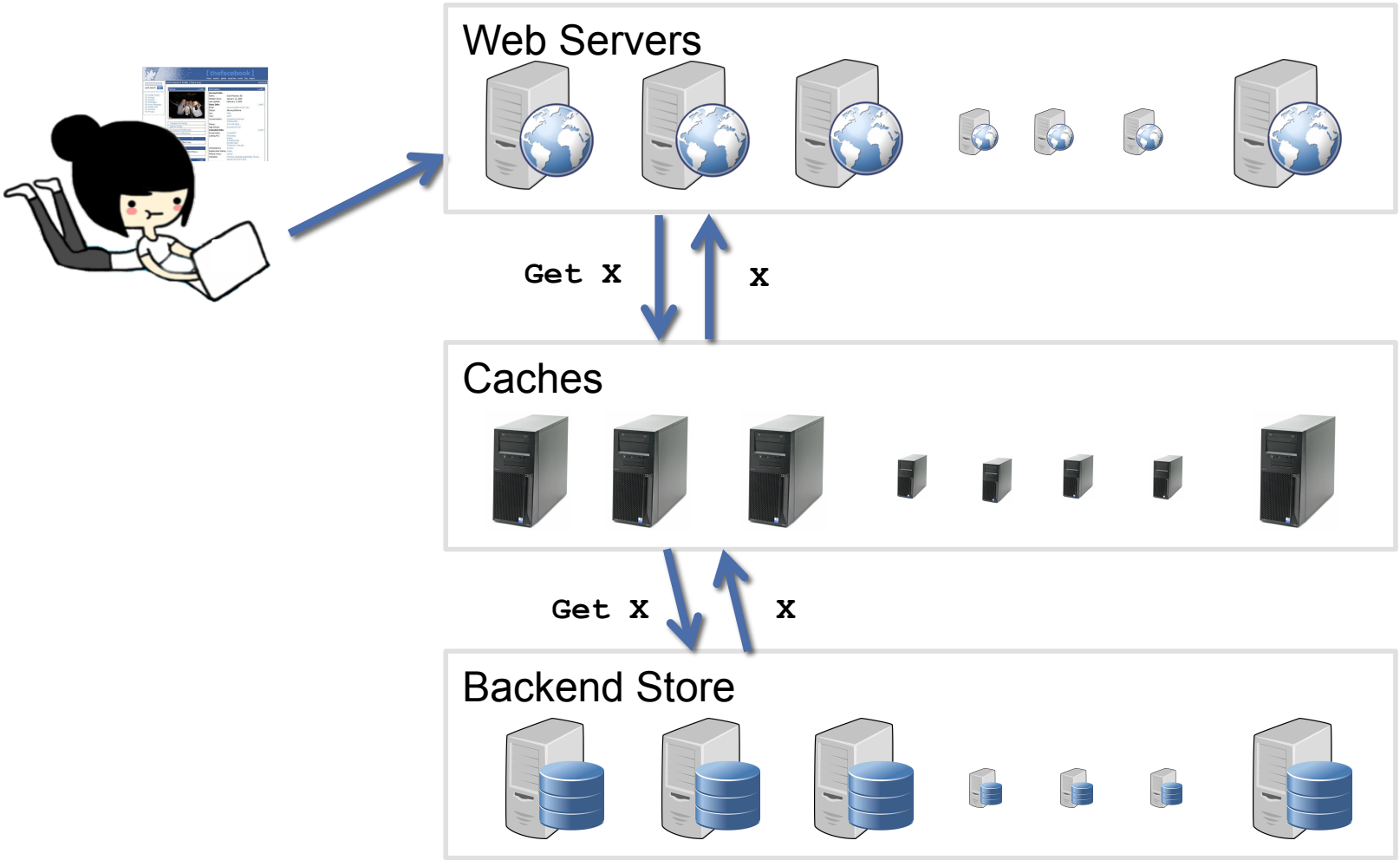
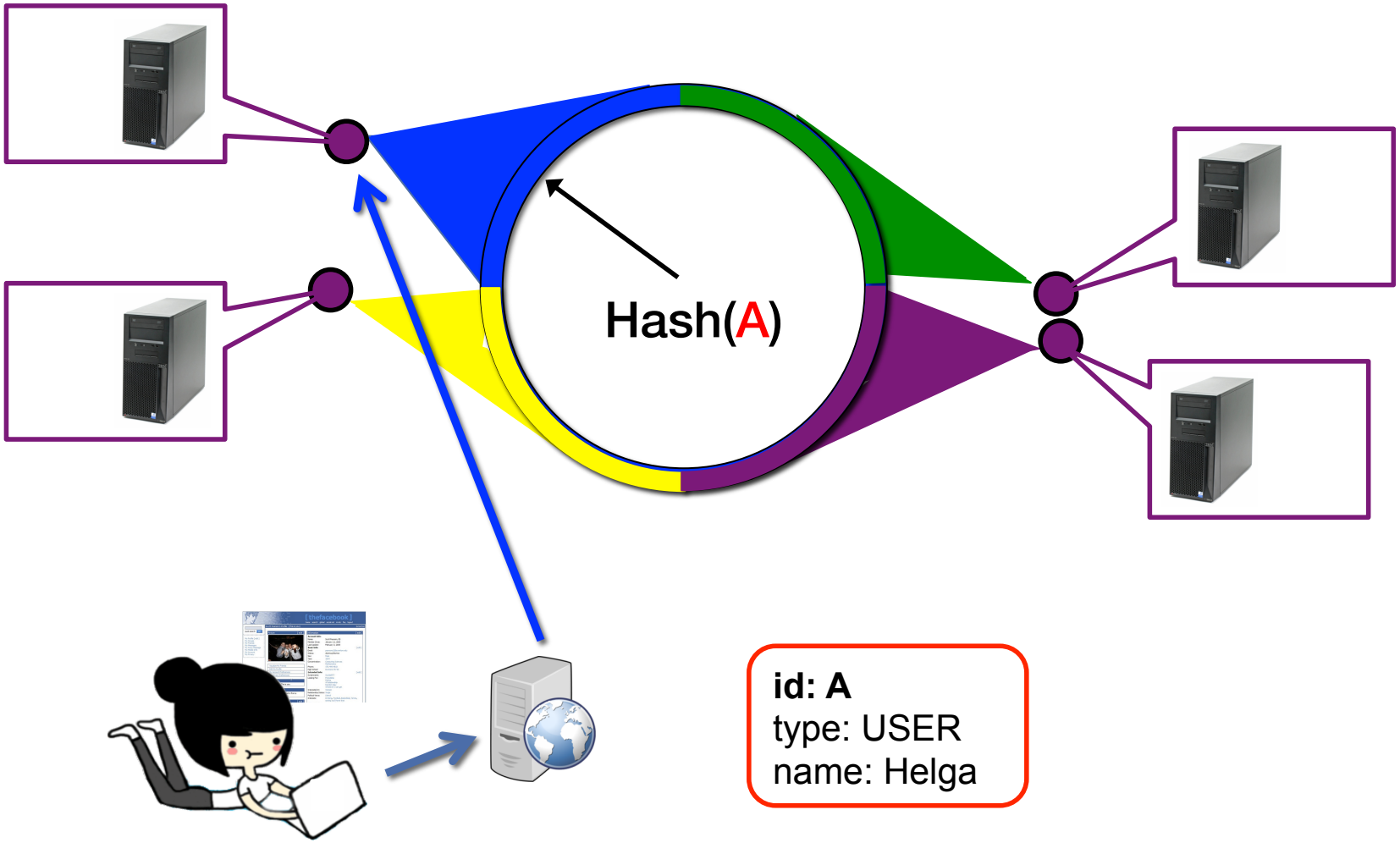
Characterizing Load Imbalance in Real-World Networked Caches

Qi Huang	Cornell U, Facebook
Helga Gudmundsdottir	**Emory U, Reykjavik U**
Ymir Vigfusson	Emory U, Reykjavik U
Daniel A. Freedman	Technion
Ken Birman	Cornell U
Robbert van Renesse	Cornell U

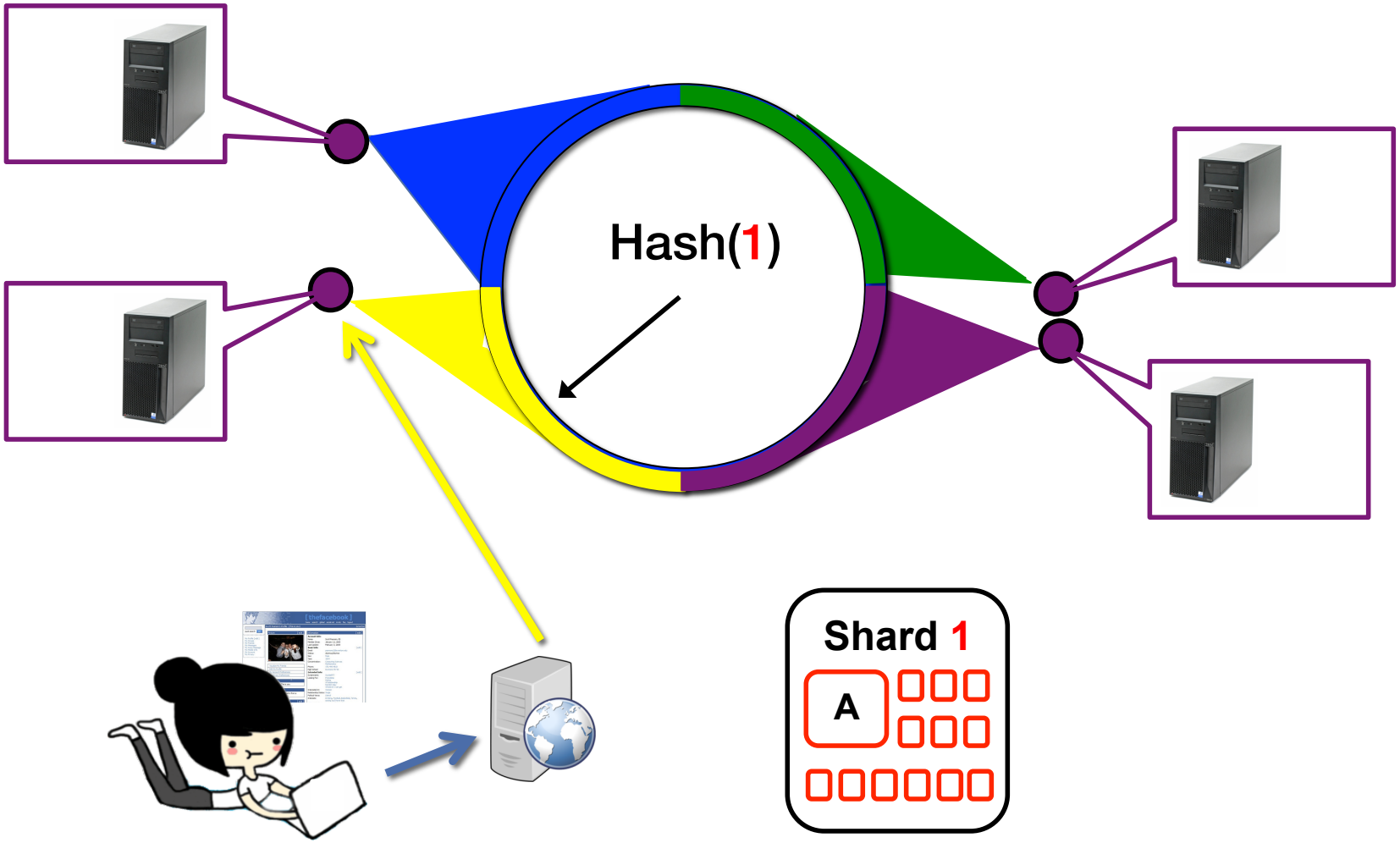
Networked Caches in Real-World Web Stack



Partitioning Data

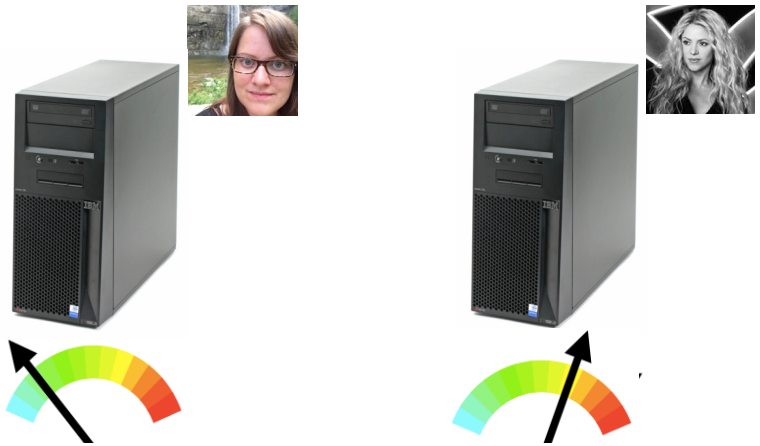


Partitioning Data



Causes of Load Imbalance

- Hashing schemes balance **quantity** of content per server.
- However, content **popularity** varies!
- Need to provision servers to handle peak load.
- Lack of real-world data to verify suspicions.



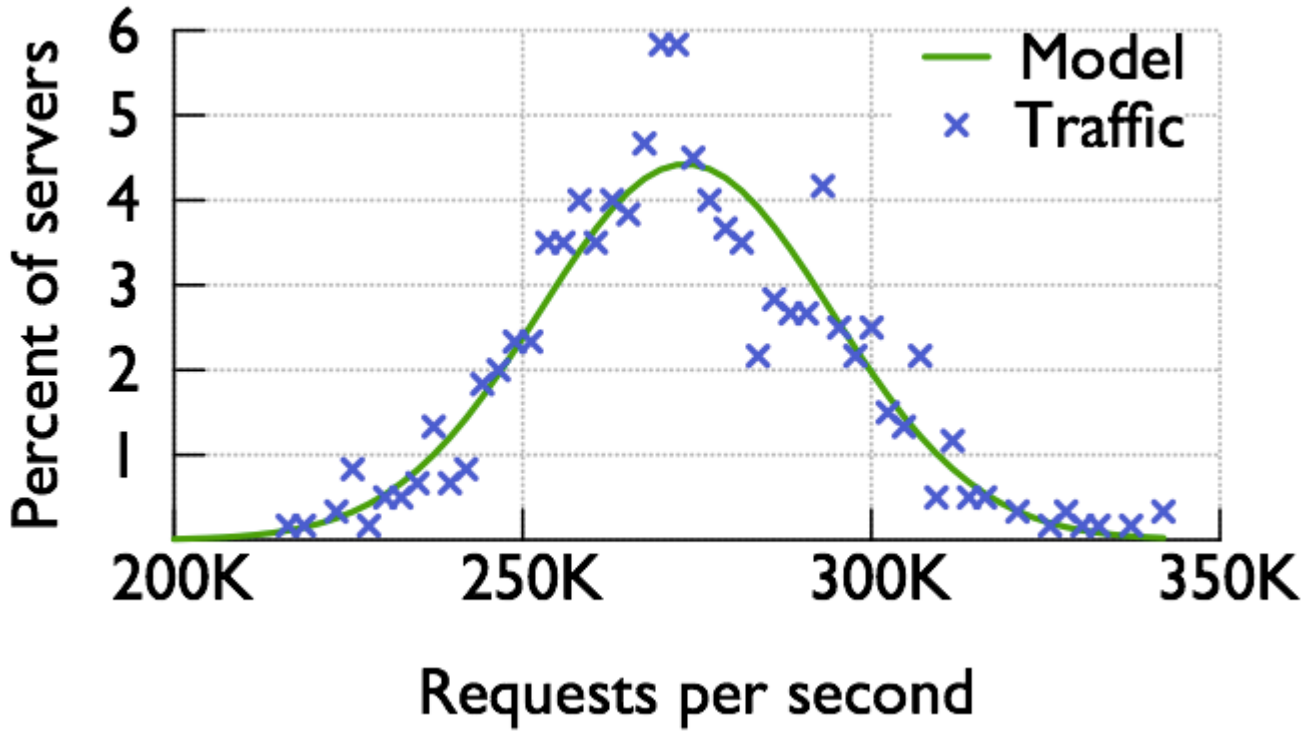
Questions

- What is the state of load imbalance?
- What contributes to load imbalance?

- How effective are current techniques?
- How might they be improved?

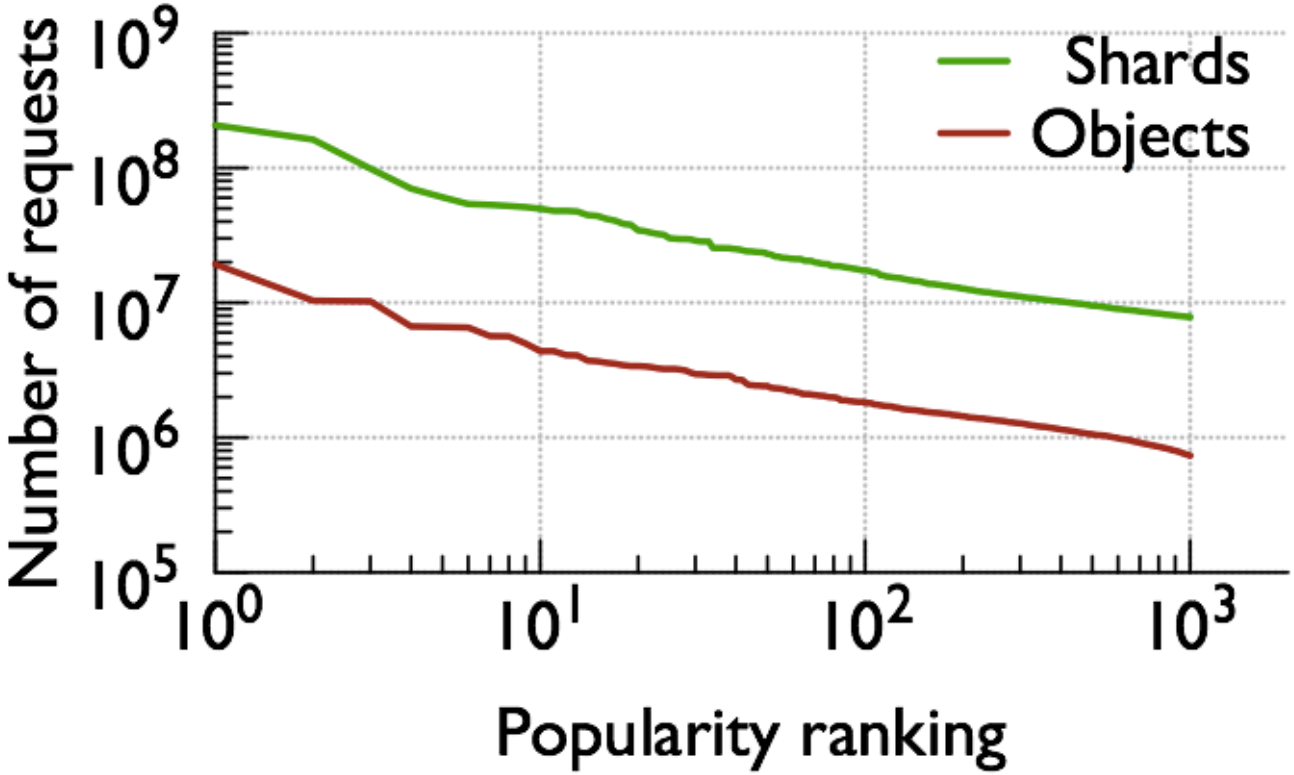
Sampled production traffic in **TAO**, serving Facebook's social graph.

What is the State of Load Imbalance?



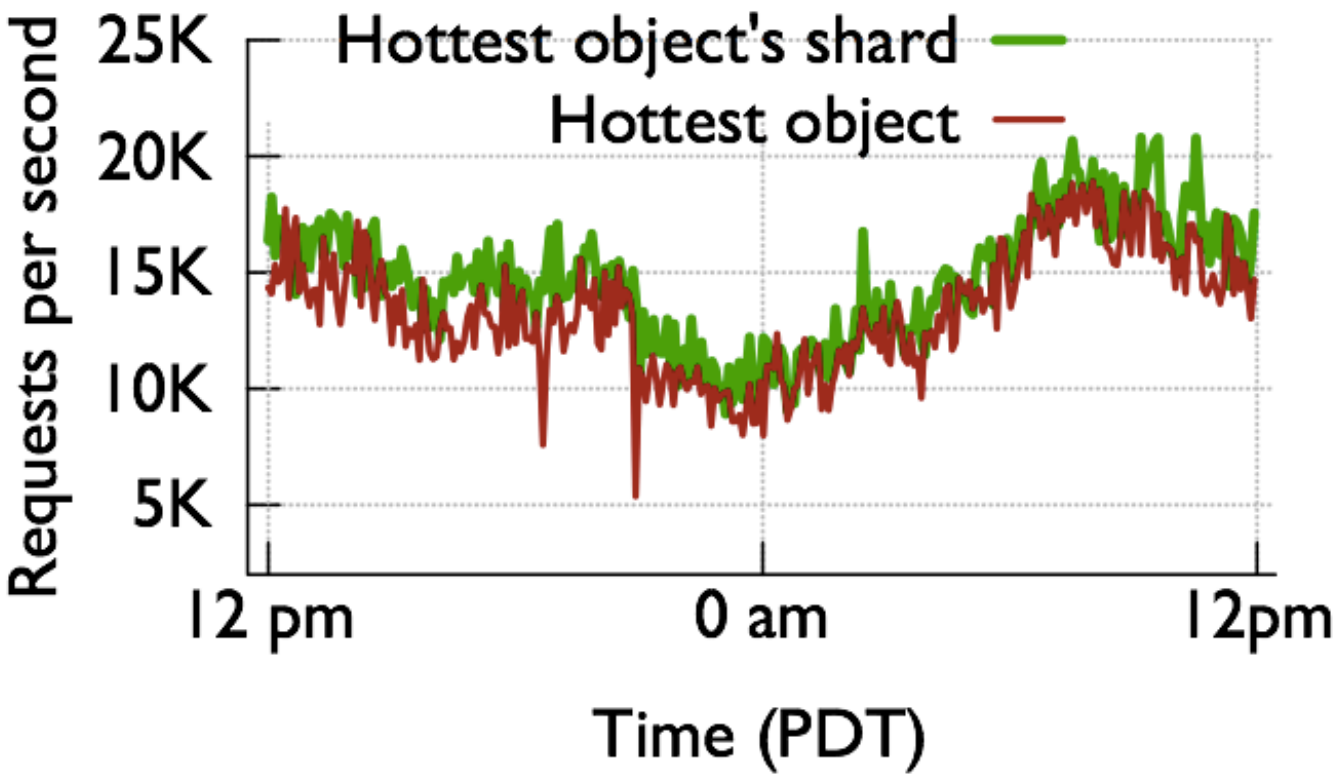
Significant load imbalance observed in TAO.

Possible Causes: Content Popularity



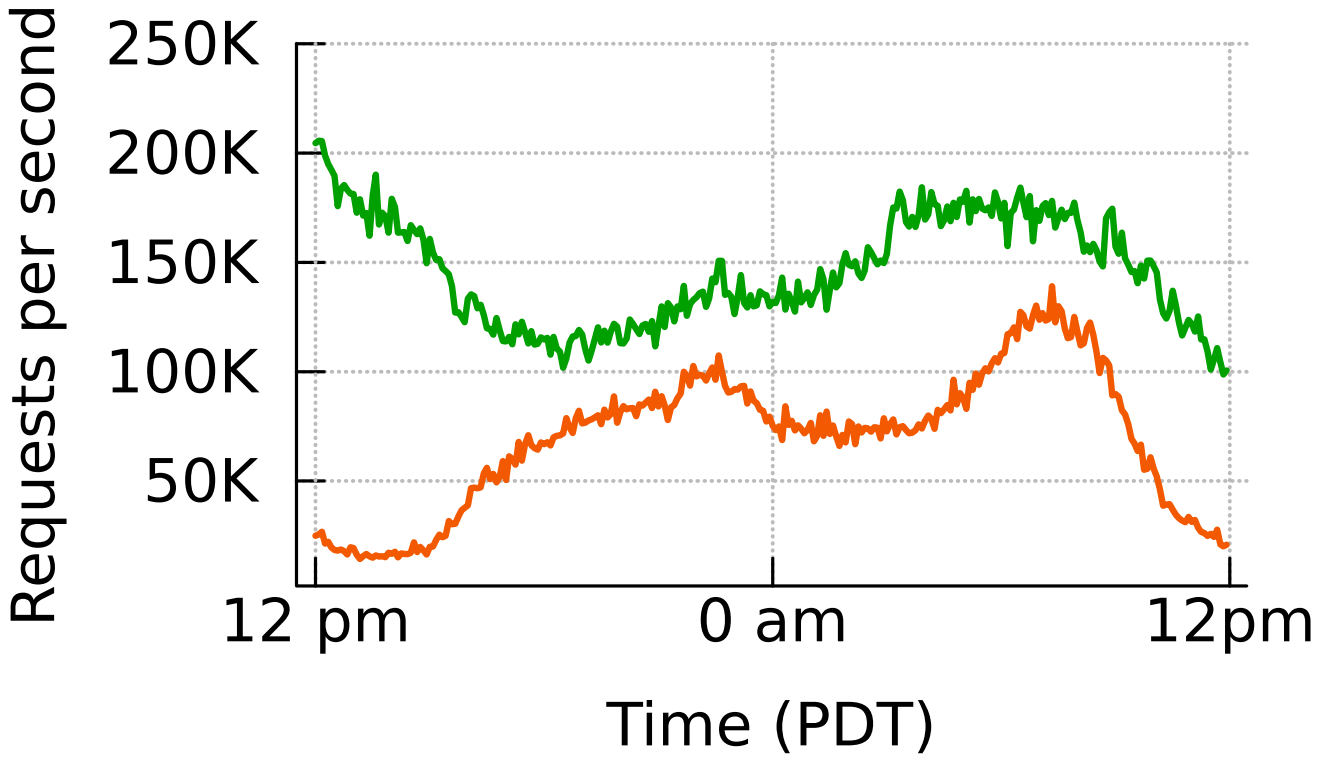
Skewed content popularity observed at both **shard** and **object** level.

Possible Causes: Hot Objects



Very hot objects alone are **not** a major cause of load imbalance.

Possible Causes: Hot Shards



Popular shards receive significantly higher load, compared to hot objects.

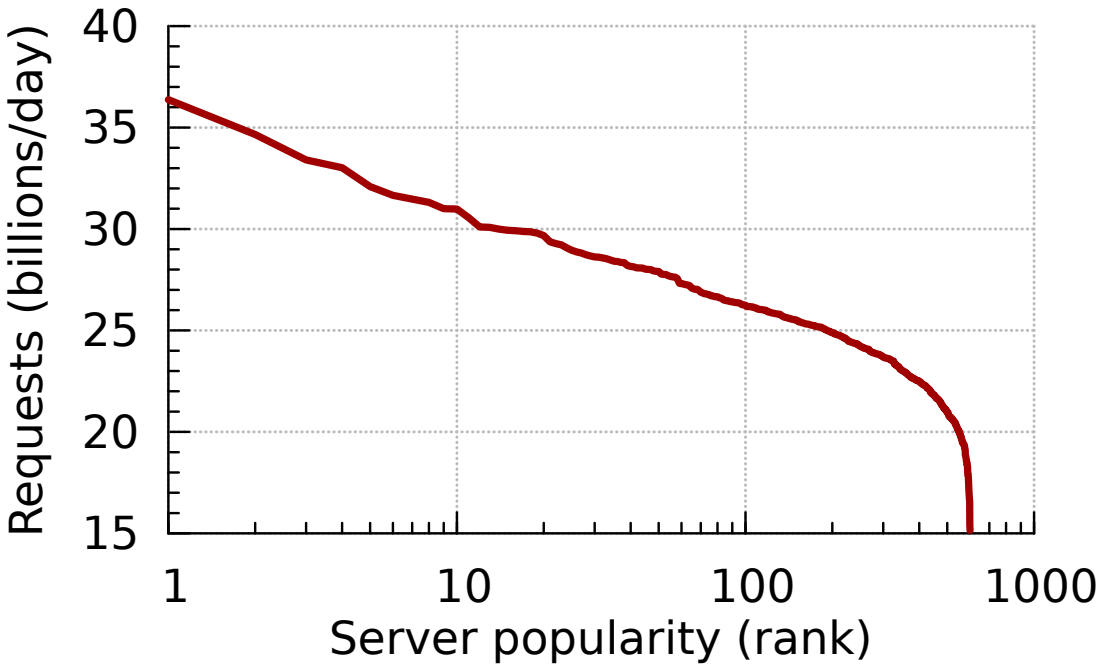
How Effective are Current Techniques?

- **Hashing:** Balance **number** of shards across servers.
- **Replication:** Divide **load** of popular shards among servers.

Replication	Hashing		
	libketama	TAO	Perfect
None			
TAO			
Perfect			

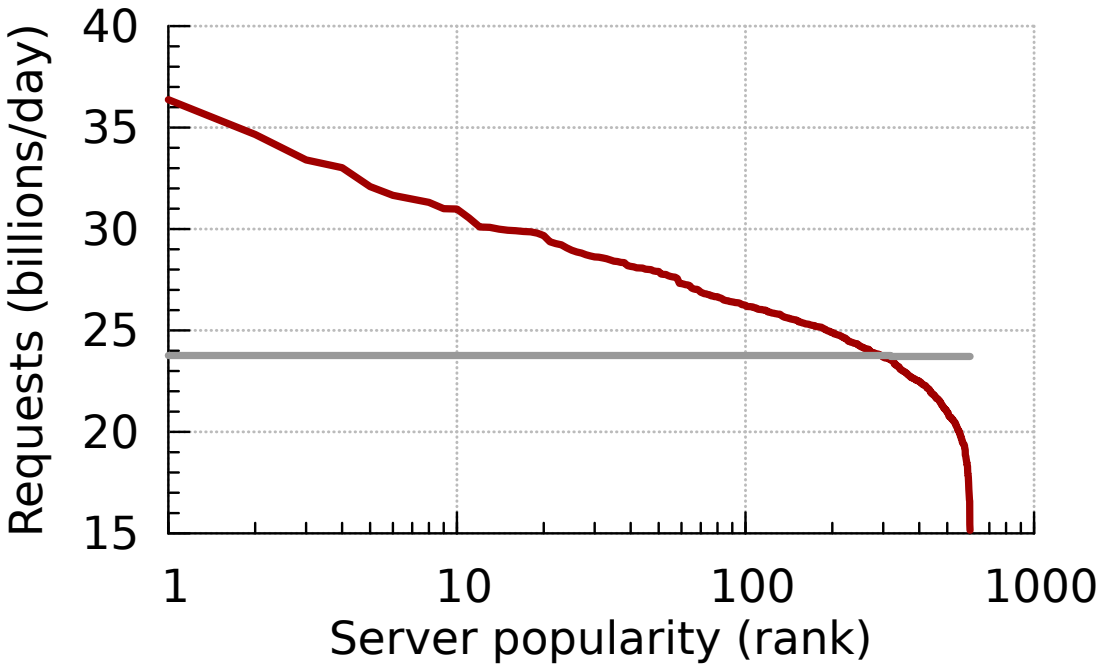
- **Metric: Maximum load** versus **average**.

Where We Are Today



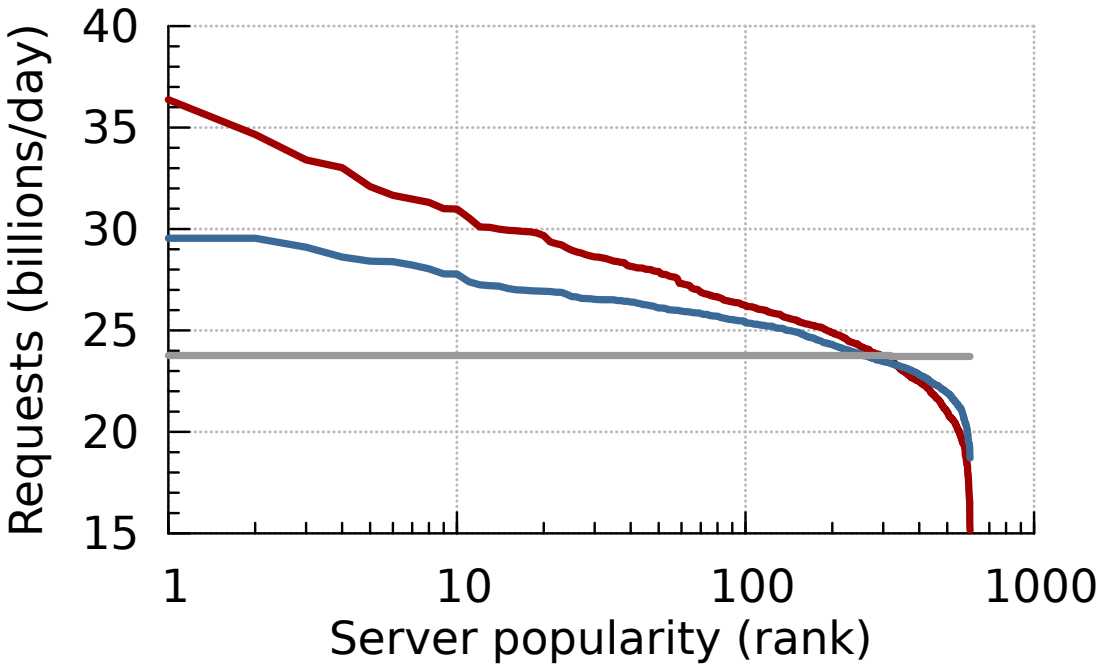
Replication	Hashing		
	libketama	TAO	Perfect
None	**1.53**		
TAO			
Perfect			

Where We Are Today



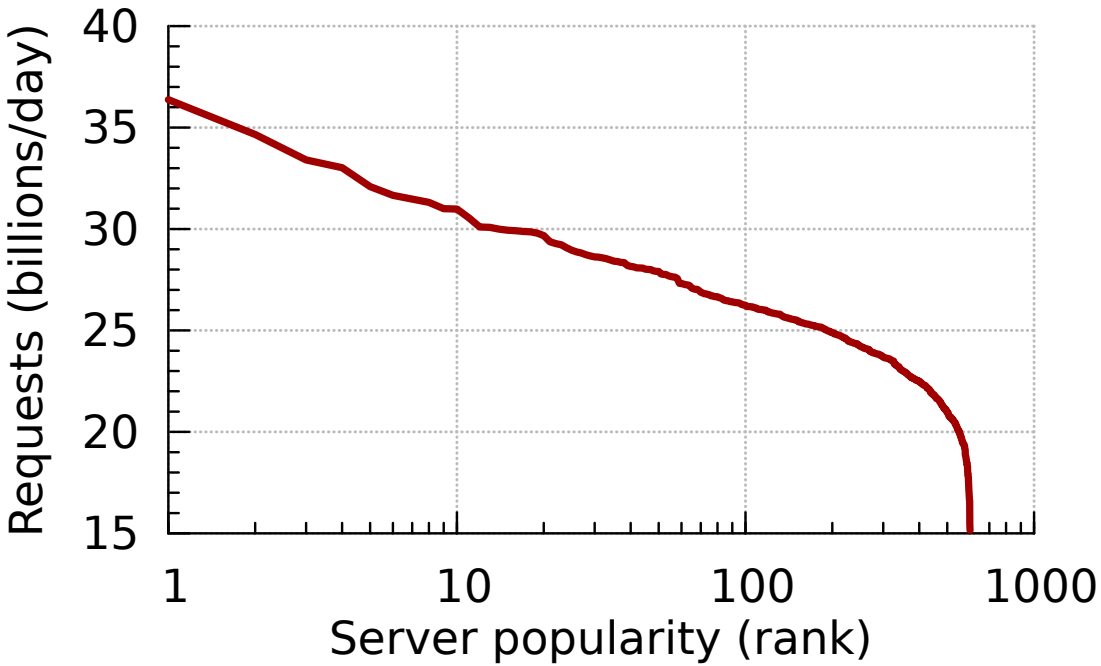
Replication	Hashing		
	libketama	TAO	**Perfect**
None	**1.53**		
TAO			
Perfect			**1.00**

Where We Are Today



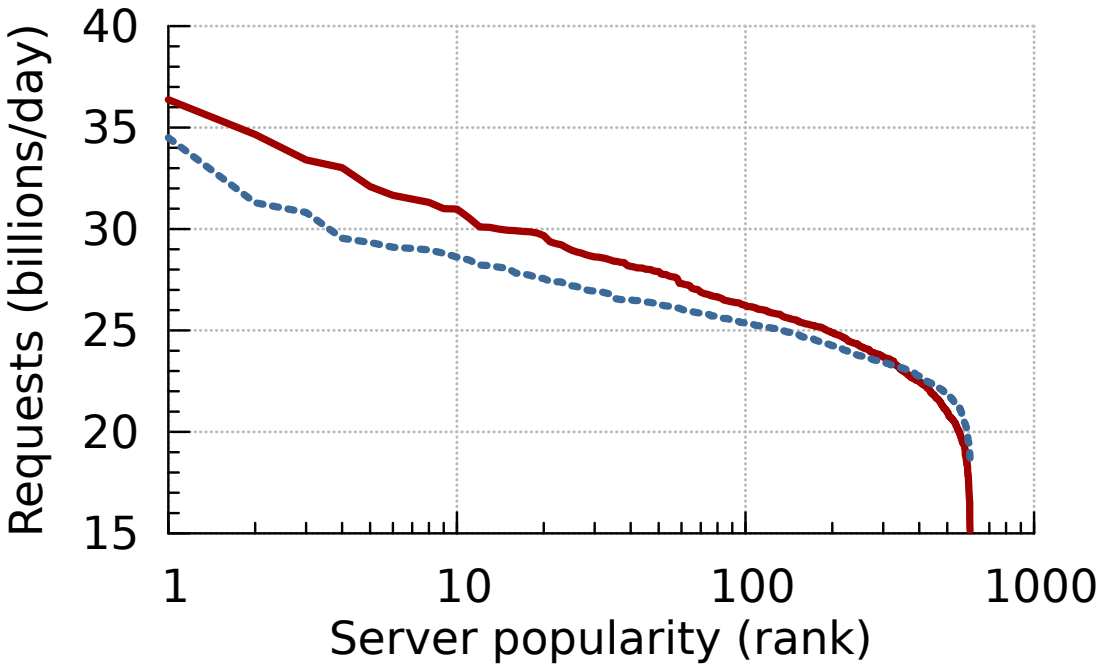
Replication	Hashing		
	libketama	**TAO**	Perfect
None	**1.53**		
TAO		**1.25**	
Perfect			**1.00**

Hashing Schemes w/o Replication



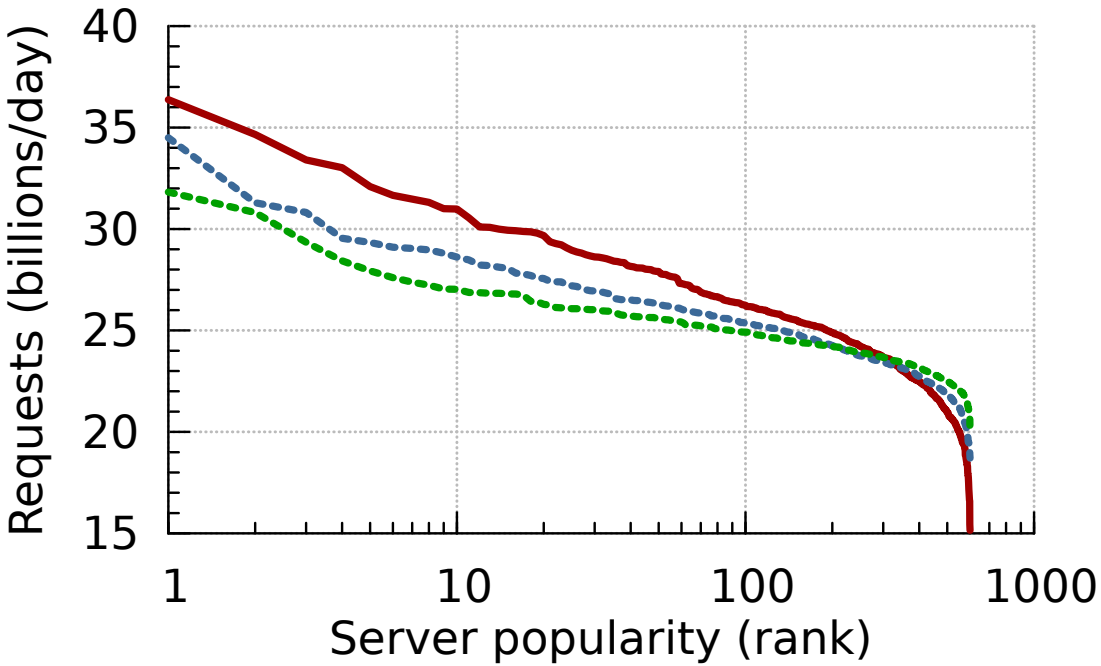
Replication	Hashing		
	libketama	TAO	Perfect
None	**1.53**		
TAO		1.25	
Perfect			1.00

Hashing Schemes w/o Replication



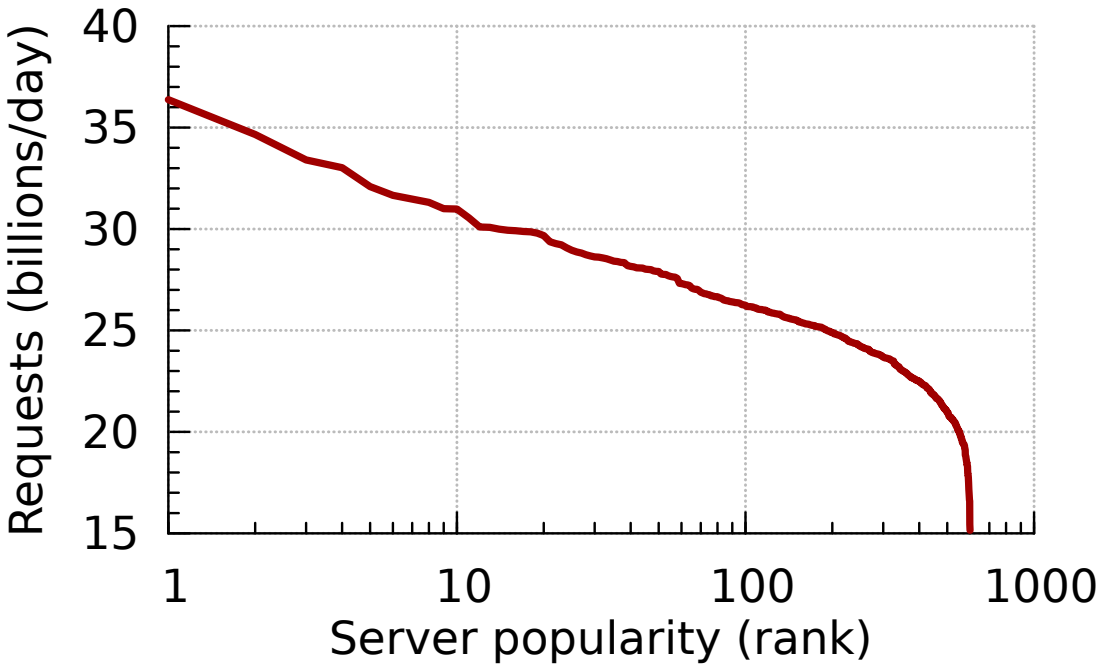
Replication	Hashing		
	libketama	**TAO**	Perfect
None	**1.53**	**1.46**	
TAO		1.25	
Perfect			1.00

Hashing Schemes w/o Replication



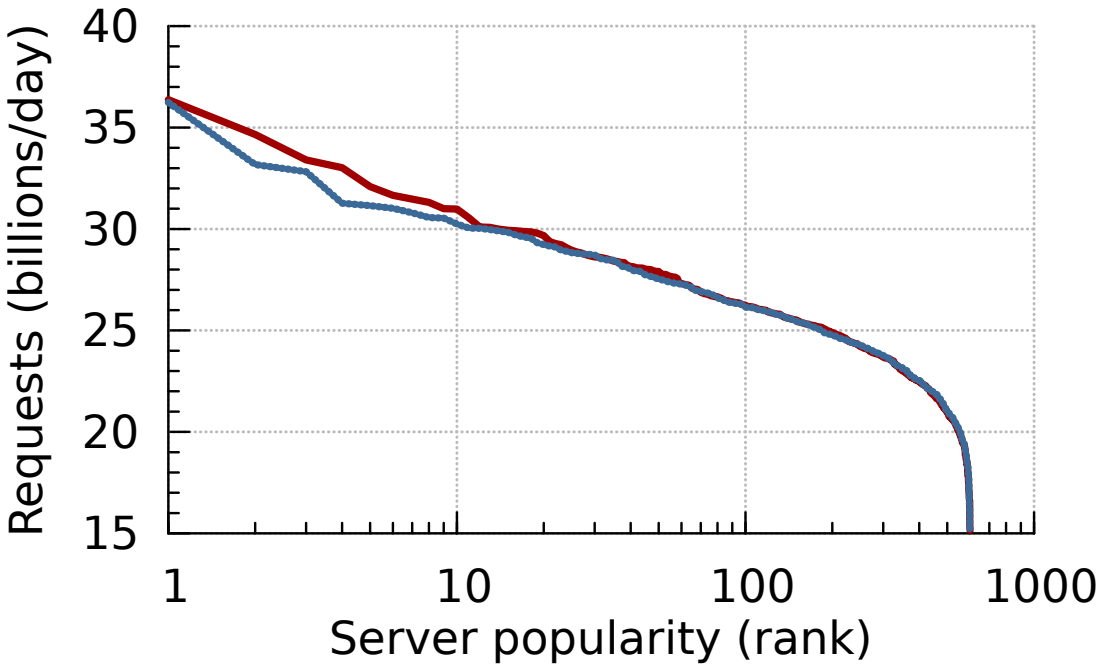
Replication	Hashing		
	libketama	TAO	**Perfect**
None	**1.53**	**1.46**	**1.34**
TAO		1.25	
Perfect			1.00

libketama Hashing



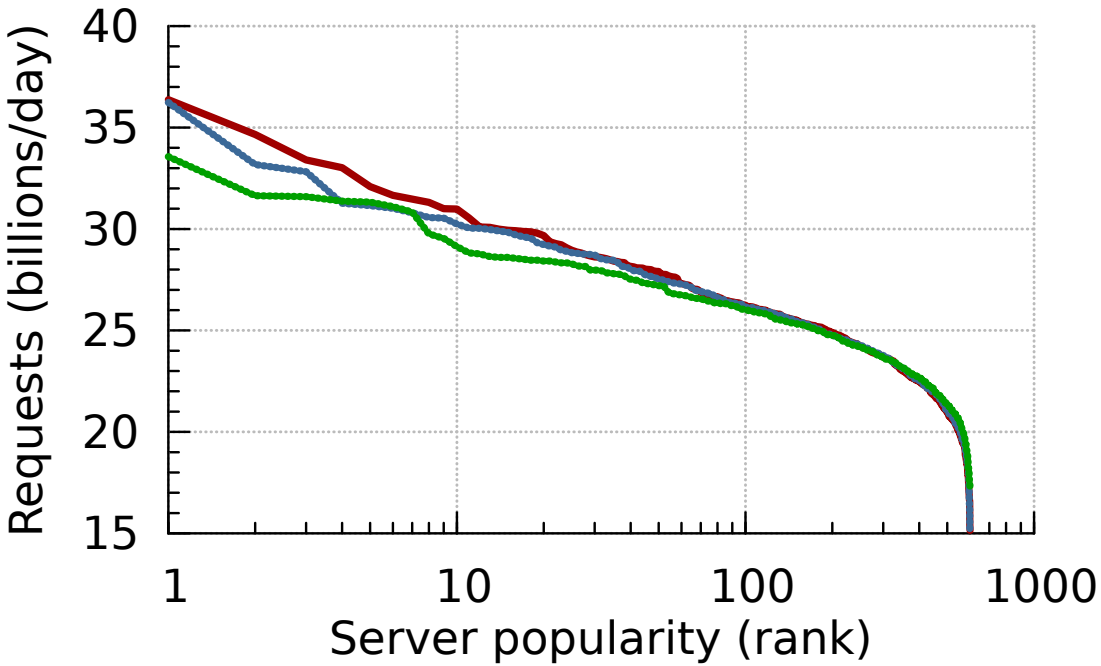
Replication	Hashing		
	libketama	TAO	Perfect
None	**1.53**	1.46	1.34
TAO		1.25	
Perfect			1.00

libketama Hashing



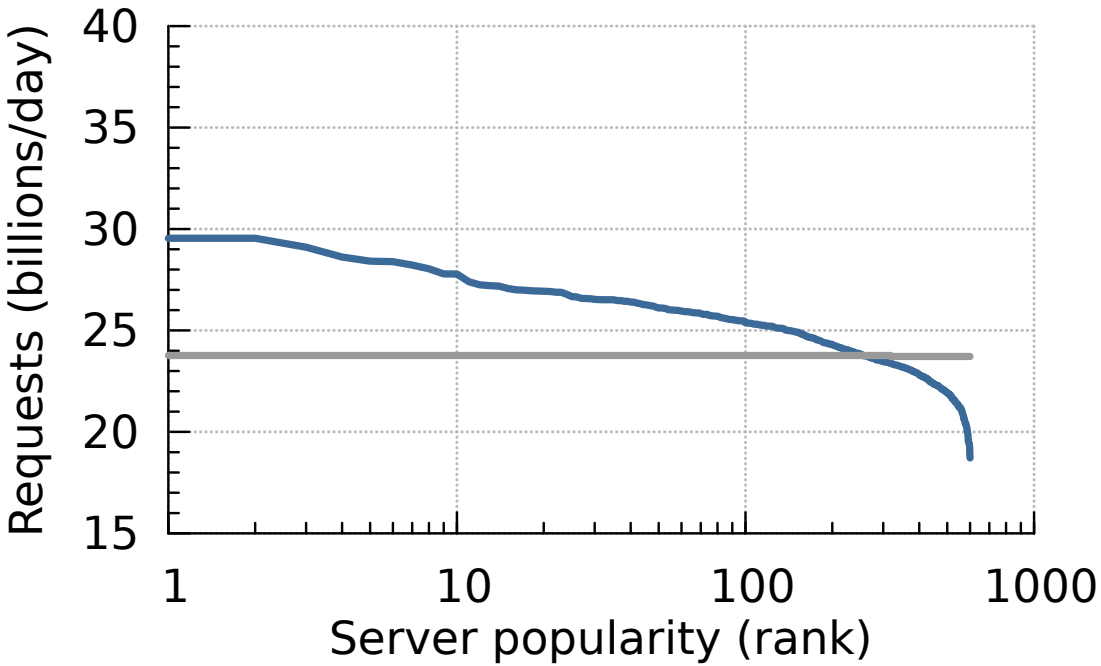
Replication	Hashing		
	libketama	TAO	Perfect
None	**1.53**	1.46	1.34
TAO	**1.53**	1.25	
Perfect			1.00

libketama Hashing



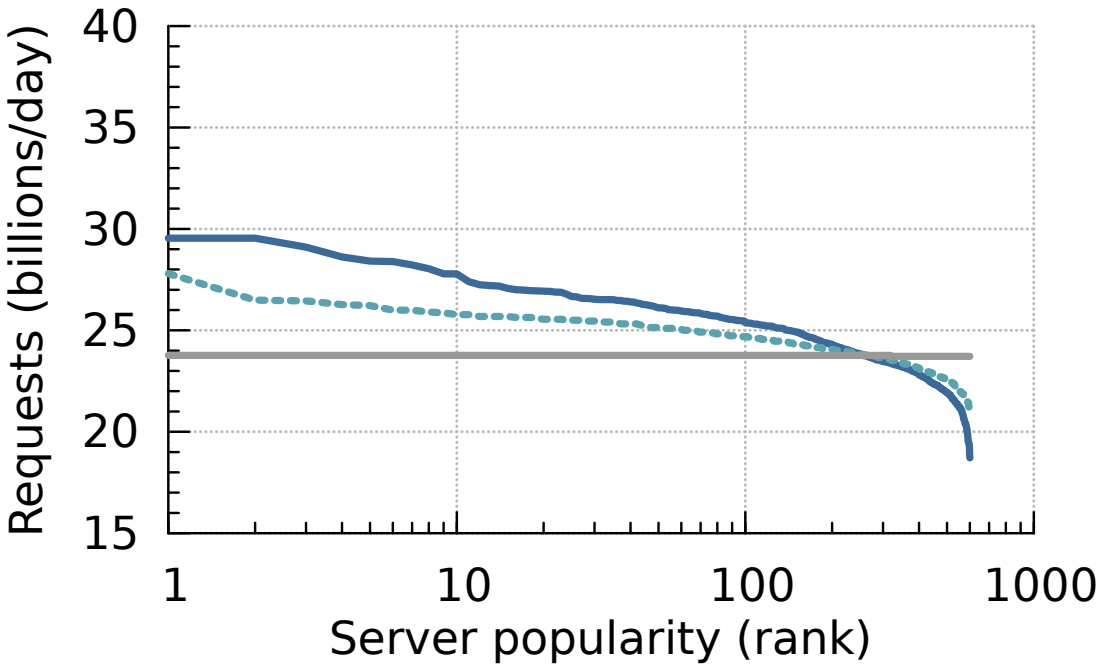
Replication	Hashing		
	libketama	TAO	Perfect
None	**1.53**	1.46	1.34
TAO	**1.53**	1.25	
Perfect	**1.41**		1.00

TAO: Room For Improvement



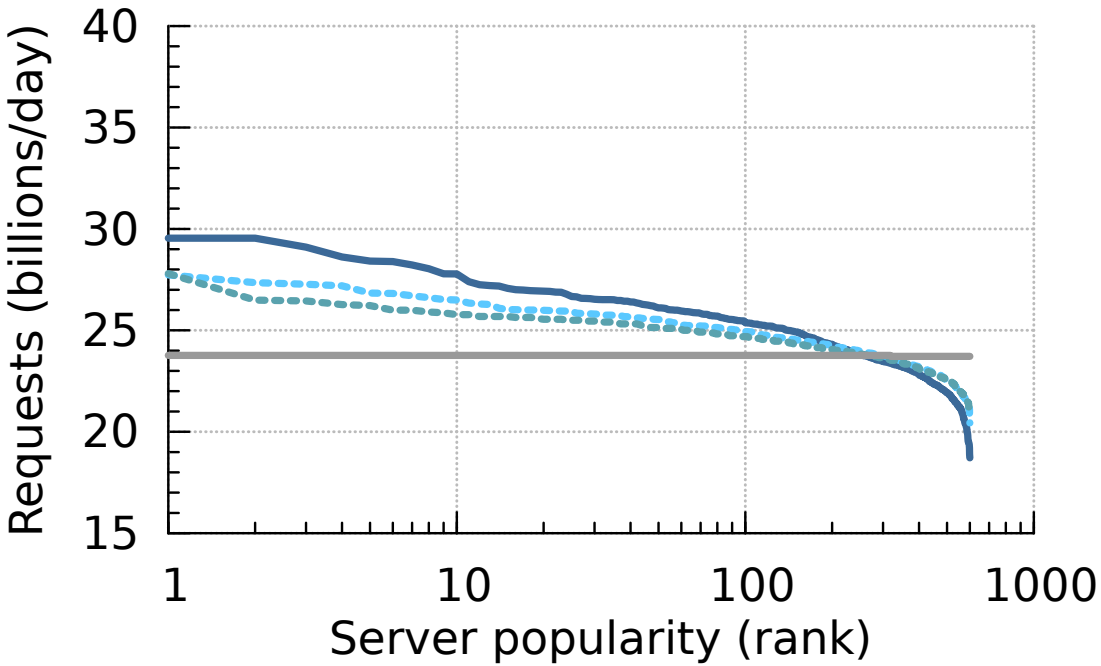
Replication	Hashing		
	libketama	**TAO**	**Perfect**
None	1.53	1.46	1.34
TAO	1.53	**1.25**	
Perfect	1.41		**1.00**

TAO: Room For Improvement



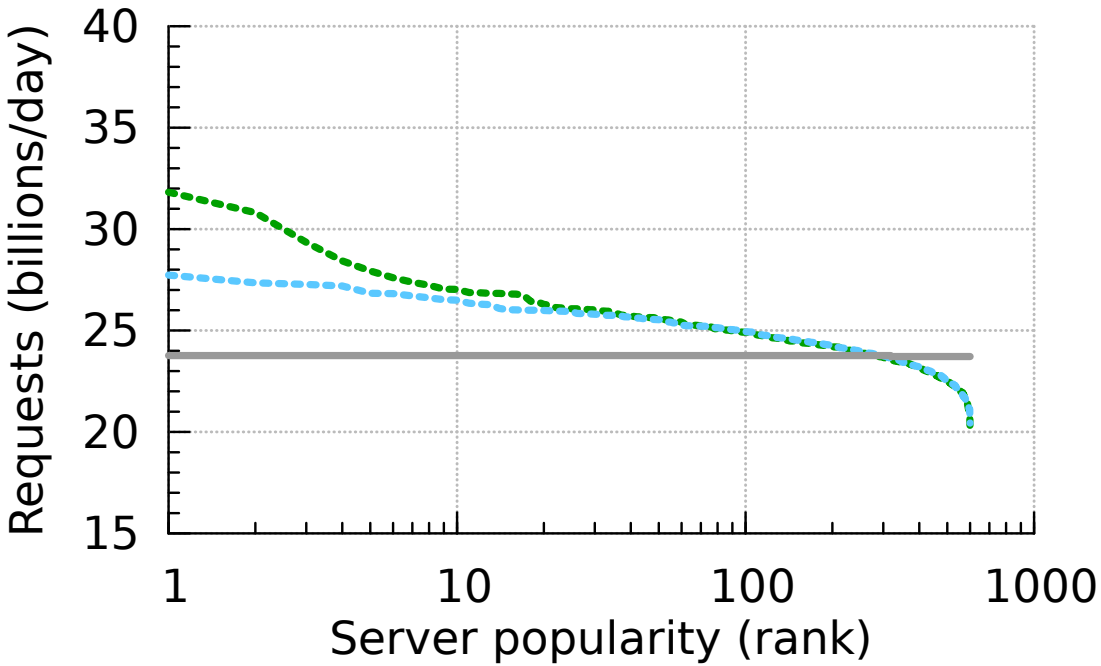
Replication	Hashing		
	libketama	**TAO**	Perfect
None	1.53	1.46	1.34
TAO	1.53	**1.25**	
Perfect	1.41	**1.18**	**1.00**

TAO: Room For Improvement



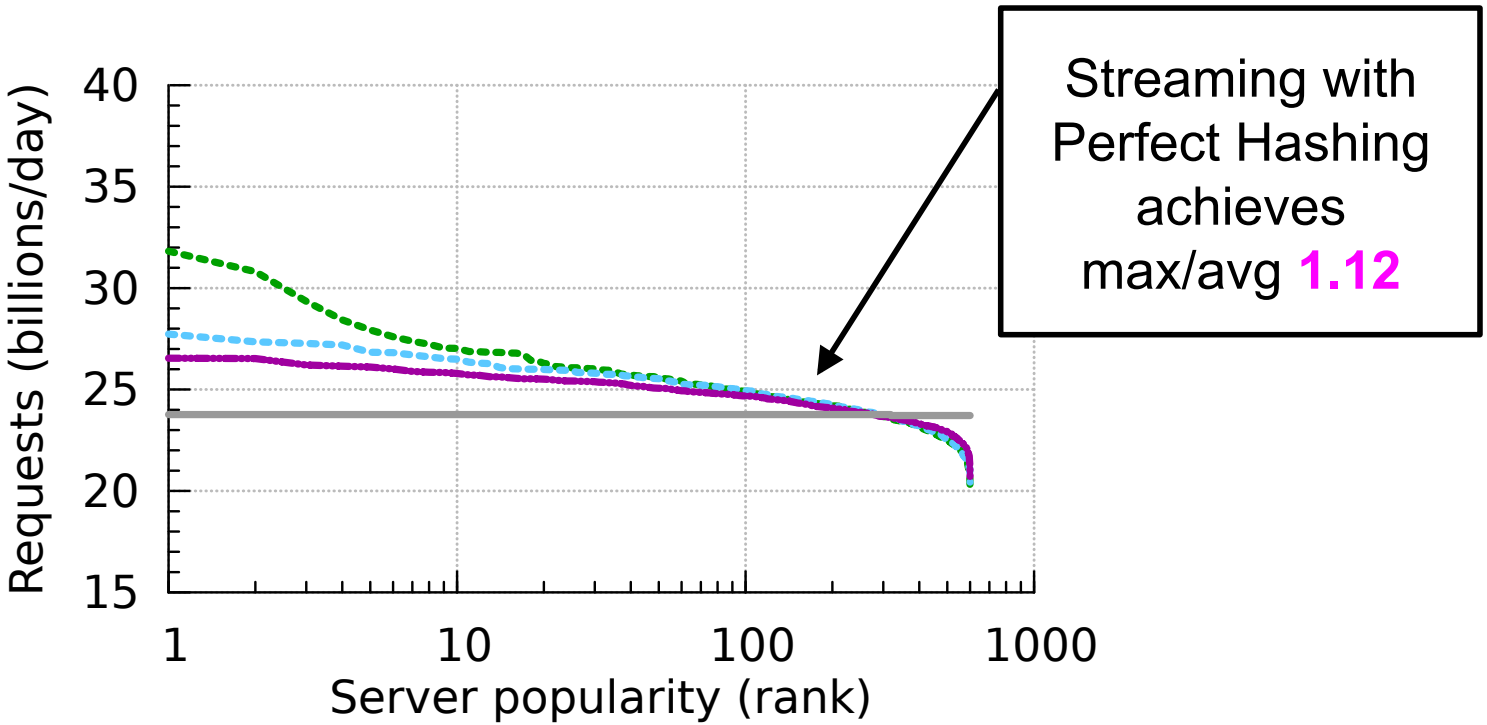
Replication	Hashing		
	libketama	TAO	**Perfect**
None	1.53	1.46	1.34
TAO	1.53	**1.25**	1.17
Perfect	1.41	1.18	**1.00**

Streaming Replication



Replication	Hashing		
	libketama	TAO	**Perfect**
None	1.53	1.46	**1.34**
TAO	1.53	1.25	1.17
Perfect	1.41	1.18	**1.00**

Streaming Replication



Replication	Hashing		
	libketama	TAO	**Perfect**
None	1.53	1.46	**1.34**
TAO	1.53	1.25	1.17
Perfect	1.41	1.18	**1.00**

Summary and Future Work

- Characterized how hashing and replication affect load imbalance.
- Can streaming algorithms replicate content before its popularity surges?
- Can we predict popularity spikes and prevent hotspots?

Questions?