Scaling Distributed Filesystems in Resource-Harvesting Datacenters

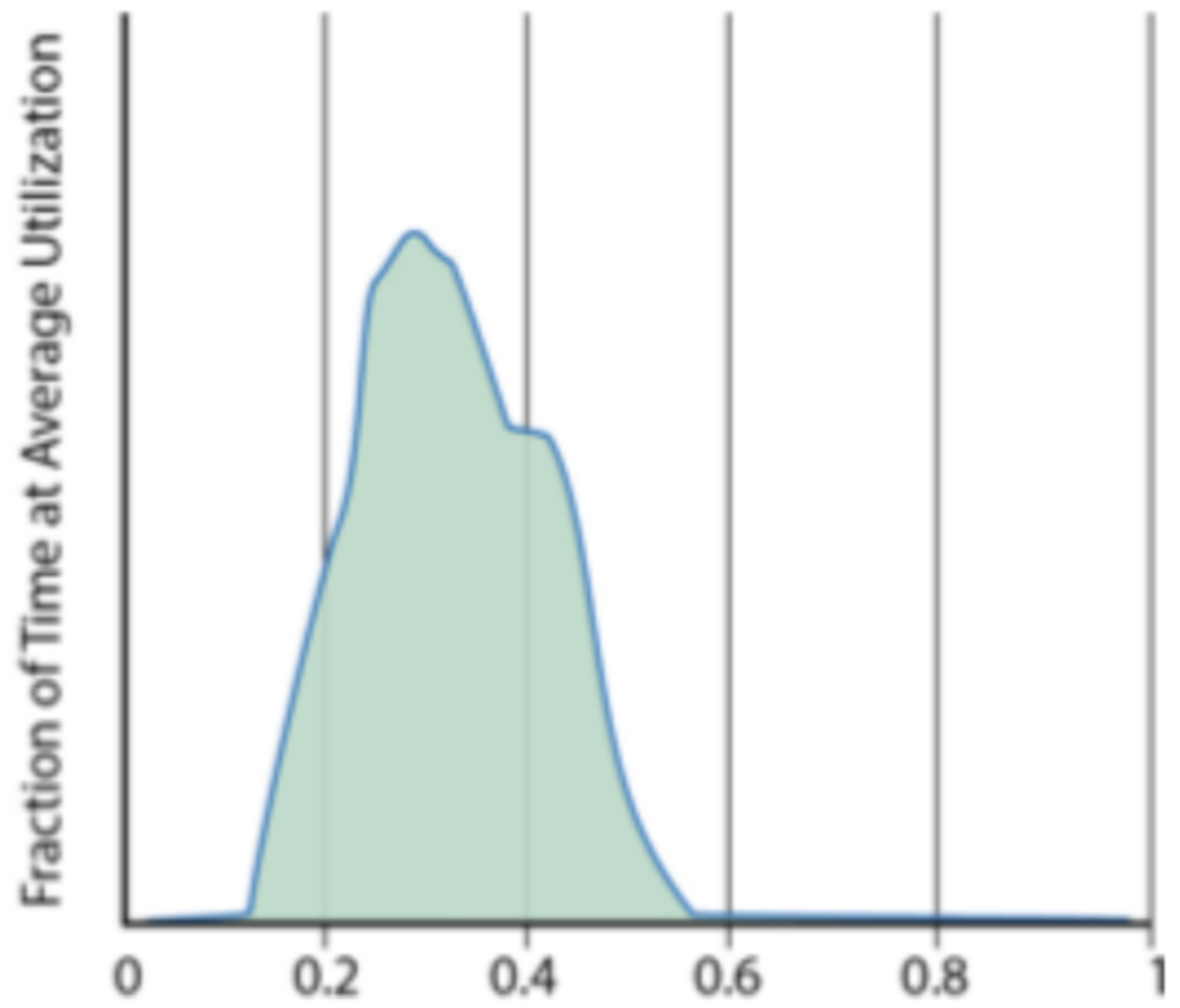
Pulkit A. Misra, Íñigo Goiri, Jason Kace, Ricardo Bianchini

Duke University

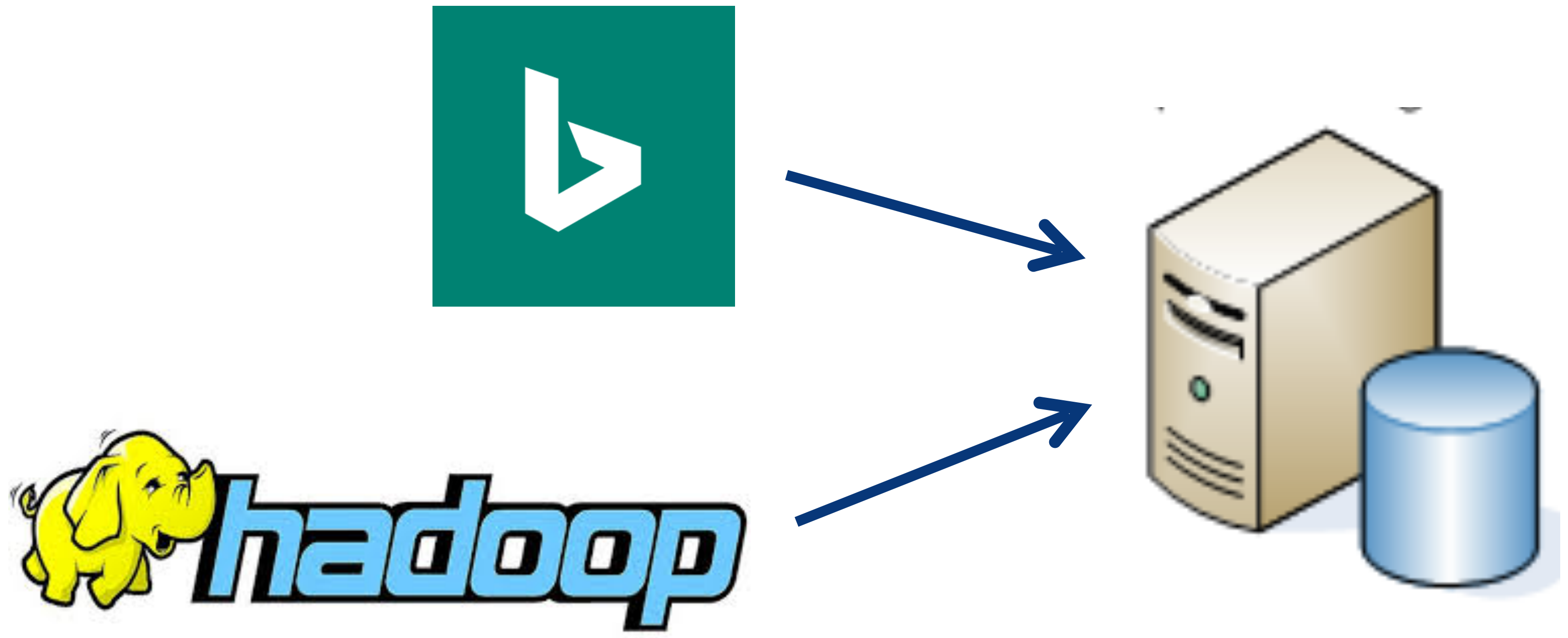
Microsoft Research

Resource-Harvesting Datacenters

- Datacenters are under-utilized
  - Provisioned for peak load, low tail latency
- Harvest spare resources
  - Co-locate services + batch jobs [Zhang, OSDI'16]
- Enable datacenter-wide harvesting
  - Scale distributed file systems

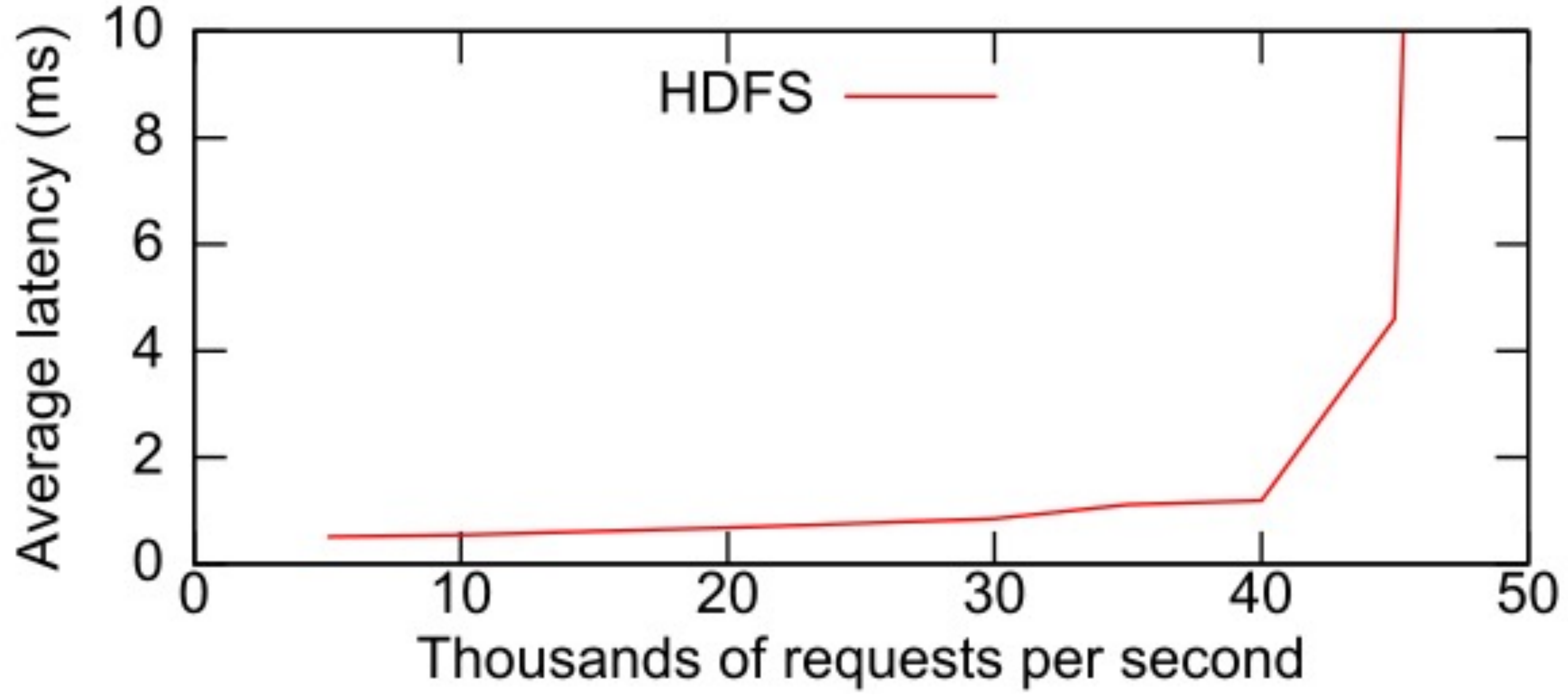


Server utilization distribution of a Google cluster



Scaling Distributed File Systems

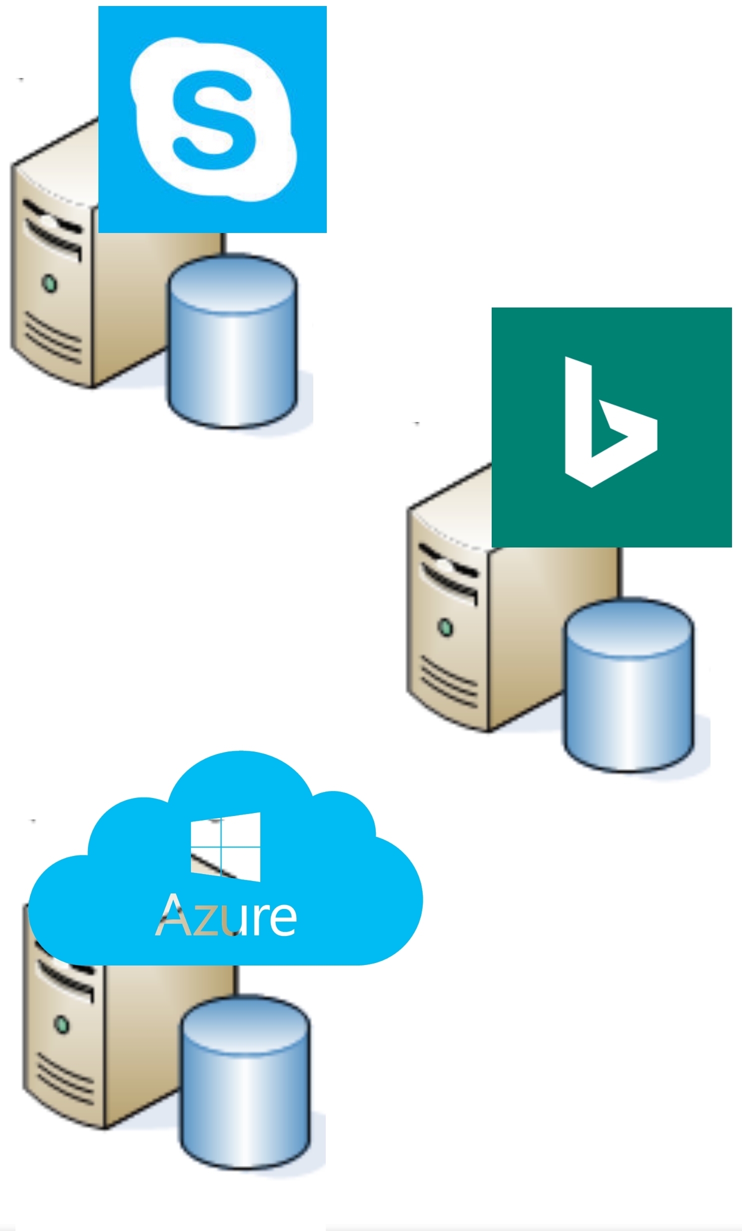
- More storage capacity demands
  - Need bigger file system installations
- Limitations to horizontal scaling
  - Bottleneck at centralized components
- Centralized metadata manager
  - Manages namespace and blocks
  - Simplifies design and maintenance
  - Saturation: 4000 servers, ~40k reqs/sec



Throughput vs Latency

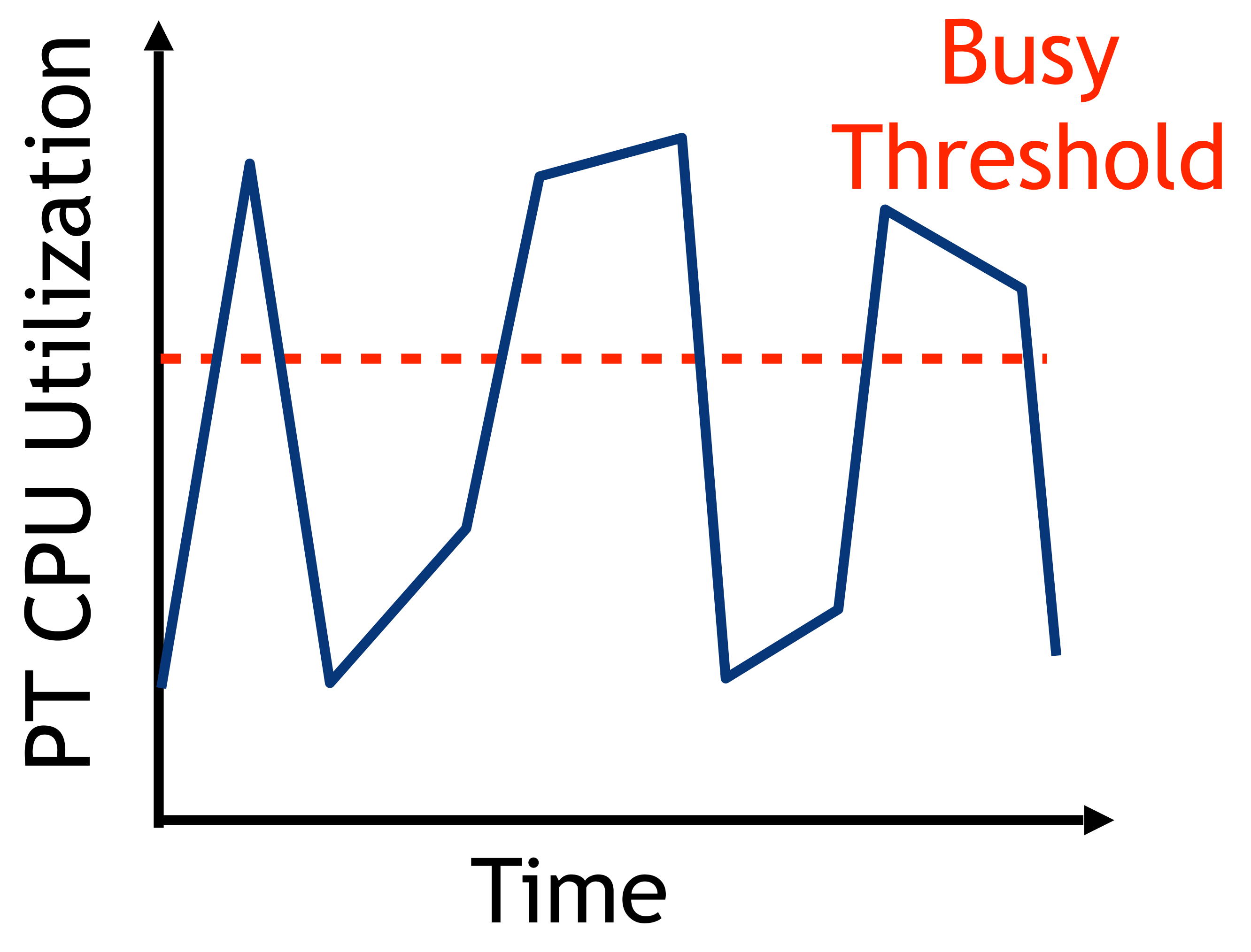
Resource-Harvesting Challenges

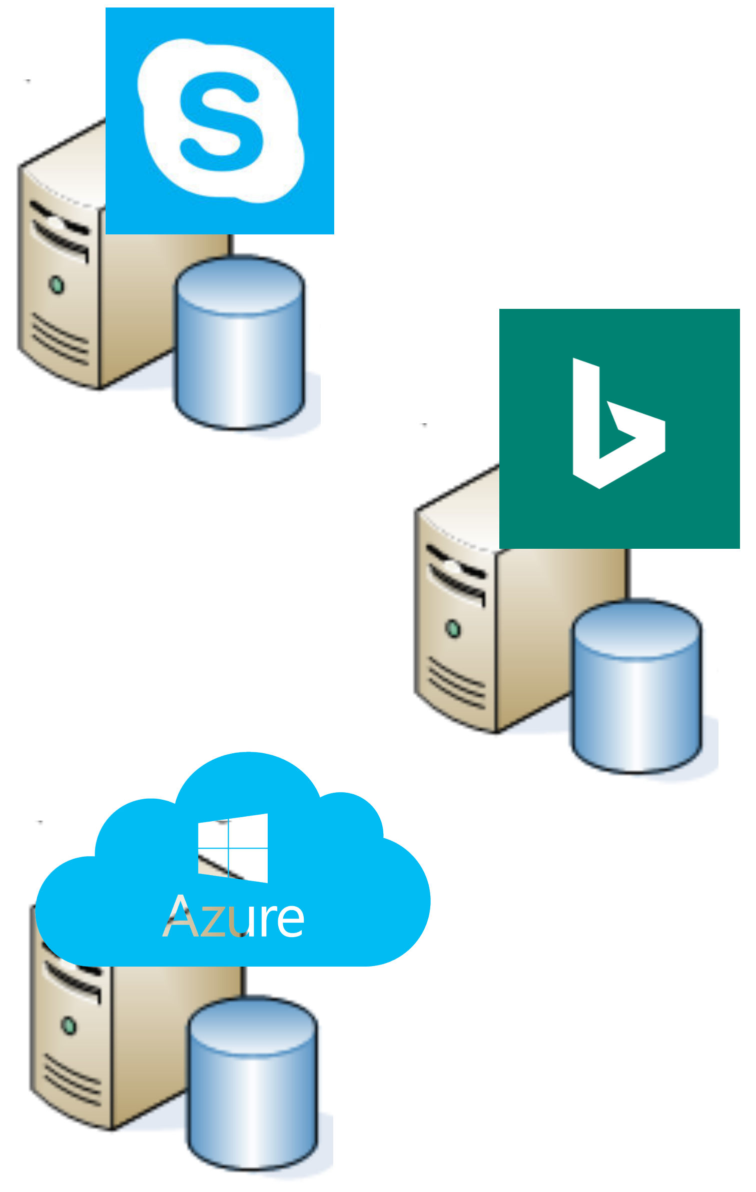
- Primary Tenants (PTs) *own* servers
  - Interactive services (e.g., Bing) are PTs
- Harvest resources from PTs
  - Avoid performance impact to the PT
- Challenges for distributed file systems



Resource-Harvesting Challenges

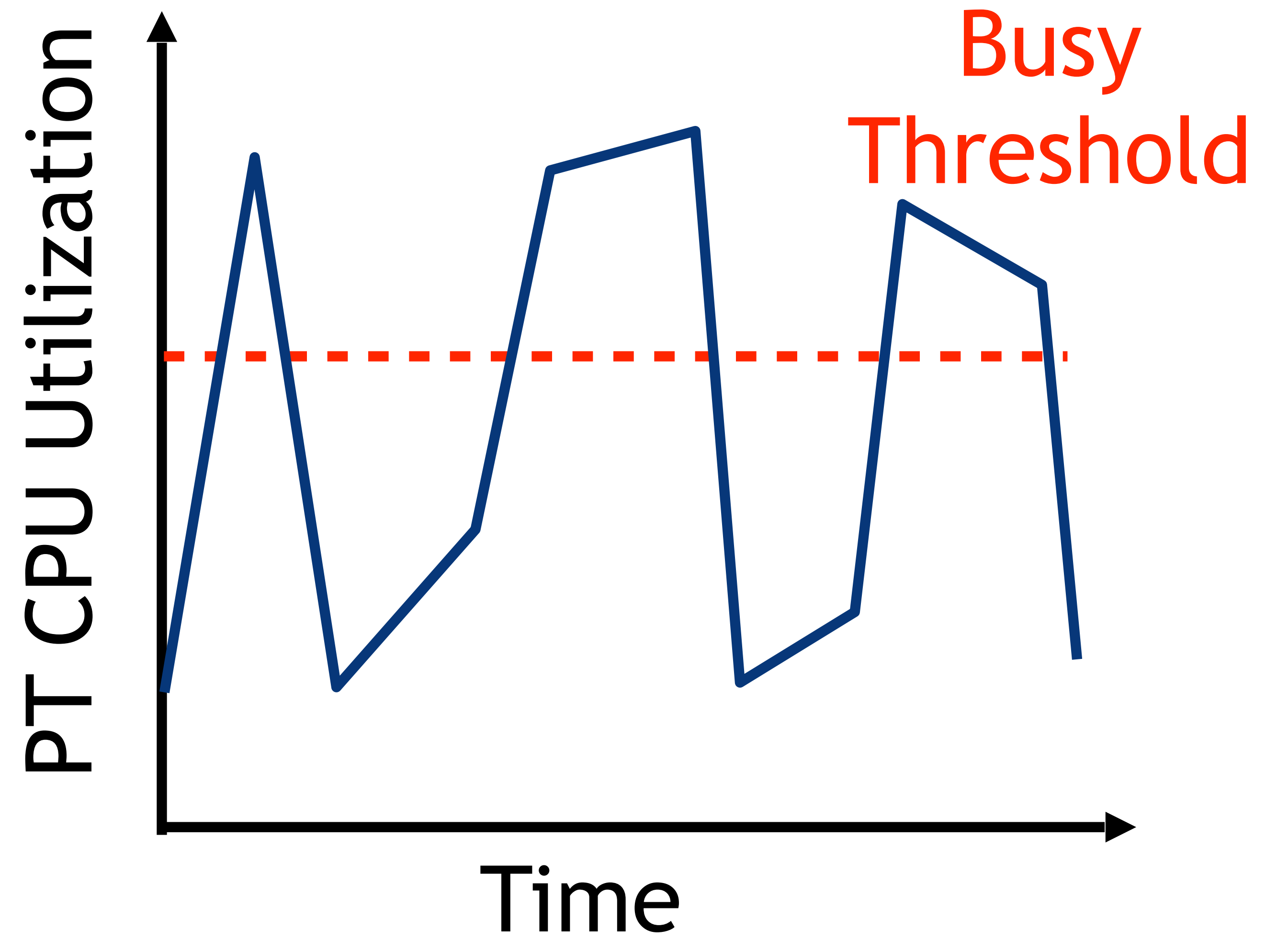
- Primary Tenants (PTs) *own* servers
  - Interactive services (e.g., Bing) are PTs
- Harvest resources from PTs
  - Avoid performance impact to the PT
- Challenges for distributed file systems
  - Busy servers fail accesses → lower availability

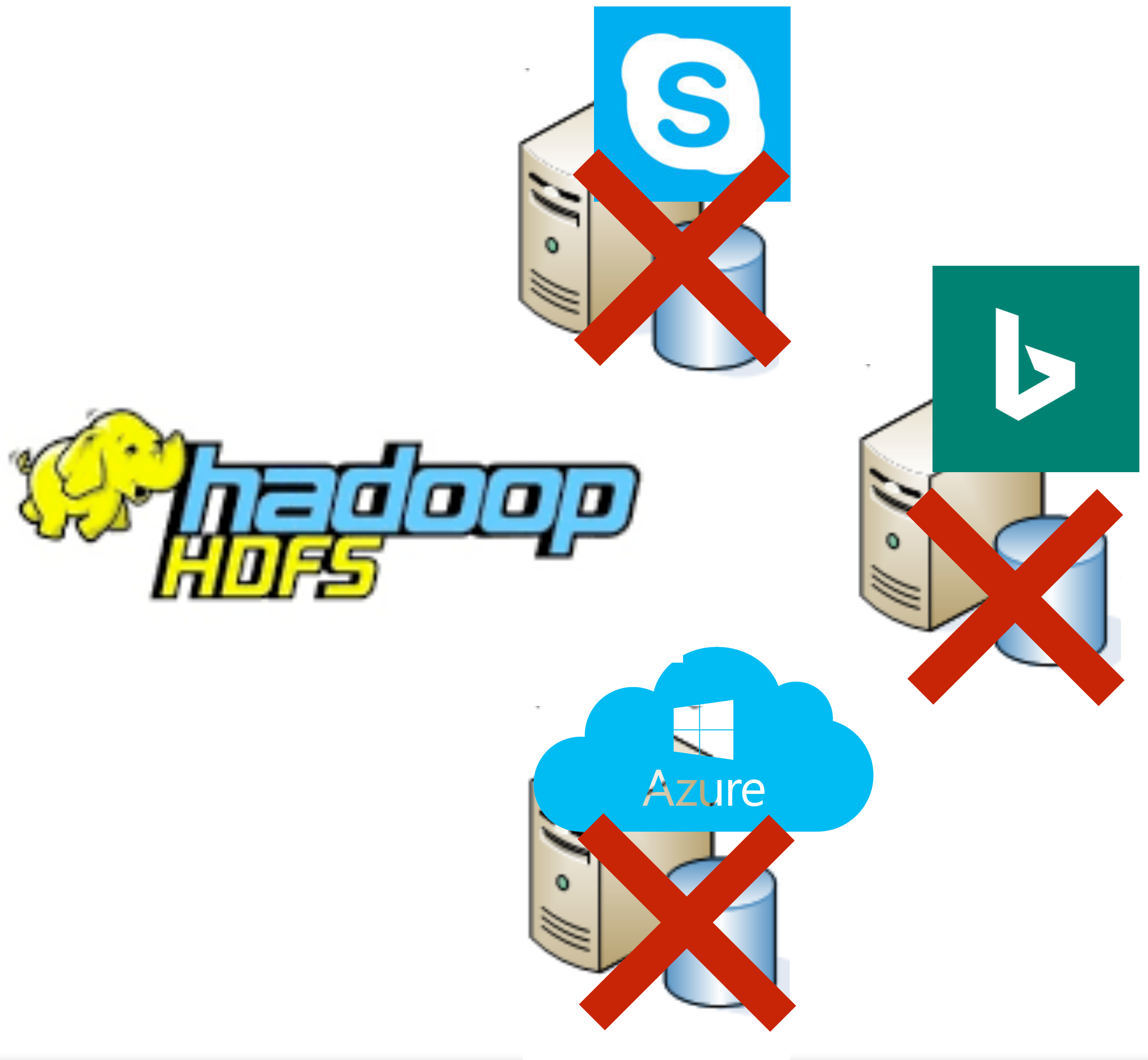




Resource-Harvesting Challenges

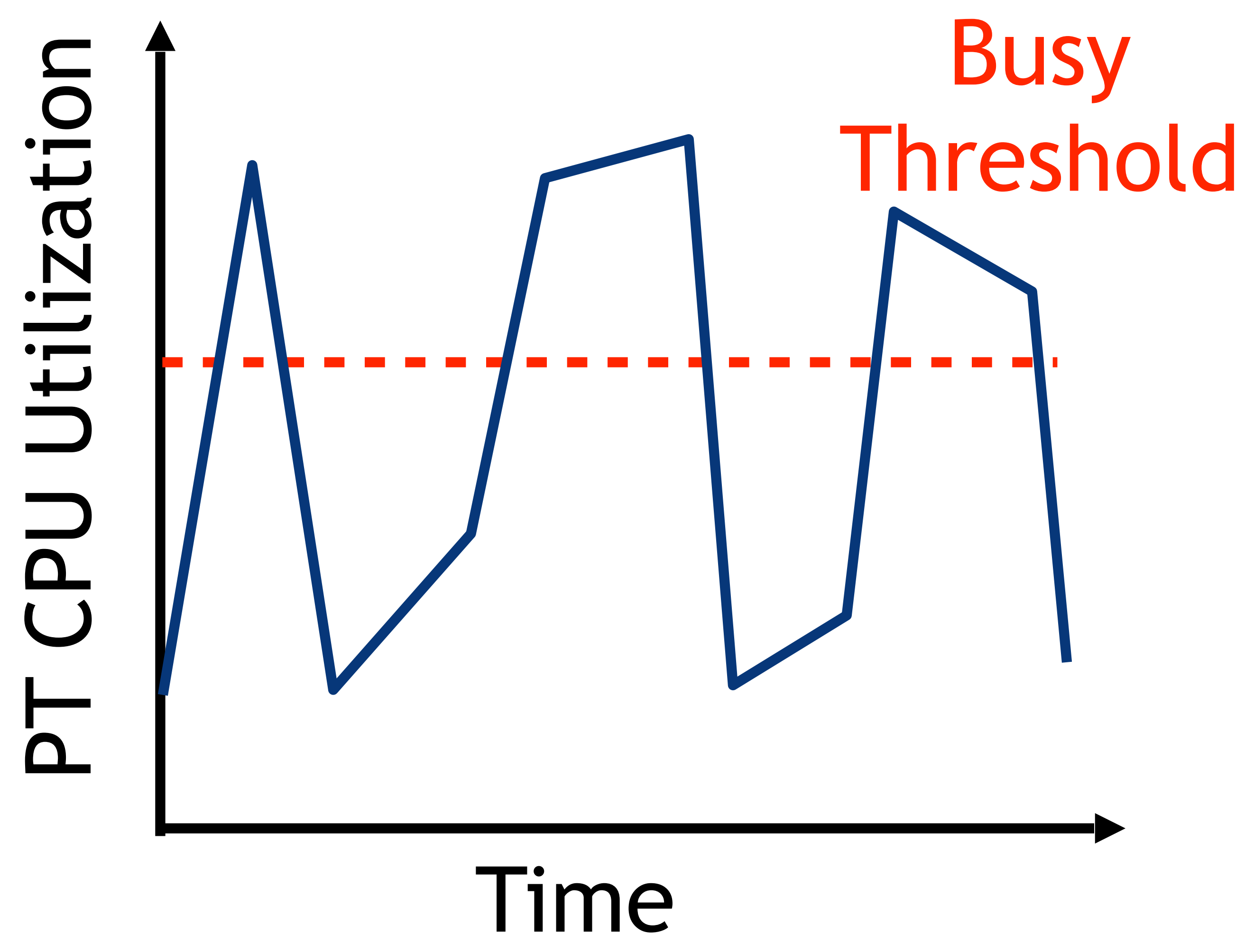
- Primary Tenants (PTs) *own* servers
  - Interactive services (e.g., Bing) are PTs
- Harvest resources from PTs
  - Avoid performance impact to the PT
- Challenges for distributed file systems
  - Busy servers fail accesses → lower availability
  - Re-image disks → lower durability

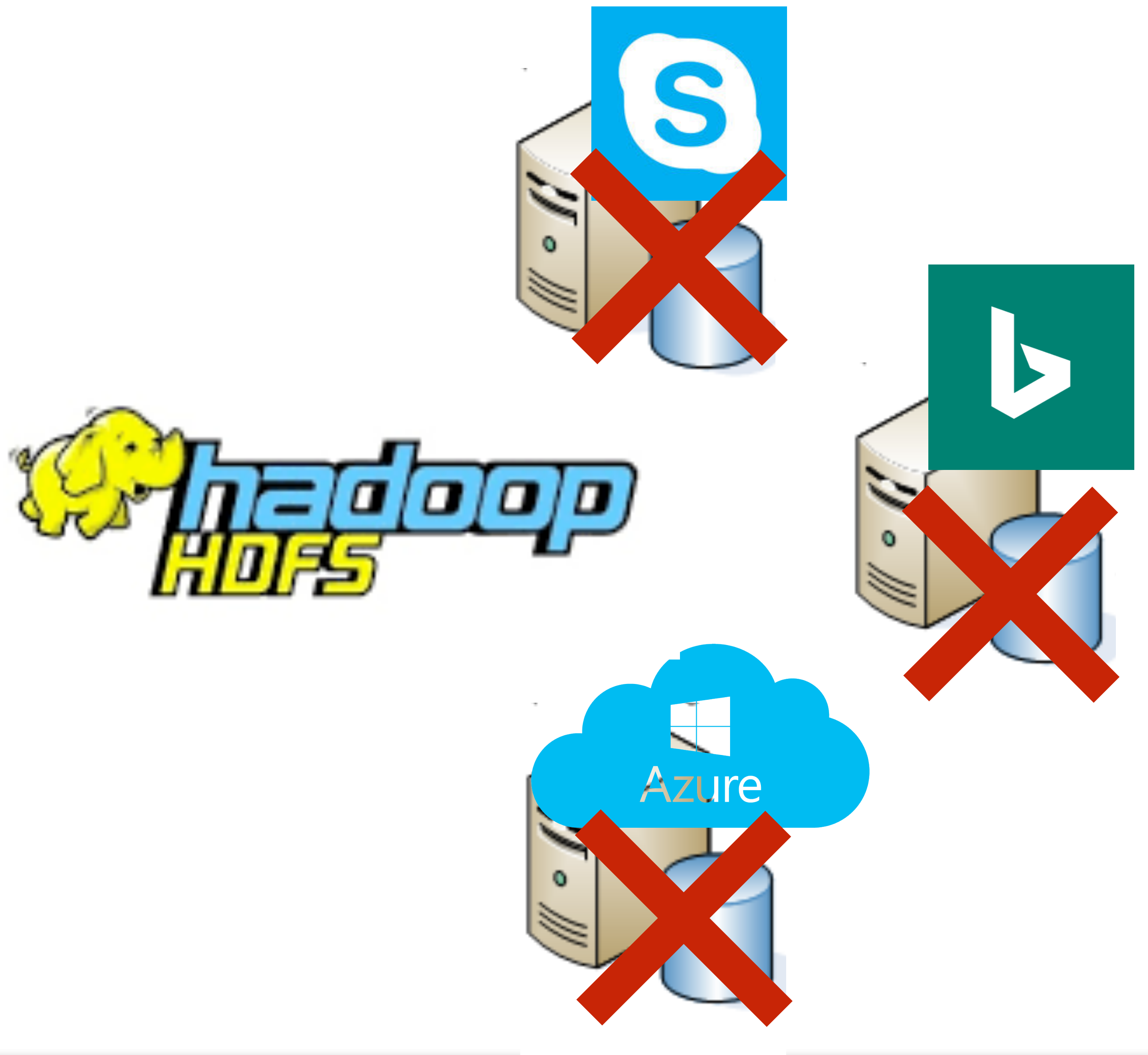




Resource-Harvesting Challenges

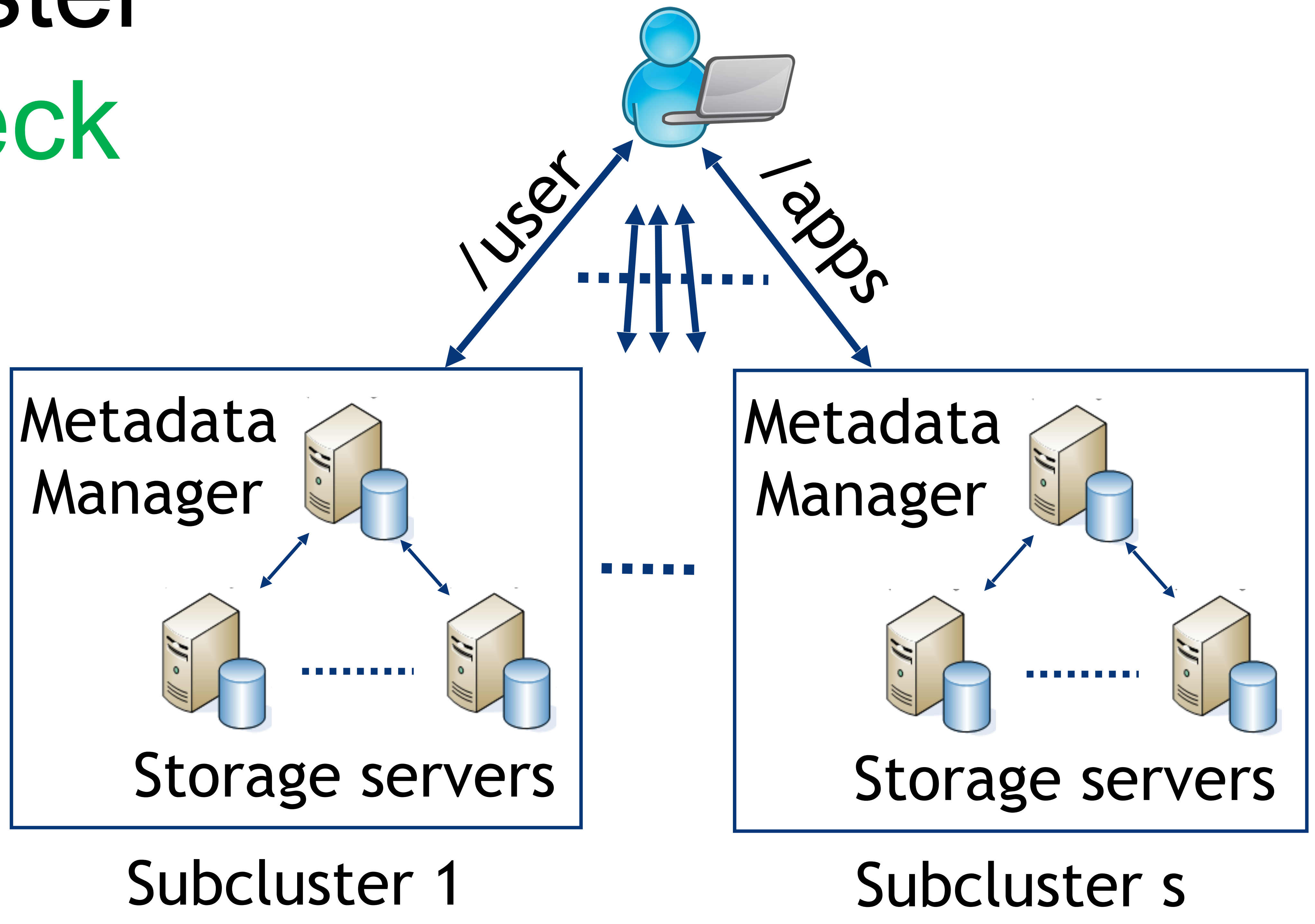
- Primary Tenants (PTs) *own* servers
  - Interactive services (e.g., Bing) are PTs
- Harvest resources from PTs
  - Avoid performance impact to the PT
- Challenges for distributed file systems
  - Busy servers fail accesses → lower availability
  - Re-image disks → lower durability
- Place replicas across PTs [Zhang,OSDI'16]
  - Need diversity of PT servers in filesystem





Scaling Technique #1: ViewFS

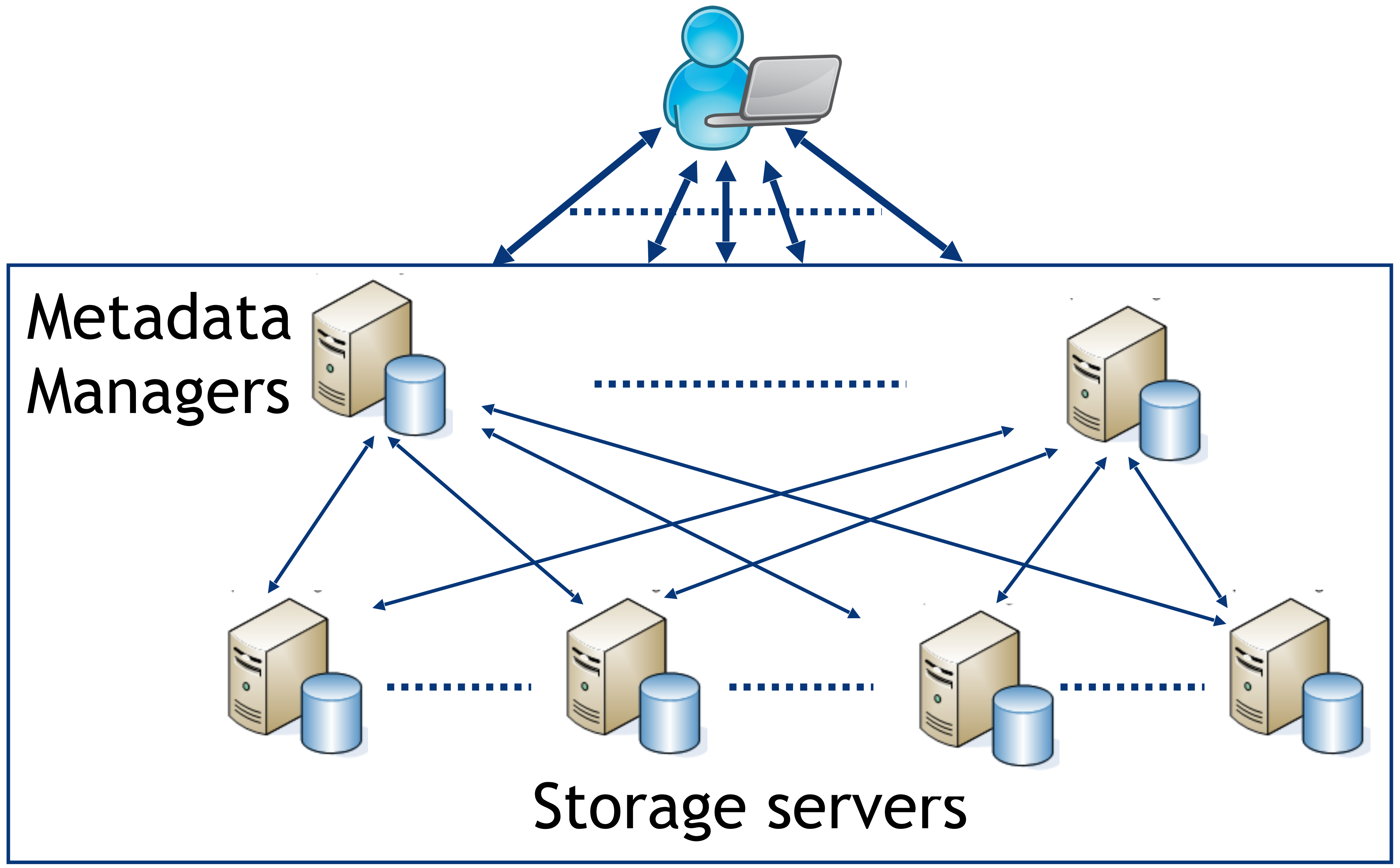
- Partition of namespace on a subcluster
- Mitigate metadata manager bottleneck
- Users manually place data
  - Unbalanced subclusters
  - Complex rebalance



- *Need global view of the namespace, automated management*

Scaling Technique #2: Multiple Metadata Managers

- Single cluster with multiple strongly consistent metadata managers
- Global view of the namespace
- More complex
- No isolation from bugs or failures



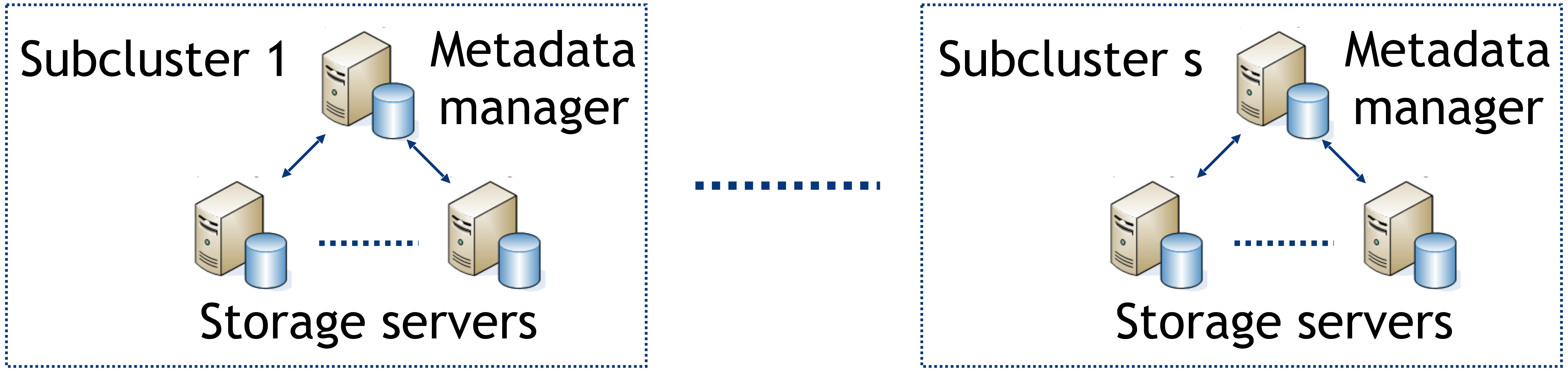
- *Need small independent subclusters for isolation* [Verma, EuroSys'15]

Goals

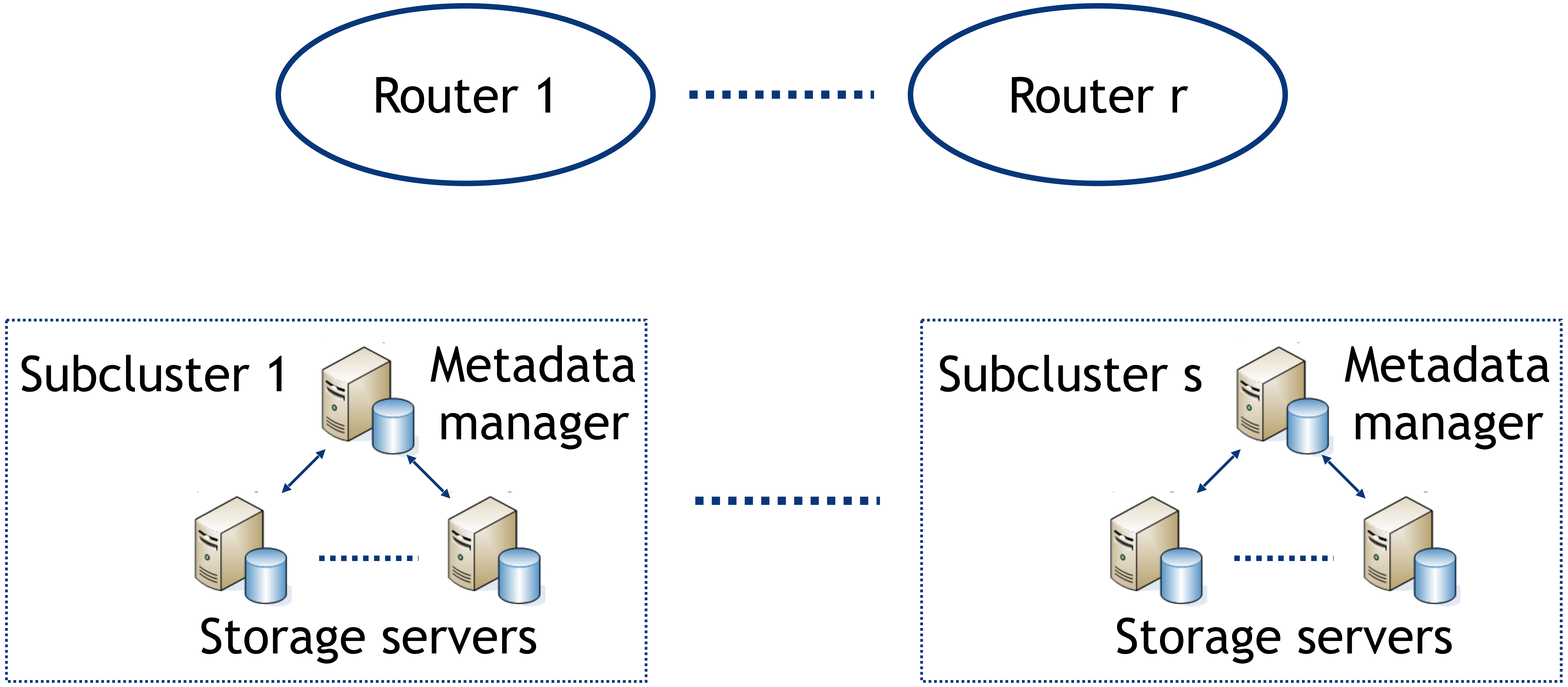
1. Scale file systems to entire datacenter
   - Run independent subclusters → isolation
   - *Federate* subclusters transparently → global namespace
2. Enable resource-harvesting
   - Promote behavioral diversity → improve durability and availability
3. Good performance for users
   - Balance load and capacity

Our Solution: Datacenter-Harvesting File System

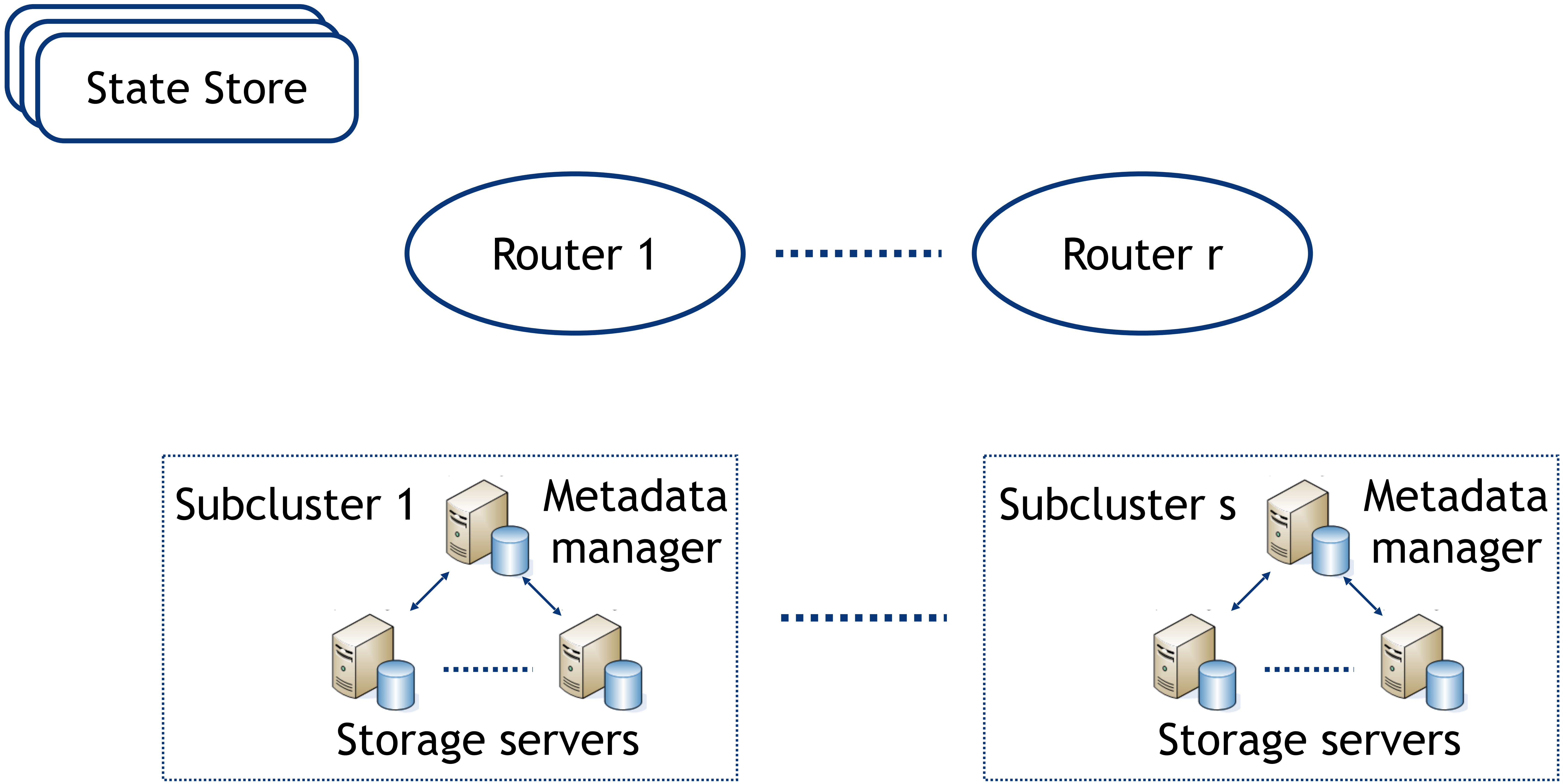
Our Solution: Datacenter-Harvesting File System



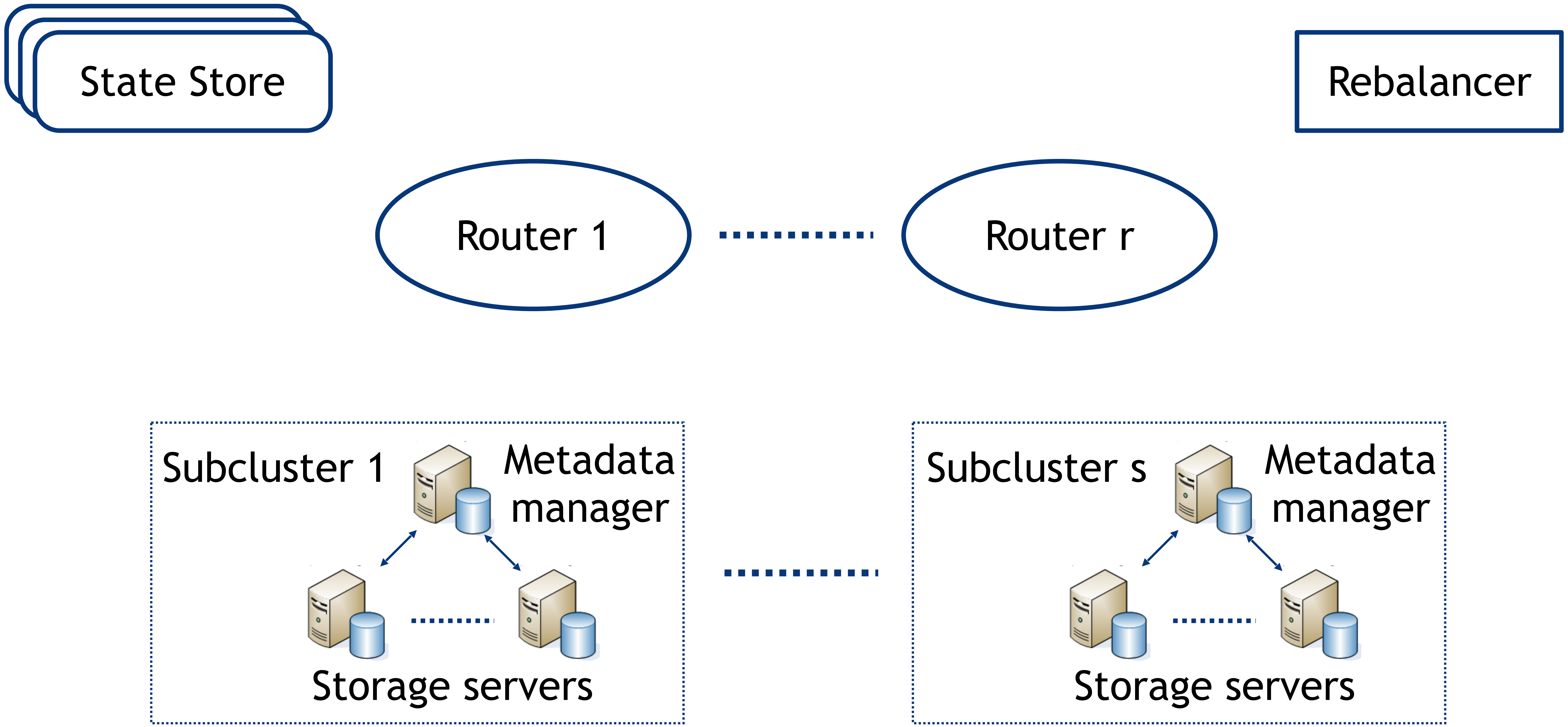
Our Solution: Datacenter-Harvesting File System



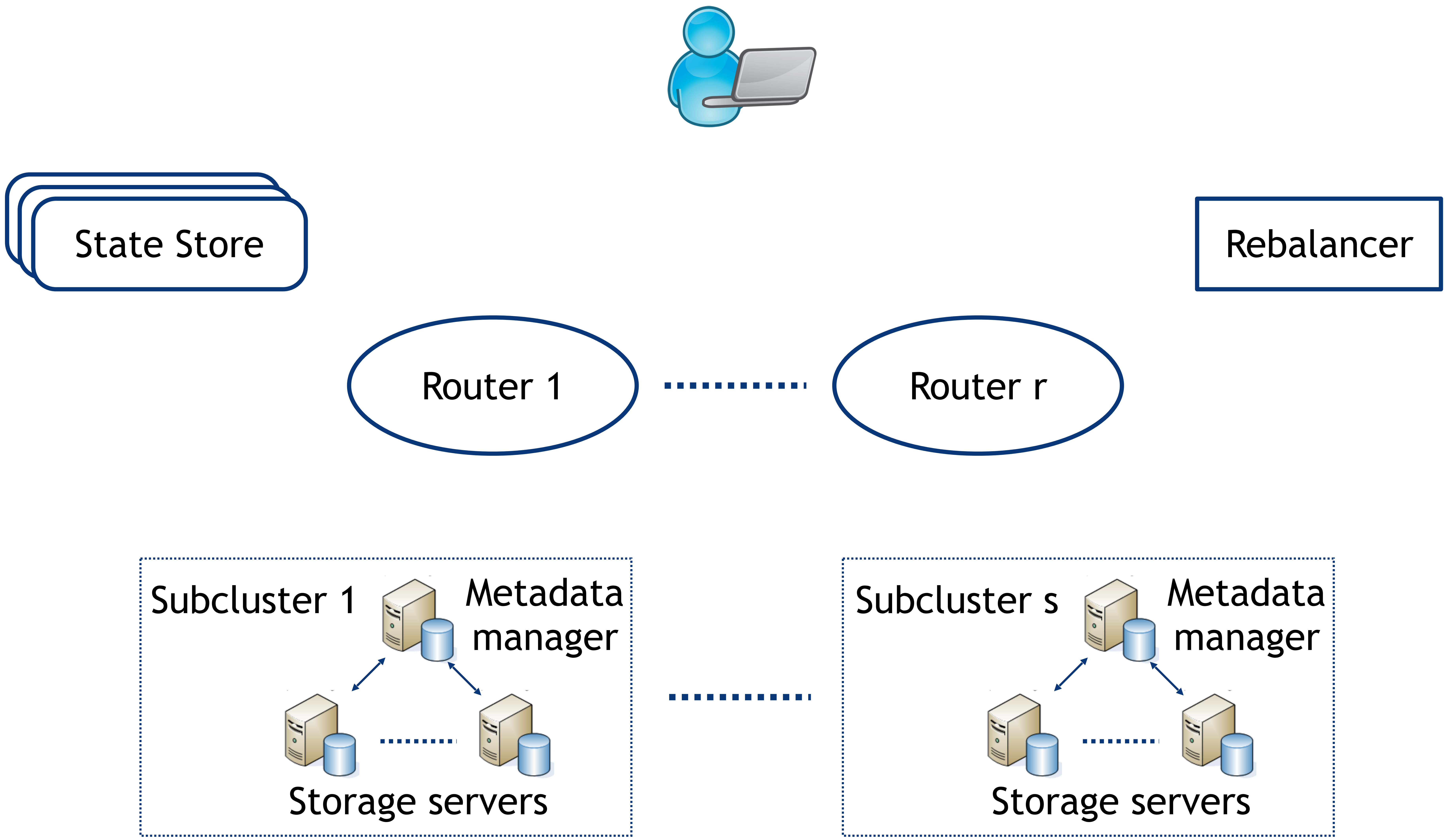
Our Solution: Datacenter-Harvesting File System



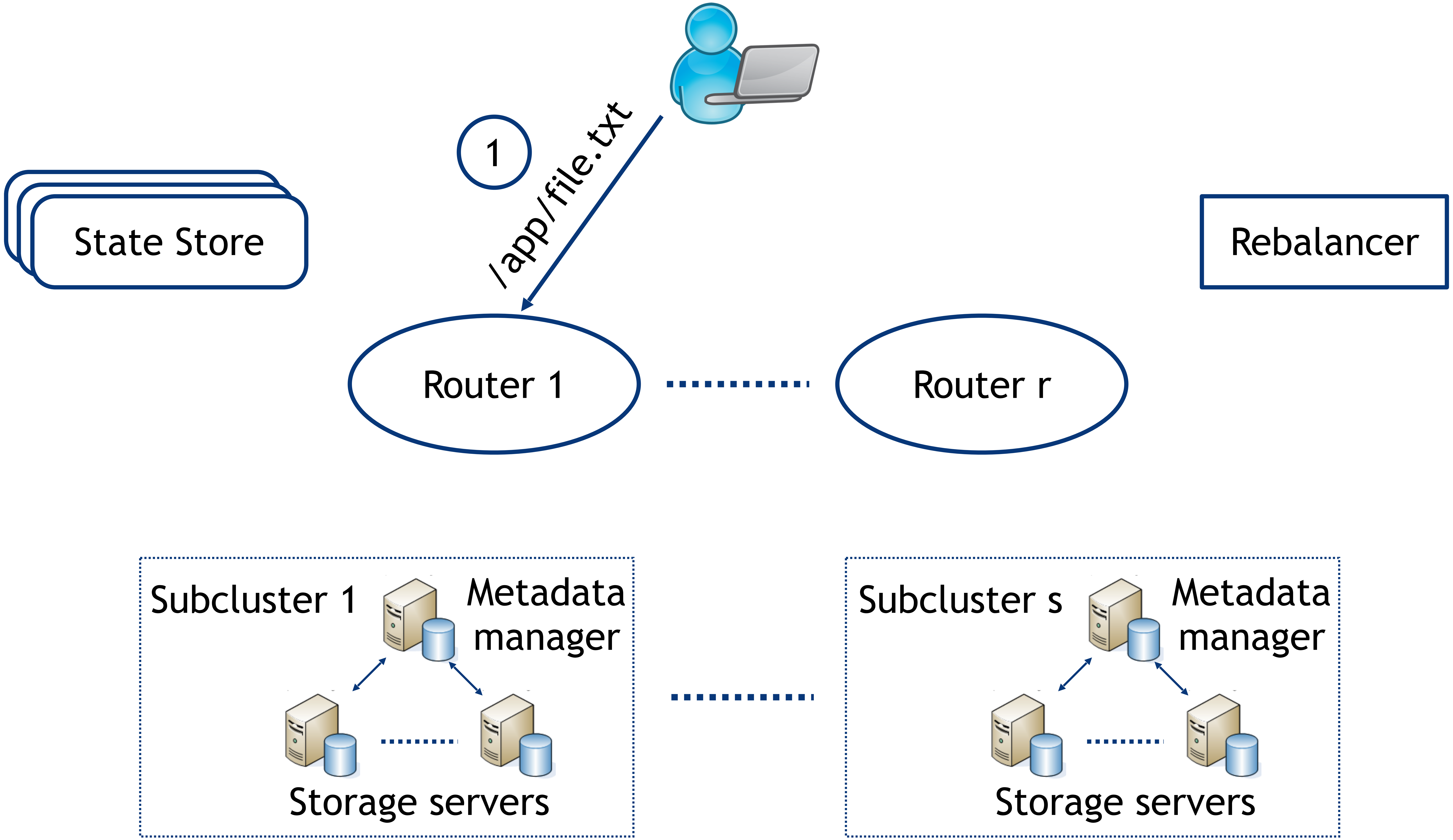
Our Solution: Datacenter-Harvesting File System



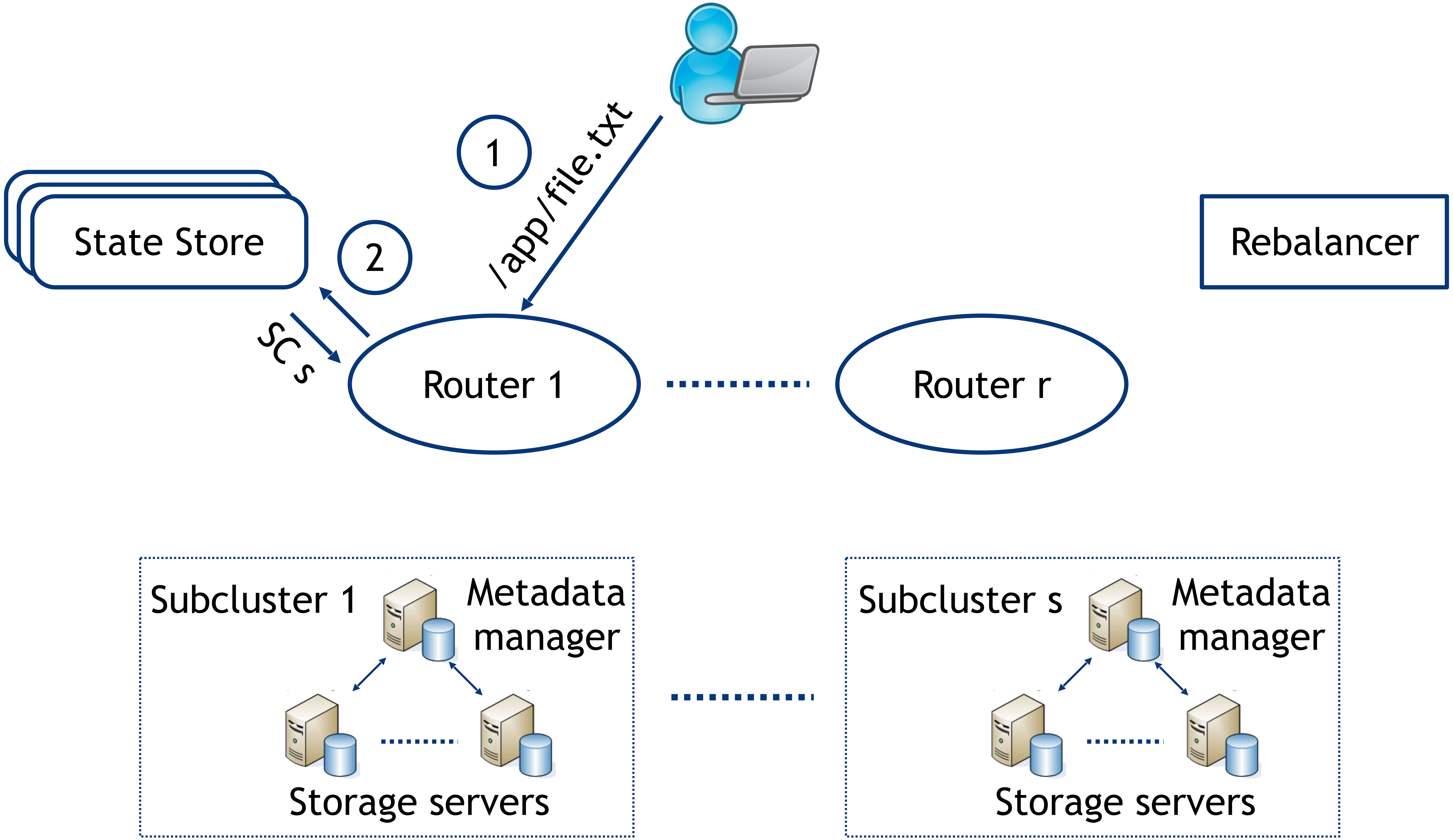
Our Solution: Datacenter-Harvesting File System



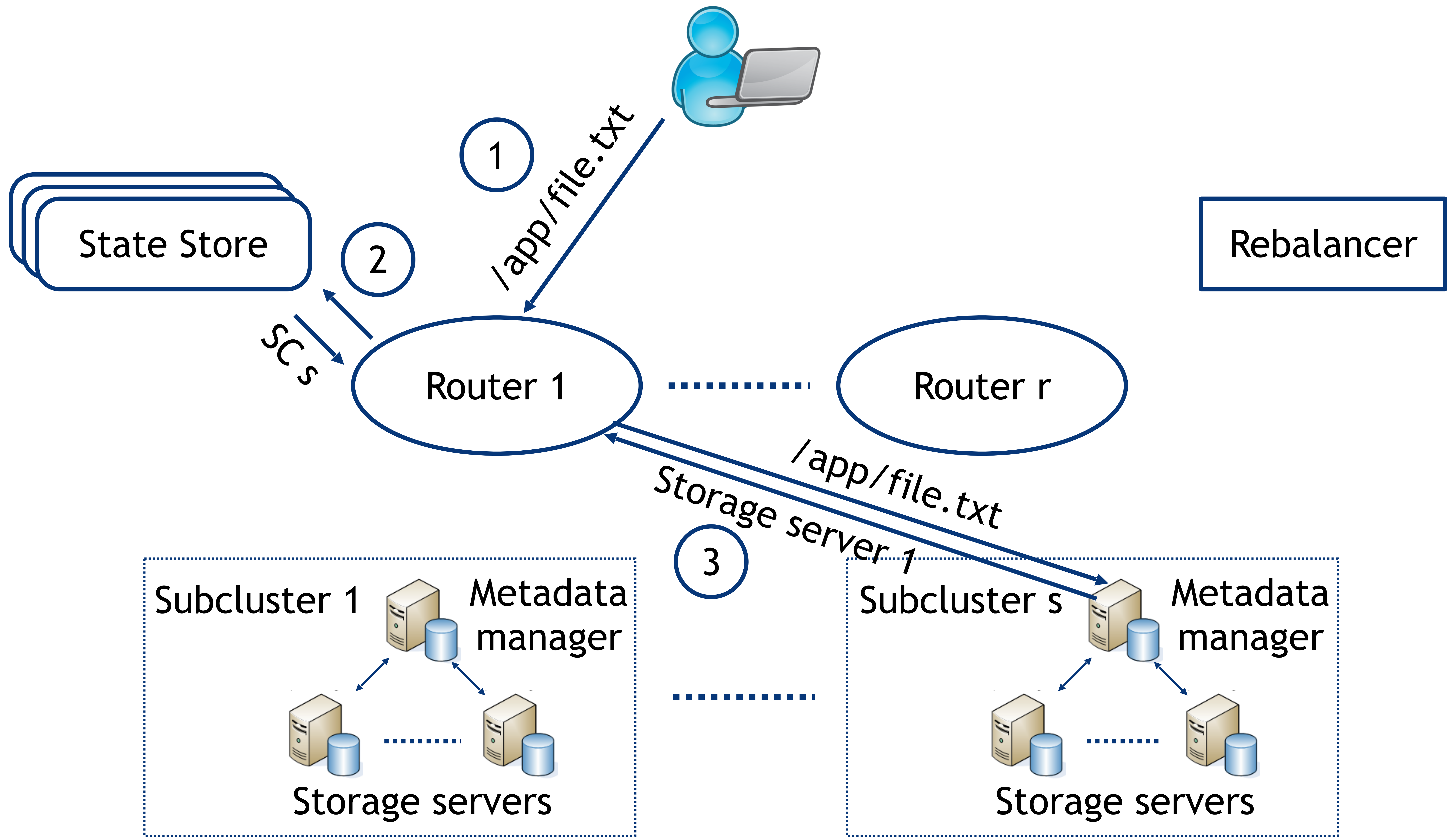
Our Solution: Datacenter-Harvesting File System



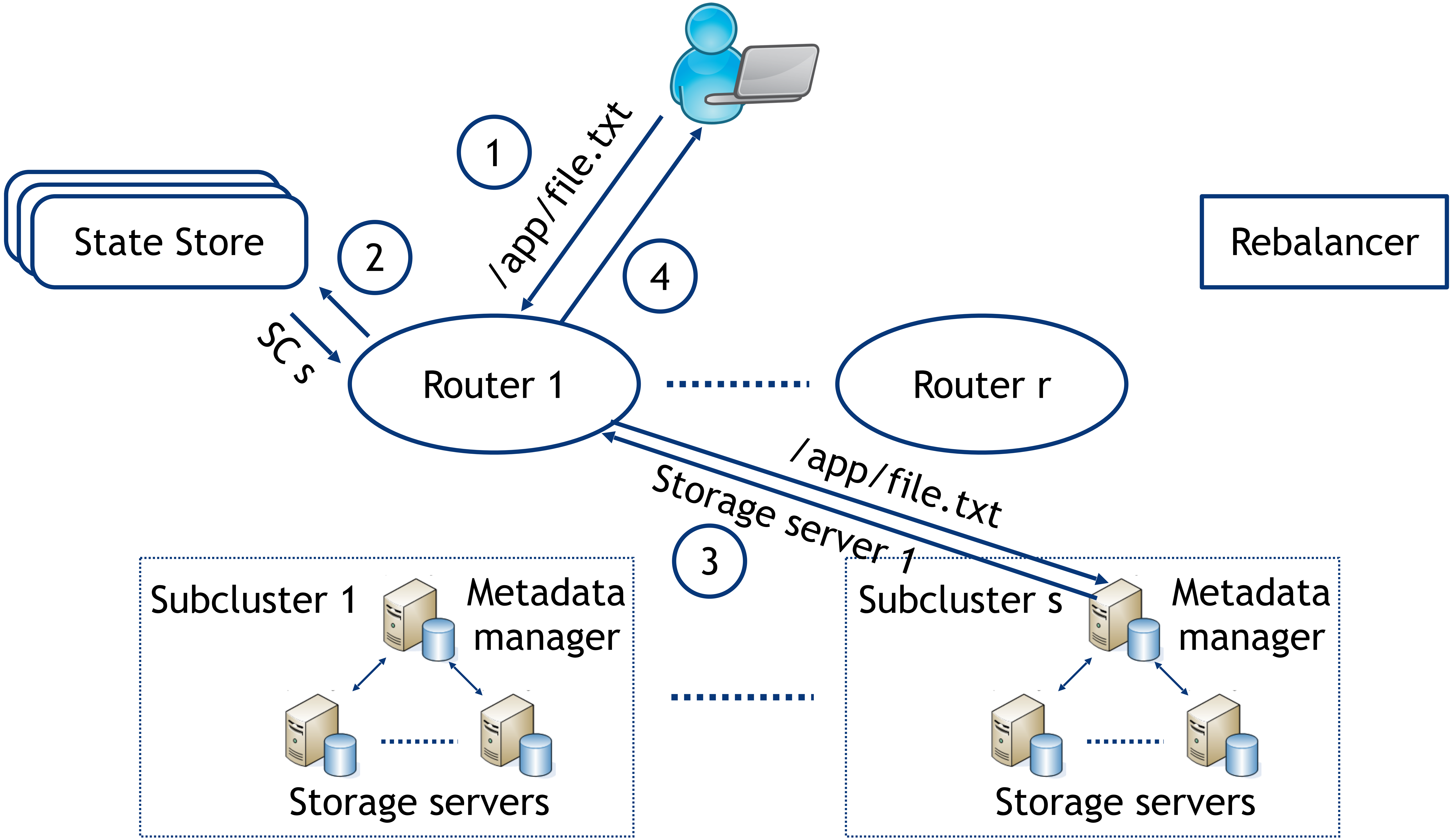
Our Solution: Datacenter-Harvesting File System



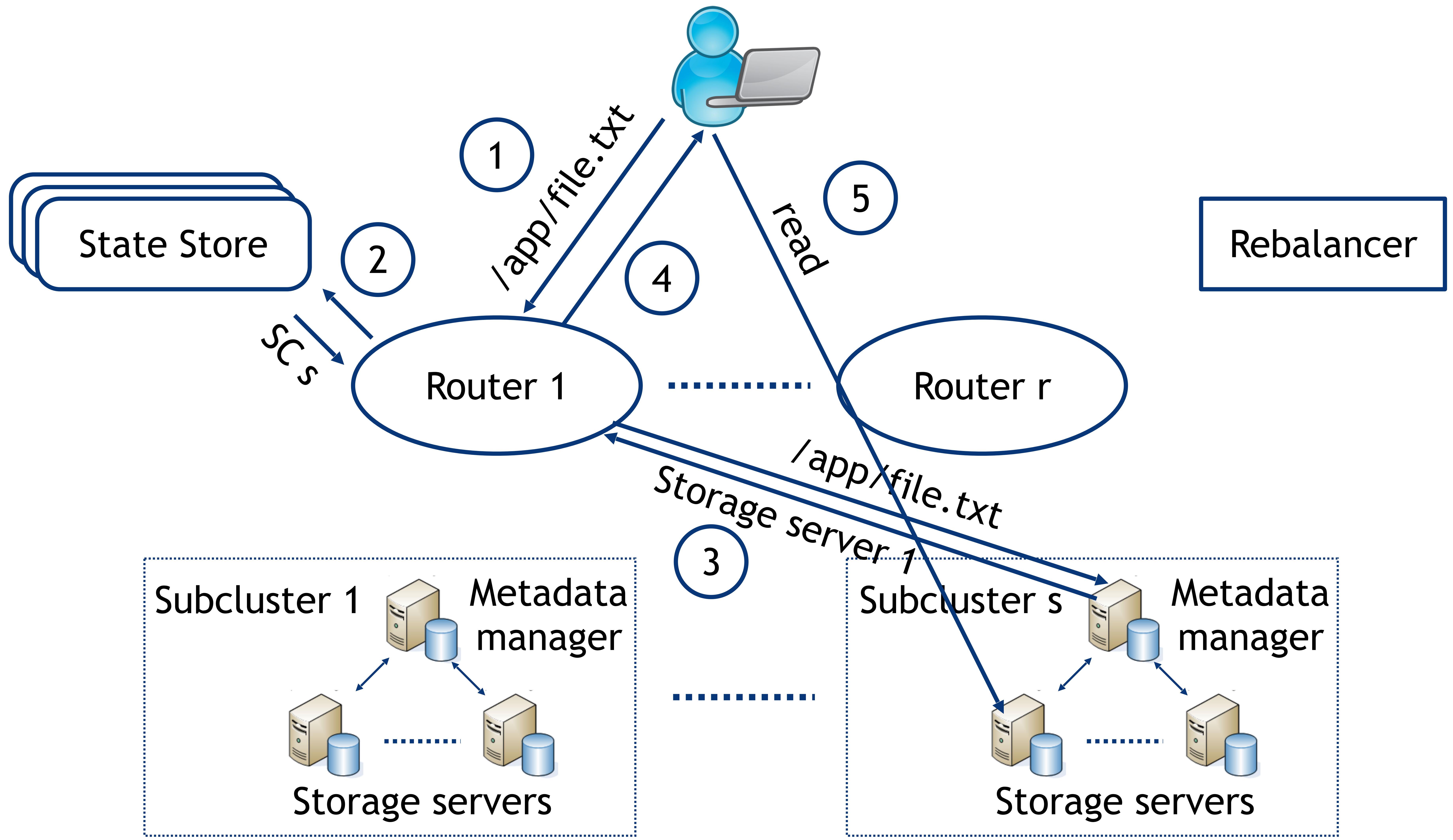
Our Solution: Datacenter-Harvesting File System



Our Solution: Datacenter-Harvesting File System



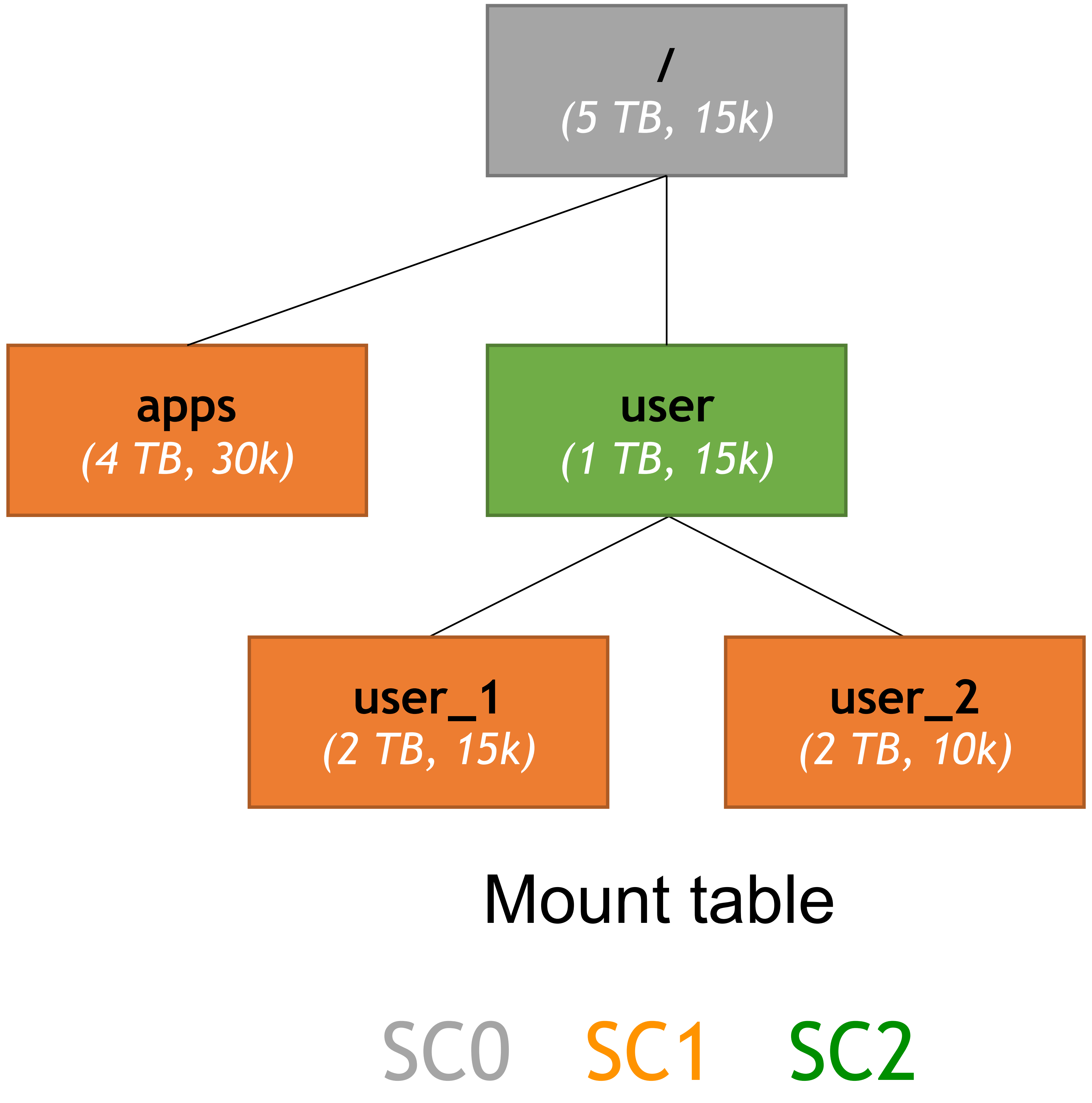
Our Solution: Datacenter-Harvesting File System



Goal #1: Transparent Scaling of File Systems

- State Store
  - Mount table: path → subcluster (SC)
  - Access load and capacity metrics
  - Router and rebalancer state

- Routers
  - Expose global namespace
  - Consult state store for path → sub cluster
  - Cache path resolutions



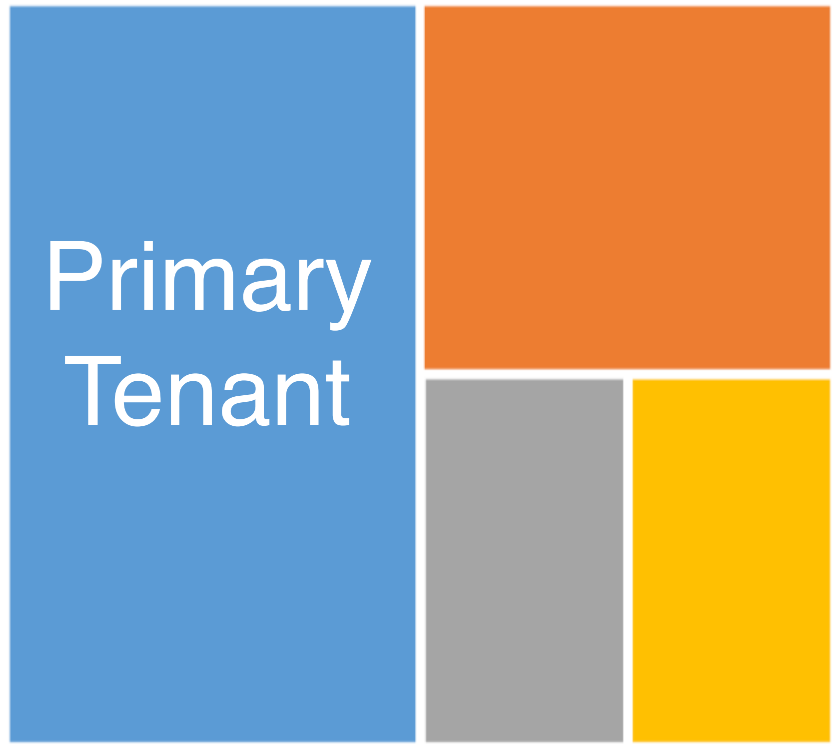
Goal #2: Enable Resource Harvesting

- Provide high availability and durability
  - Exploit behavioral diversity [Zhang,OSDI'16]

Goal #2: Enable Resource Harvesting

- Provide high availability and durability
  - Exploit behavioral diversity [Zhang,OSDI'16]
- Manual: Primary Tenant → SC
  - Less diversity in subclusters

Diversity in subclusters
(# of Primary Tenants)

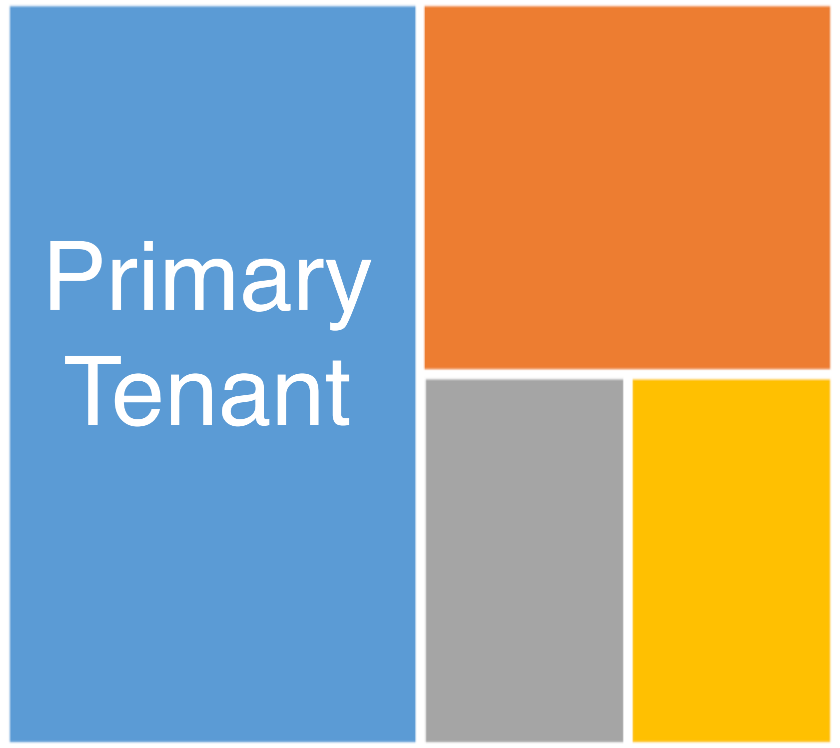


Manual assignment

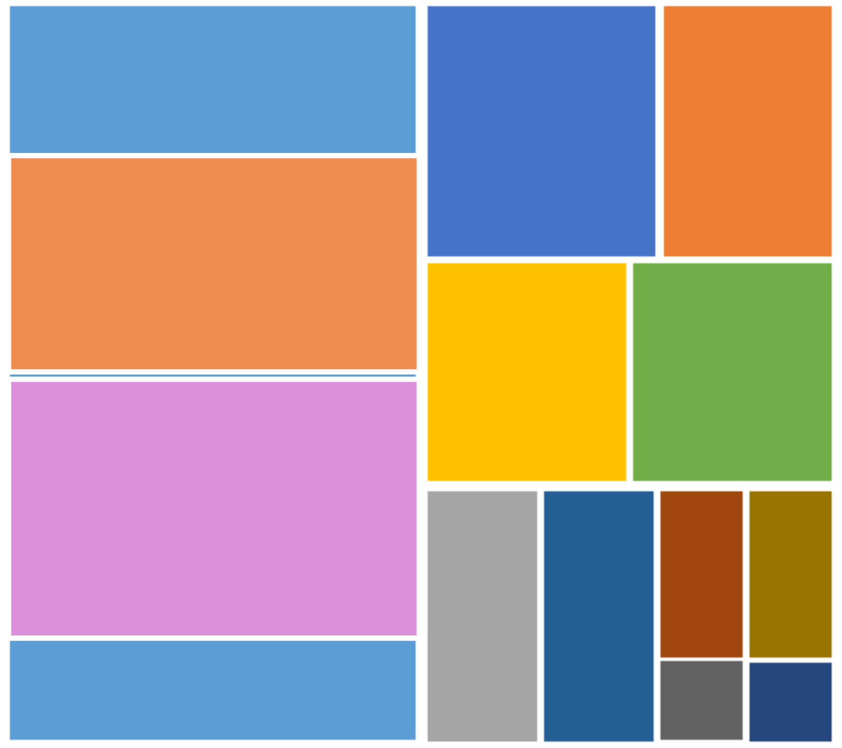
Goal #2: Enable Resource Harvesting

- Provide high availability and durability
  - Exploit behavioral diversity [Zhang,OSDI'16]
- Manual: Primary Tenant → SC
  - Less diversity in subclusters
- Consistent hashing: racks → SC
  - Randomization to promote diversity
  - Promote network locality (racks → SC)
  - Reduce data movement on SC add/remove

Diversity in subclusters
(# of Primary Tenants)



Manual assignment

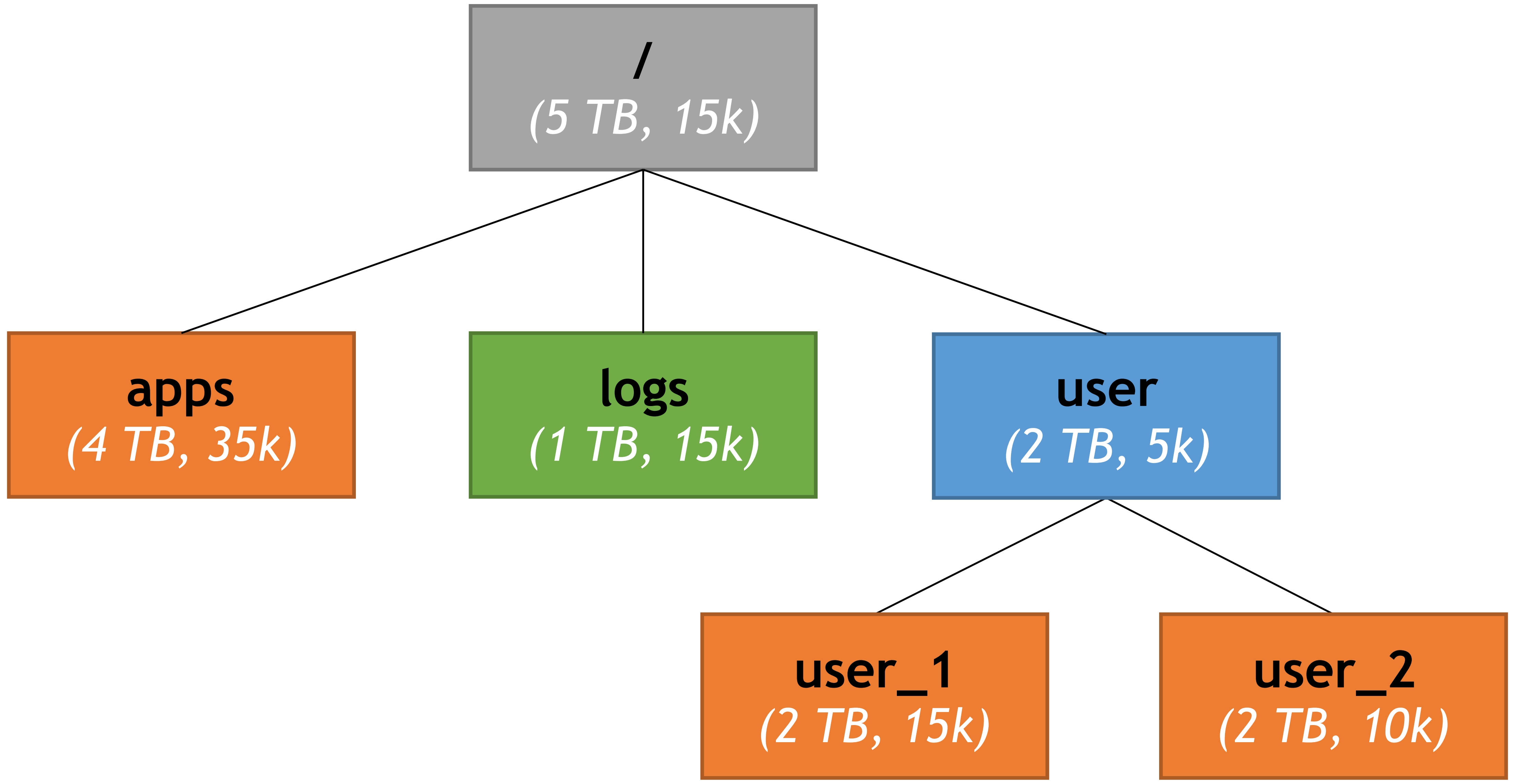


Consistent hashing

Goal #3: Ensure Good Performance for Users

- Rebalancer as a minimization problem
  - Used capacity (< 80% of available capacity)
  - Access load (< 40k reqs/sec over a 5 minute period)
  - Amount of data moved for rebalancing
  - Mount table size

Goal #3: Ensure Good Performance for Users

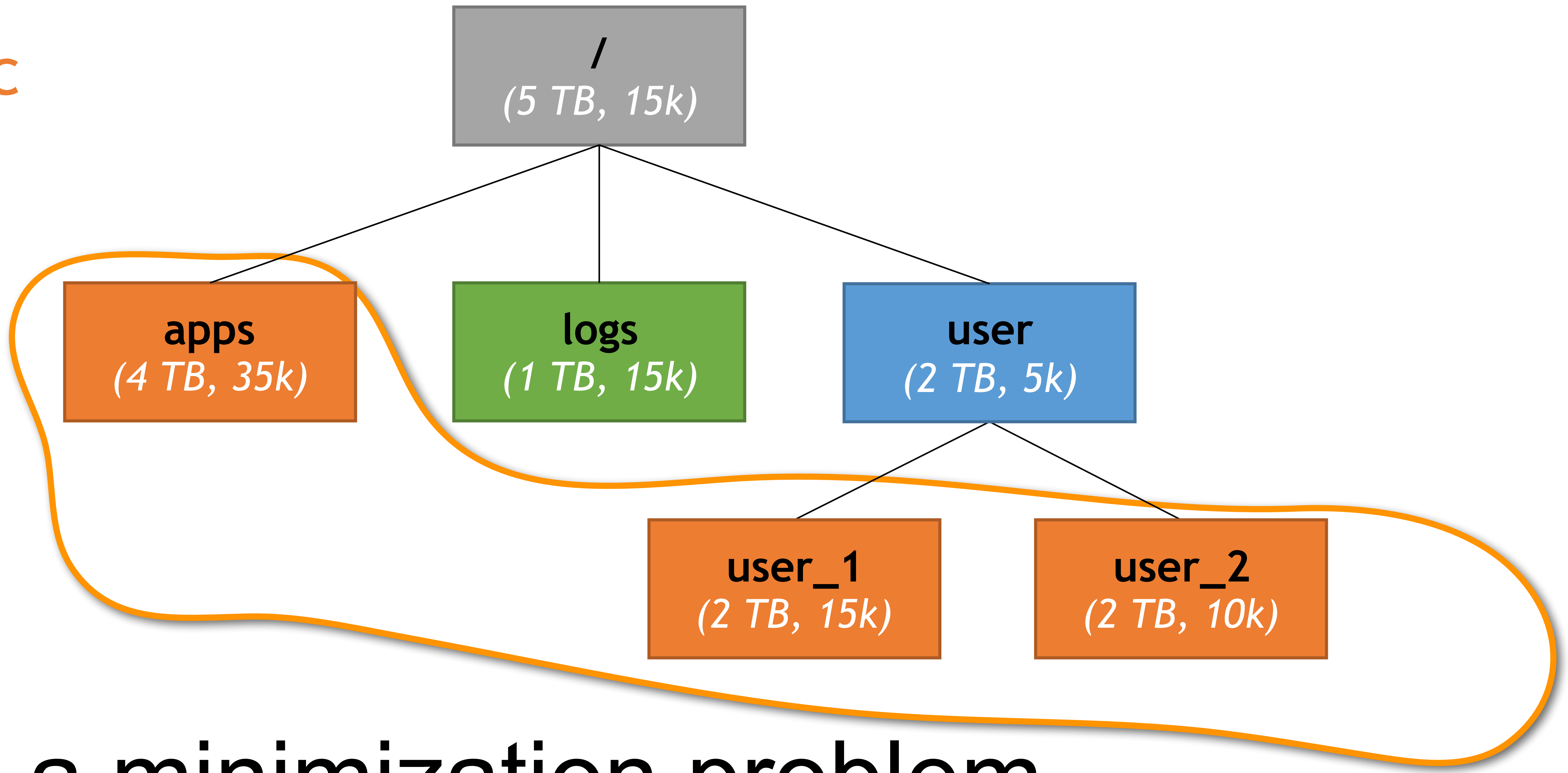


- Rebalancer as a minimization problem
  - Used capacity (< 80% of available capacity)
  - Access load (< 40k reqs/sec over a 5 minute period)
  - Amount of data moved for rebalancing
  - Mount table size

Goal #3: Ensure Good Performance for Users

SC1: 8 TB, 60k reqs/sec

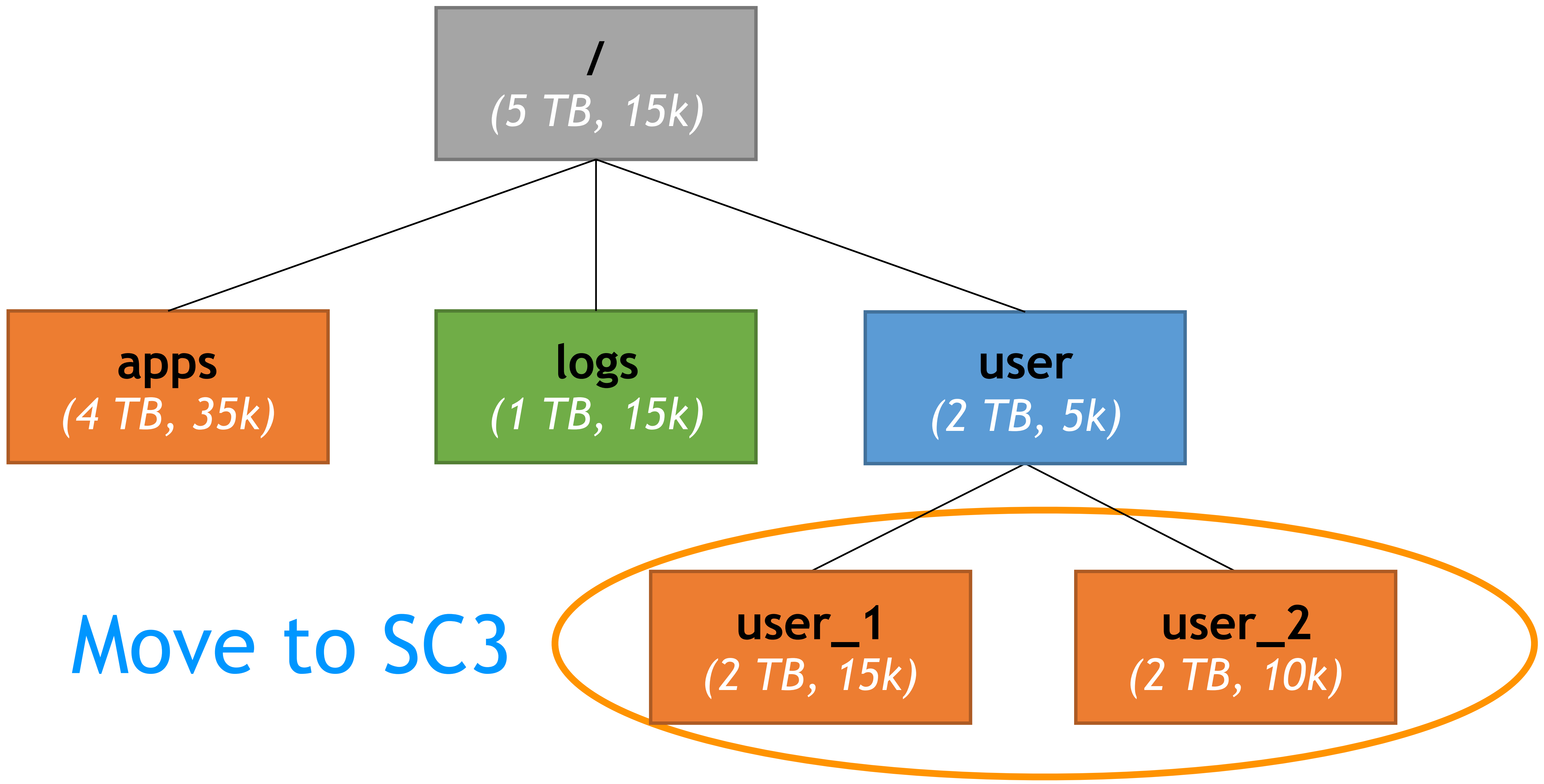
SC1 overloaded



- Rebalancer as a minimization problem
  - Used capacity (< 80% of available capacity)
  - Access load (< 40k reqs/sec over a 5 minute period)
  - Amount of data moved for rebalancing
  - Mount table size

Goal #3: Ensure Good Performance for Users

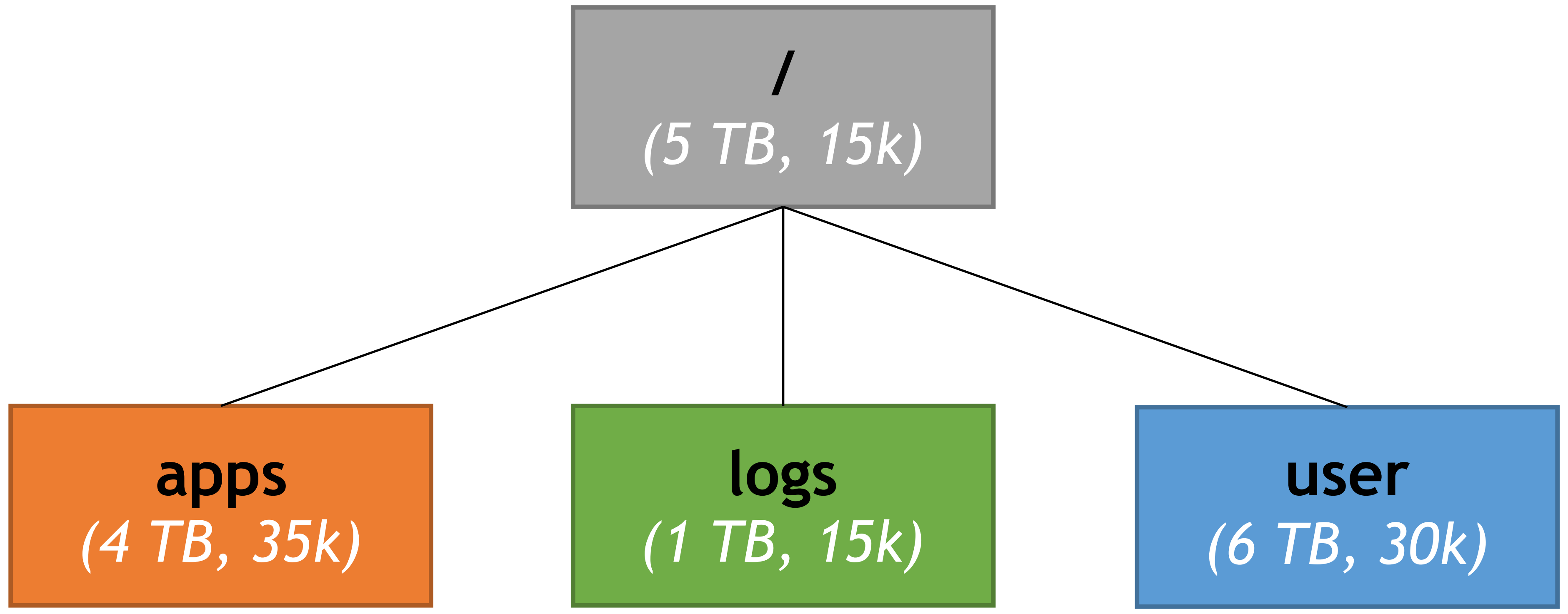
SC1: 8 TB, 60k reqs/sec



- Rebalancer as a minimization problem
  - Used capacity (< 80% of available capacity)
  - Access load (< 40k reqs/sec over a 5 minute period)
  - Amount of data moved for rebalancing
  - Mount table size

Goal #3: Ensure Good Performance for Users

SC1: 8 TB, 60k reqs/sec



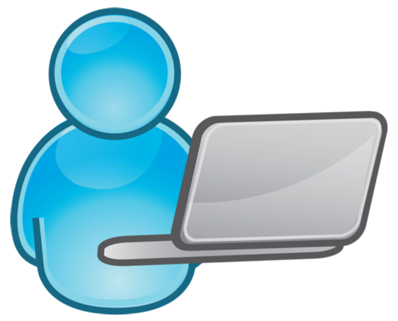
SC1: 4 TB, 35k reqs/sec
SC3: 6 TB, 30k reqs/sec

- Rebalancer as a minimization problem
  - Used capacity (< 80% of available capacity)
  - Access load (< 40k reqs/sec over a 5 minute period)
  - Amount of data moved for rebalancing
  - Mount table size

Implementation

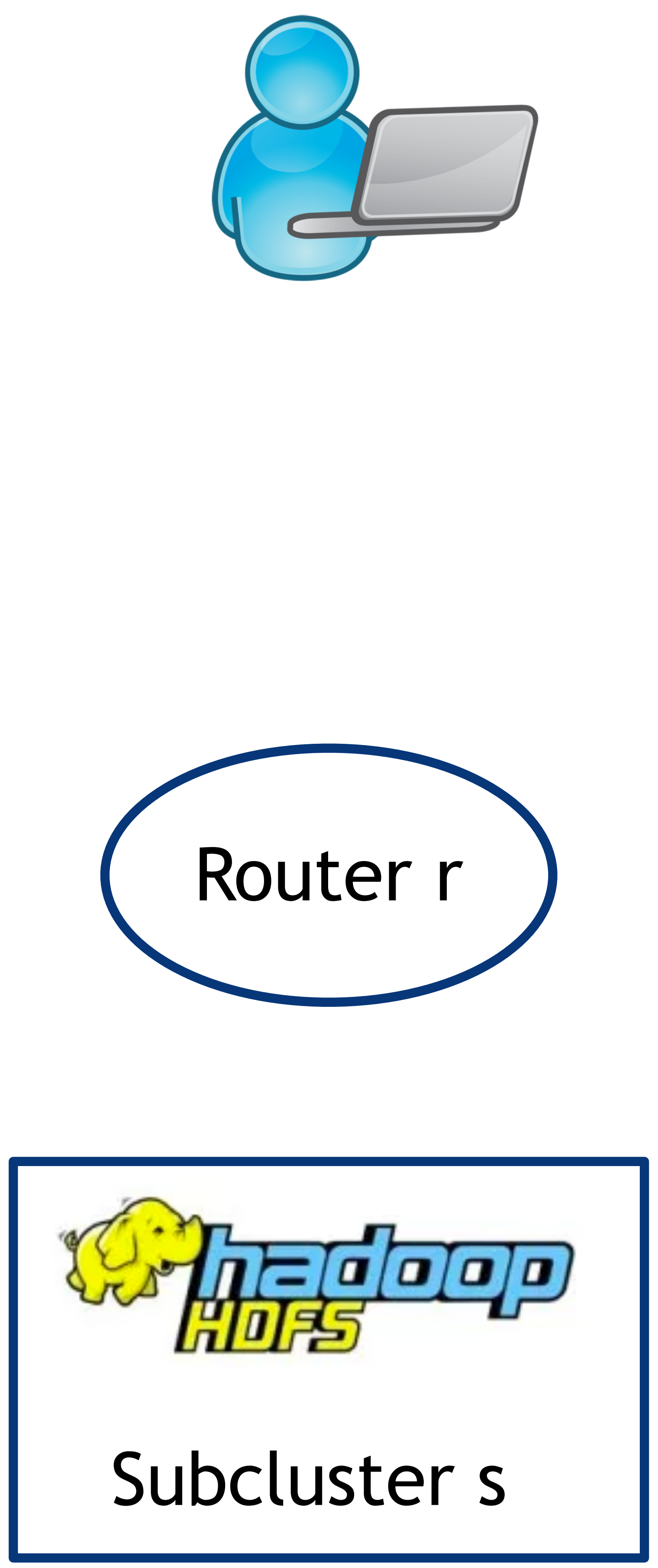
Implementation

- Datacenter-Harvesting HDFS (DH-HDFS)
  - Implement federation architecture over HDFS
  - Diversity-aware replica placement
  - Run independent instances in subclusters



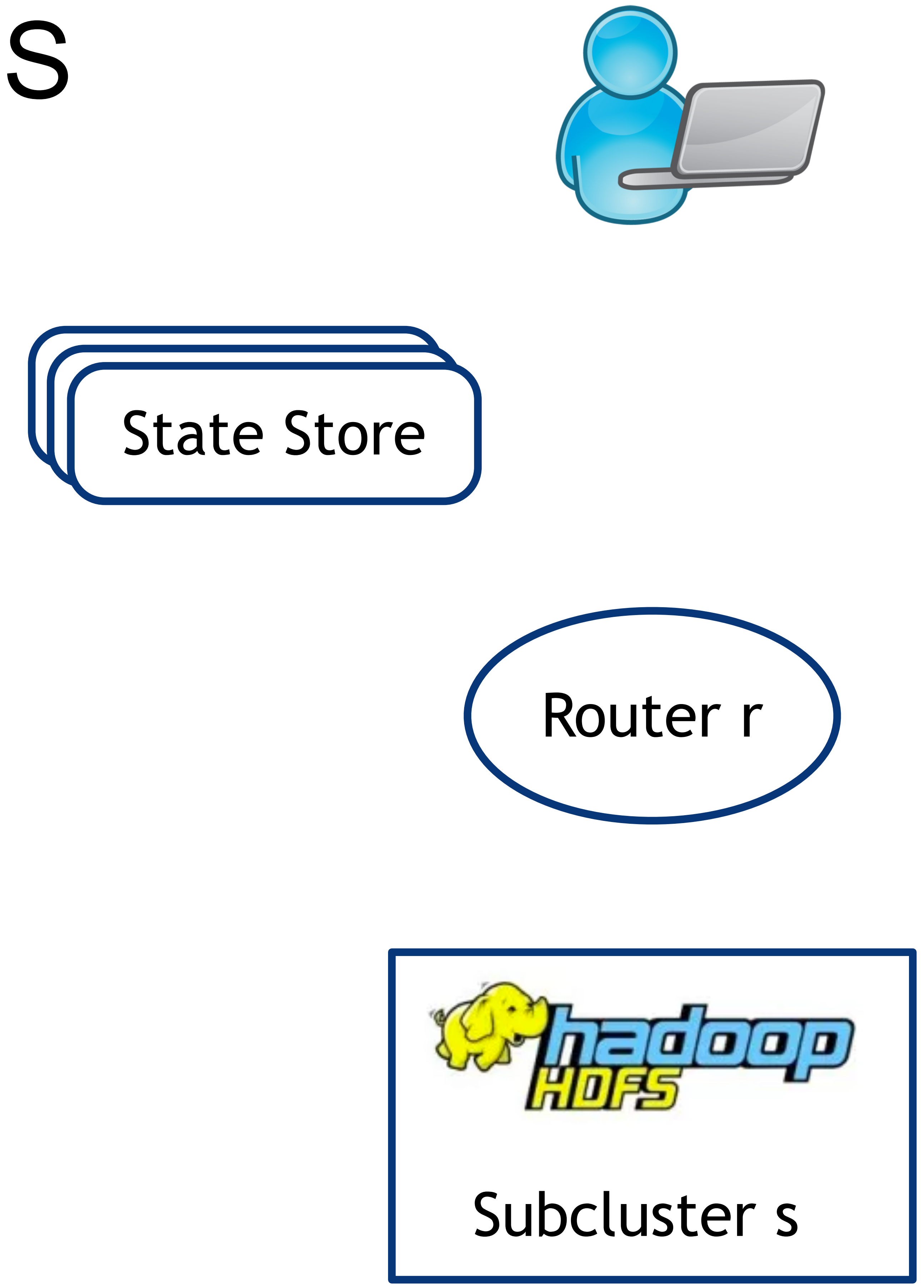
Implementation

- Datacenter-Harvesting HDFS (DH-HDFS)
  - Implement federation architecture over HDFS
  - Diversity-aware replica placement
  - Run independent instances in subclusters
- Load balancing for Routers



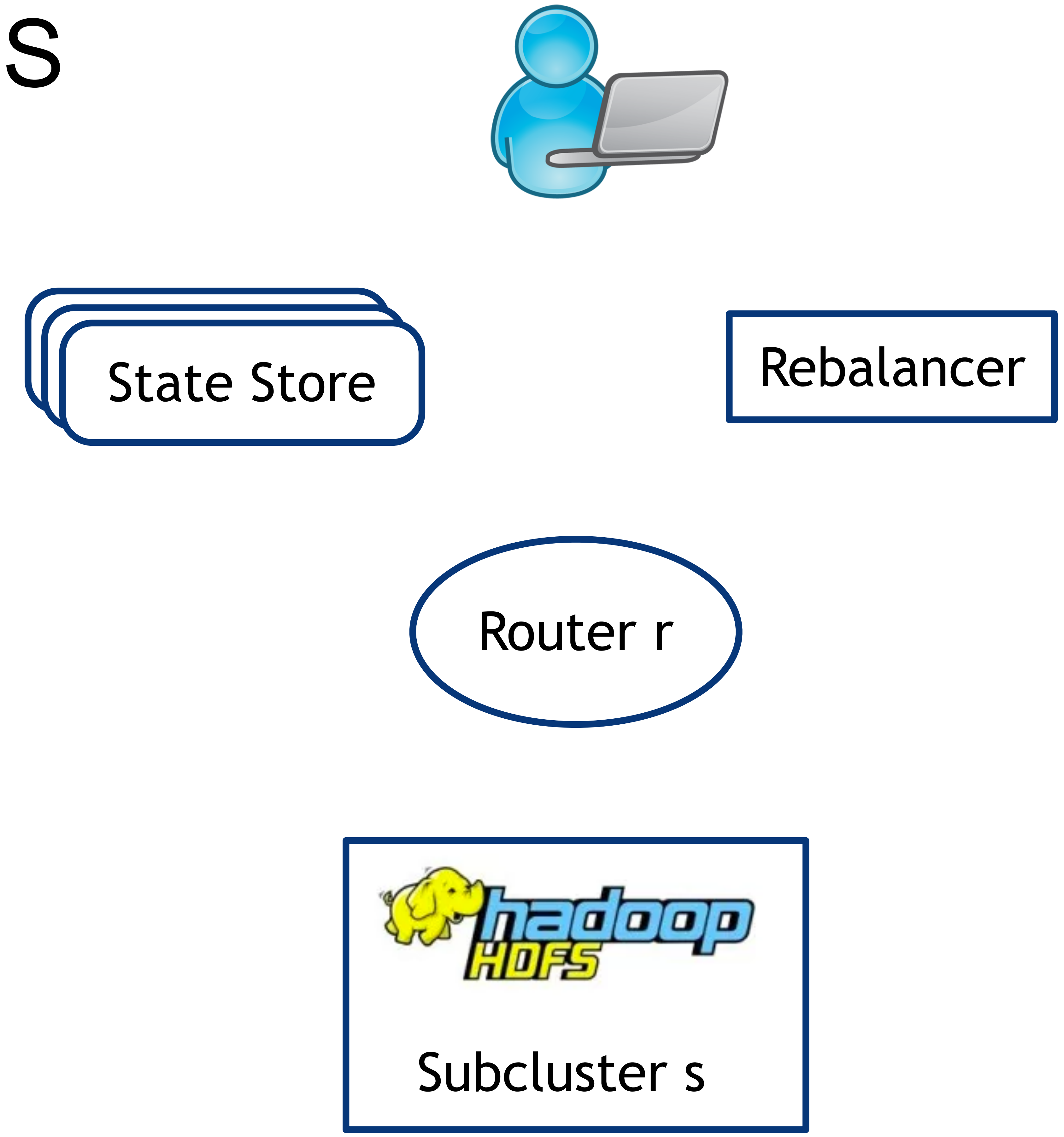
Implementation

- Datacenter-Harvesting HDFS (DH-HDFS)
  - Implement federation architecture over HDFS
  - Diversity-aware replica placement
  - Run independent instances in subclusters
- Load balancing for Routers
- Zookeeper for State Store



Implementation

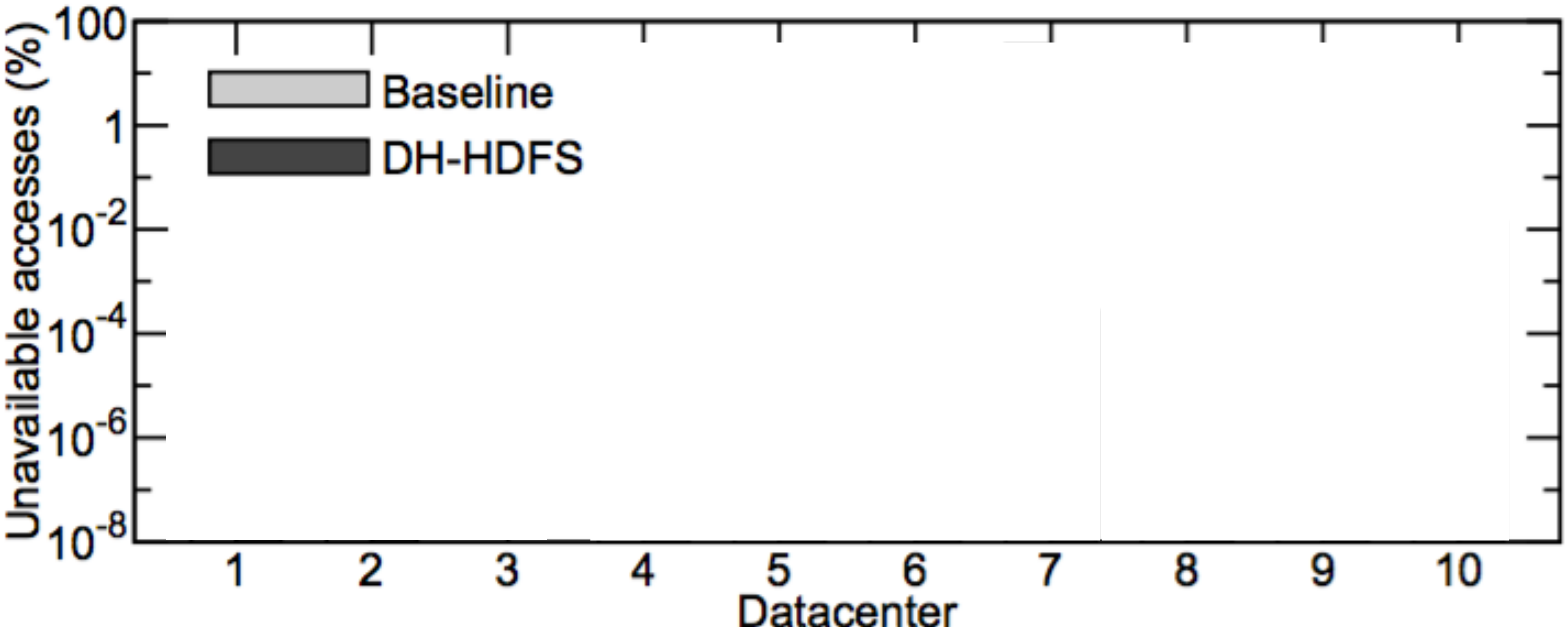
- Datacenter-Harvesting HDFS (DH-HDFS)
  - Implement federation architecture over HDFS
  - Diversity-aware replica placement
  - Run independent instances in subclusters
- Load balancing for Routers
- Zookeeper for State Store
- Rebalancer as a MapReduce job



Evaluation

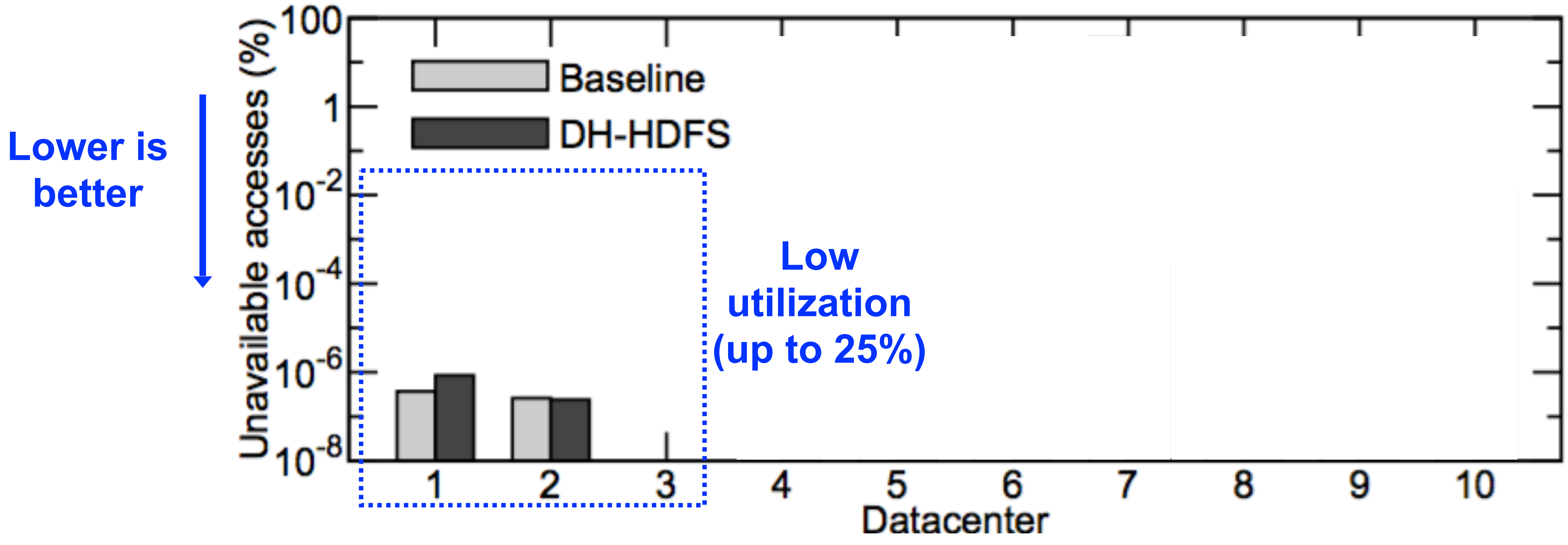
- Real deployment
  - 4k servers divided into 4 subclusters
  - Deployment in production: 30k servers across 4 datacenters
- Large-scale simulation
  - Traces from production datacenters at Microsoft
  - Simulate full datacenters for 6 months
- HDFS trace from Yahoo!
  - 700k files and 4 million accesses

Simulation: Availability (% Successful Accesses)



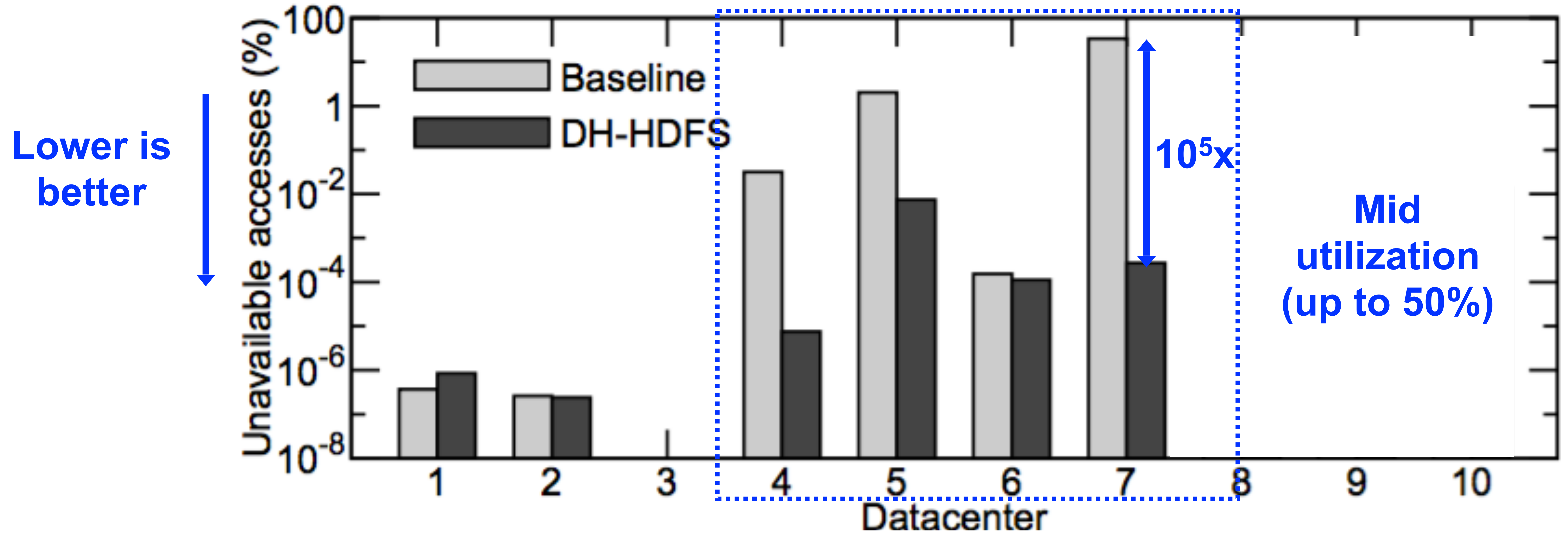
- Baseline system: Groups of primary tenants → subclusters
- Spectrum of primary tenant CPU utilization: low, mid and high
- Significantly higher availability with DH-HDFS
- Improvement in data durability (results in paper)

Simulation: Availability (% Successful Accesses)



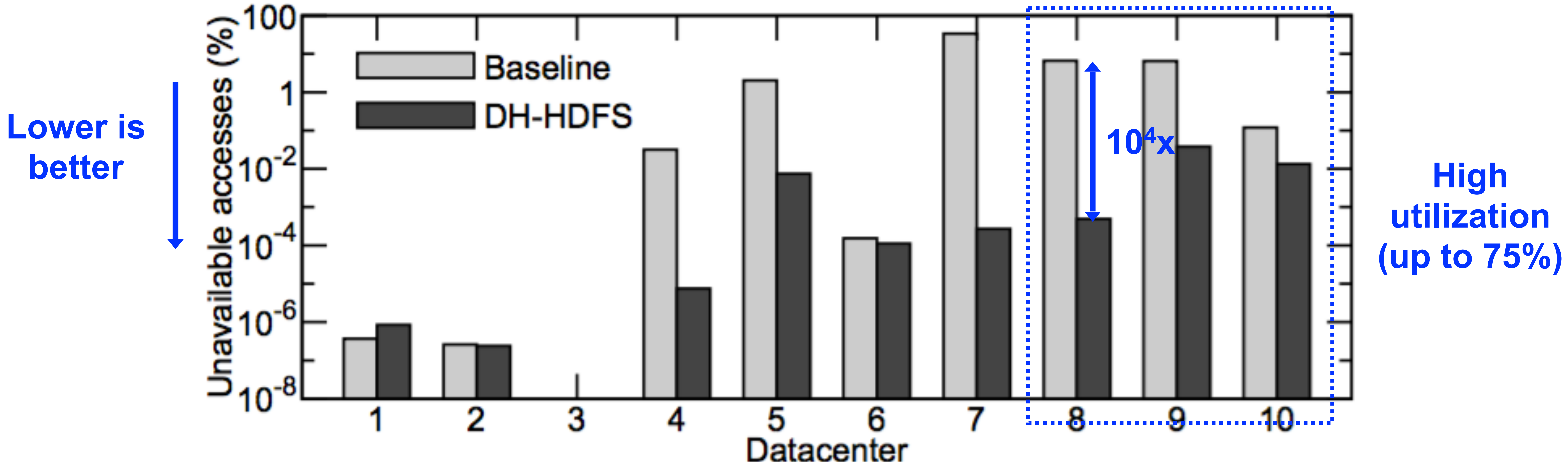
- Baseline system: Groups of primary tenants → subclusters
- Spectrum of primary tenant CPU utilization: low, mid and high
- Significantly higher availability with DH-HDFS
- Improvement in data durability (results in paper)

Simulation: Availability (% Successful Accesses)



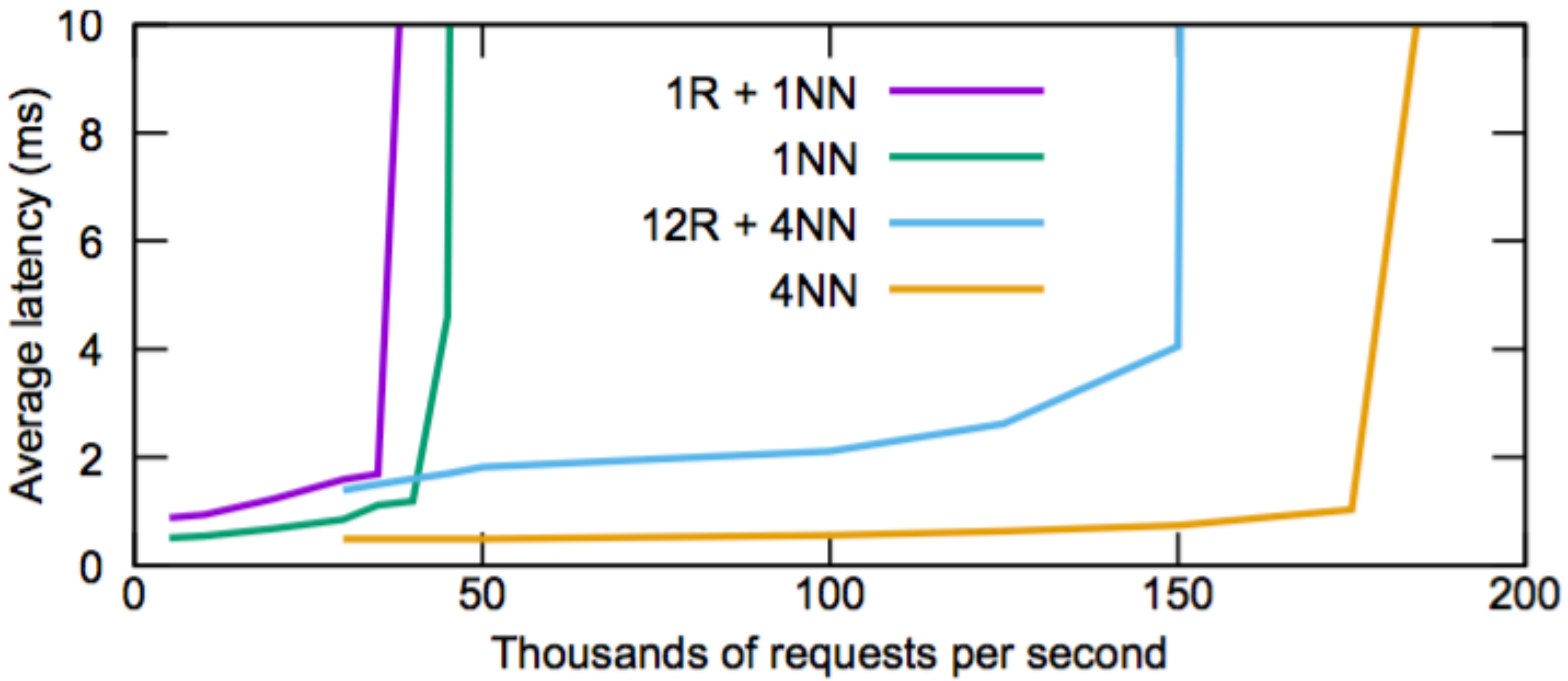
- Baseline system: Groups of primary tenants → subclusters
- Spectrum of primary tenant CPU utilization: low, mid and high
- Significantly higher availability with DH-HDFS
- Improvement in data durability (results in paper)

Simulation: Availability (% Successful Accesses)



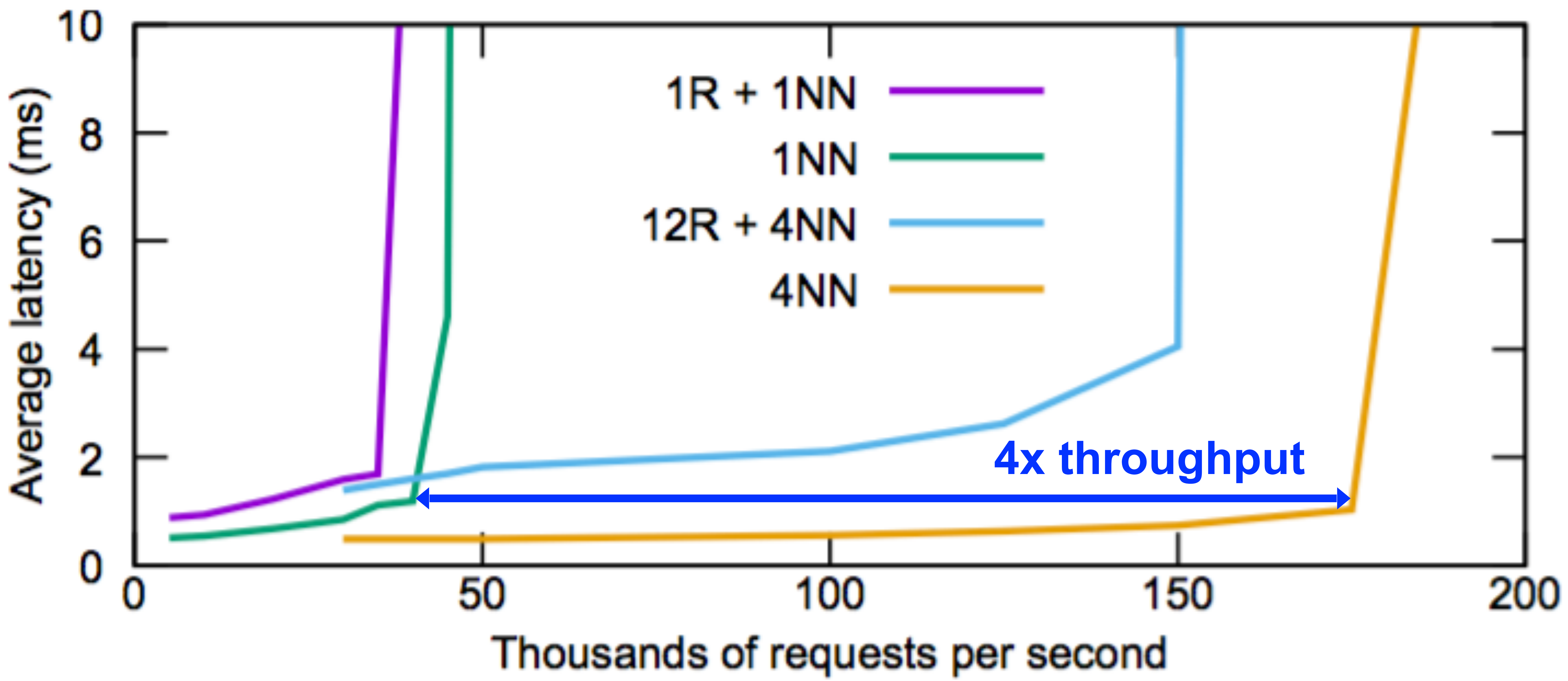
- Baseline system: Groups of primary tenants → subclusters
- Spectrum of primary tenant CPU utilization: low, mid and high
- Significantly higher availability with DH-HDFS
- Improvement in data durability (results in paper)

Real Deployment: Router Performance



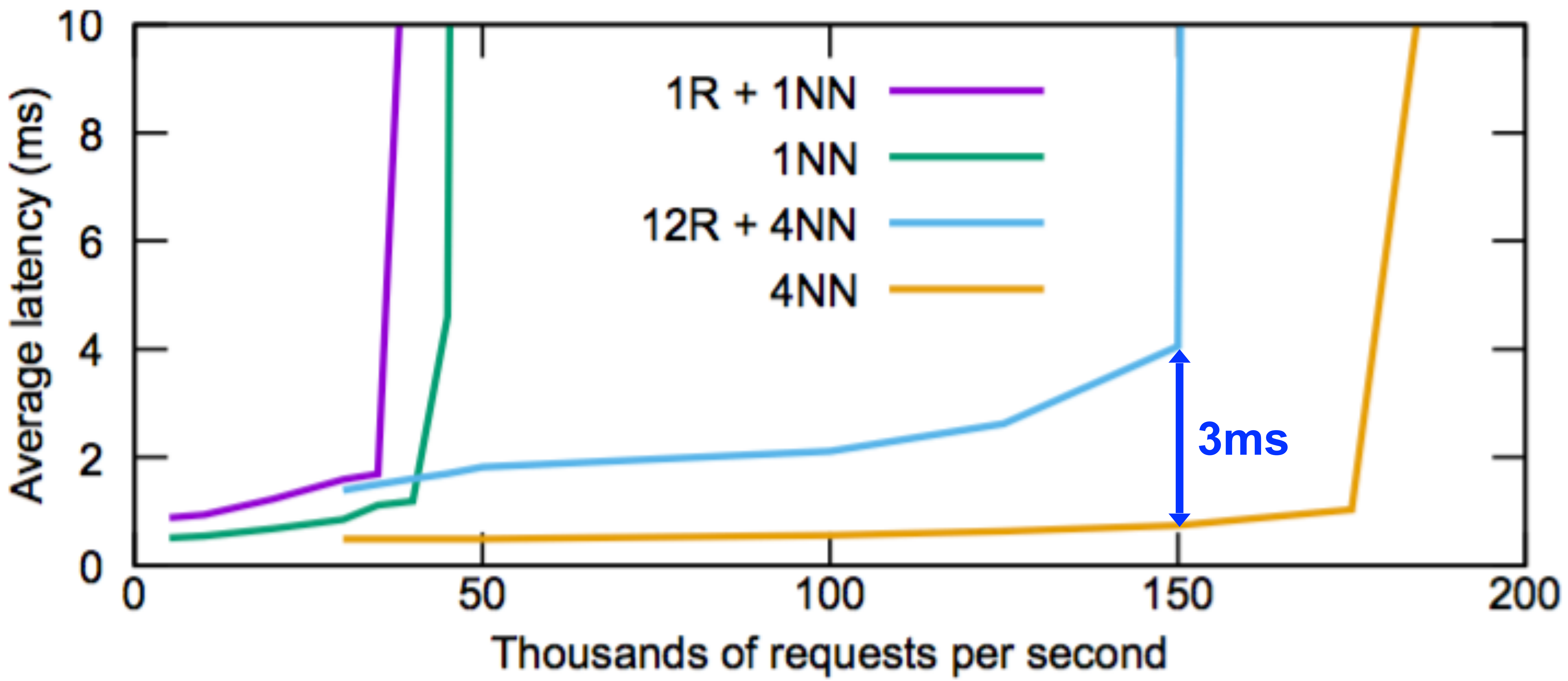
- Worst-case scenario: metadata-only operations workload
- Block read latencies dominate in real-world workloads
- Negligible router overhead in real workloads

Real Deployment: Router Performance



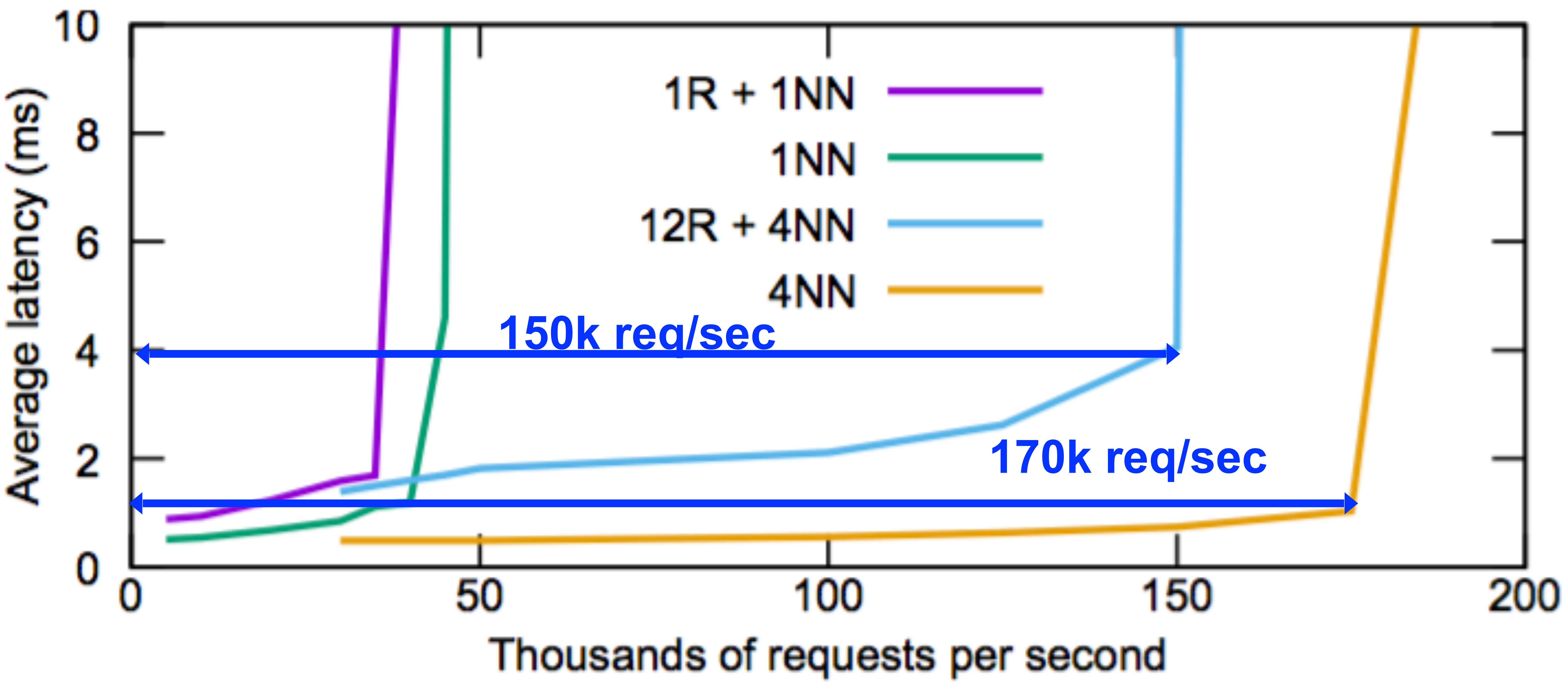
- Worst-case scenario: metadata-only operations workload
- Block read latencies dominate in real-world workloads
- Negligible router overhead in real workloads

Real Deployment: Router Performance



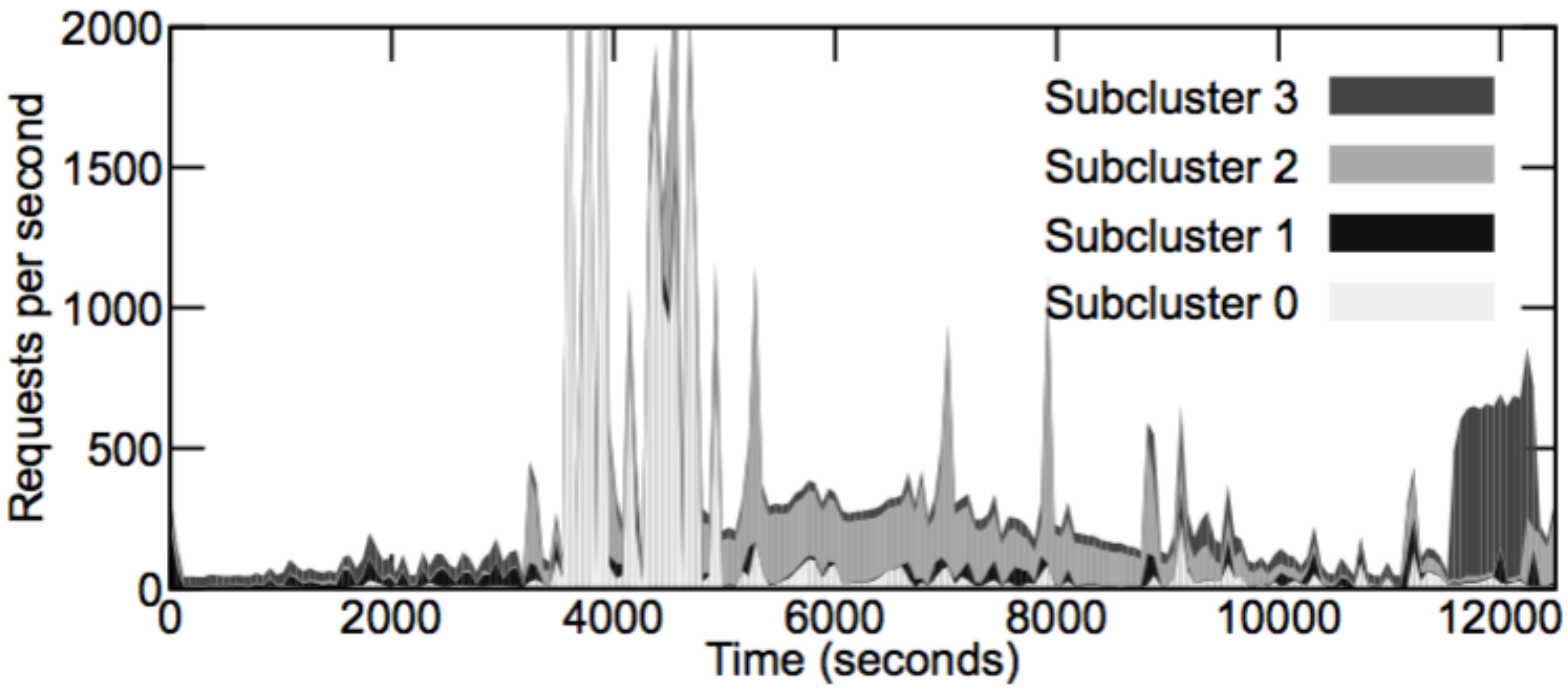
- Worst-case scenario: metadata-only operations workload
- Block read latencies dominate in real-world workloads
- Negligible router overhead in real workloads

Real Deployment: Router Performance



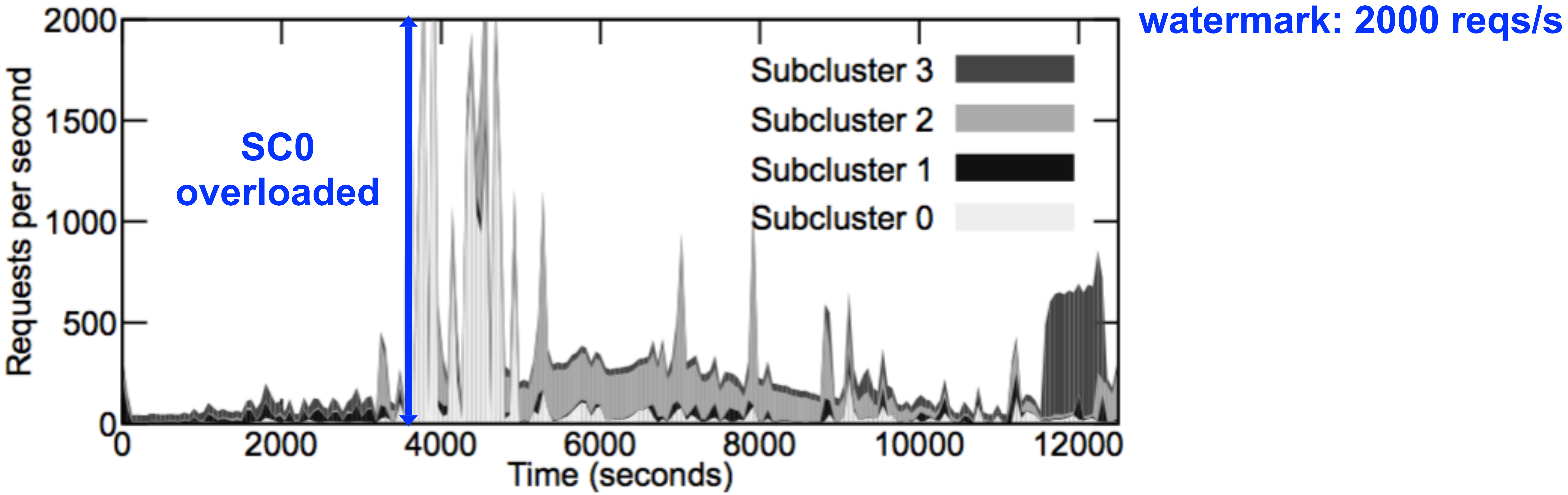
- Worst-case scenario: metadata-only operations workload
- Block read latencies dominate in real-world workloads
- Negligible router overhead in real workloads

Real Deployment: Rebalancer Performance



- 13 TB data moved to balance subcluster 0
- Average rebalance time: 6 mins
  - 100 ms to determine data to move
  - Primary tenant activity impacts data migration time (up to 4x)

Real Deployment: Rebalancer Performance



- 13 TB data moved to balance subcluster 0
- Average rebalance time: 6 mins
  - 100 ms to determine data to move
  - Primary tenant activity impacts data migration time (up to 4x)

Lessons from Production Deployment

- 30k servers spread across 4 datacenter
- Bootstrapping server → subcluster assignment
  - Switch to consistent hashing caused massive reshuffling of servers
  - Restrict movement till servers are re-imaged or decommissioned
- Spread large data across subclusters
  - Users wanted data of batch jobs in a single folder
  - Create special folders with files distributed across subclusters
- More lessons in the paper

Conclusion

- Scale file systems to entire datacenter
- Datacenter-Harvesting HDFS
  - Runs independent subclusters → isolation
  - Federates subclusters transparently → global namespace
  - Higher durability and availability on harvested resources
  - Better file access performance via rebalancing
- Deployed in production datacenters
  - 30k servers spread across 4 datacenters

Questions?

- Thanks!