MicroMon: A Monitoring Framework for Tackling Distributed Heterogeneity

Babar Khalid*+, Nolan Rudolph†+,

Ramakrishnan Durairajan†, Sudarsun Kannan*

*Rutgers University, †University of Oregon

(+co-primary authors)

Background

- Modern applications are increasingly becoming geo-distributed
  - e.g., Cassandra, Apache Spark

- Geo-distributed datacenters (DCs) use heterogeneous resources
  - storage heterogeneity (e.g., SSD, NVMe, Harddisk)
  - WAN heterogeneity (e.g., fiber optics, InfiniBand)

- Hardware heterogeneity in DCs avoids vendor lockout and reduces operational cost (by combining older/cheaper and newer/expensive hardware)

- Careful provisioning can provide high performance at lower cost

Problem With Current Systems

- Current monitoring frameworks for geo-distributed applications are unidimensional
  - can only monitor hosts, storage devices, networks in isolation

- Lack hardware heterogeneity awareness
  - e.g. no awareness for storage heterogeneity
  - could impact I/O intensive applications

- Coarse-granular monitoring
  - unaware of host-level micro-metrics in software and hardware
  - e.g. page cache, node-level I/O traffic, node's network queue delays

Our Solution - MicroMon

- MicroMon is a fined grained monitoring, dissemination, and inference framework
- Collects fine-grained (**micrometrics**) software and hardware metrics in end-hosts and network
    - e.g., page cache utilization, disk read/write throughput in end host
- Filters micrometrics into anomalies to efficiently disseminate
- Enables replica selection for geo-distributed Cassandra
- Preliminary study of Micromon integrated with geo-distributed Cassandra shows high throughput gains

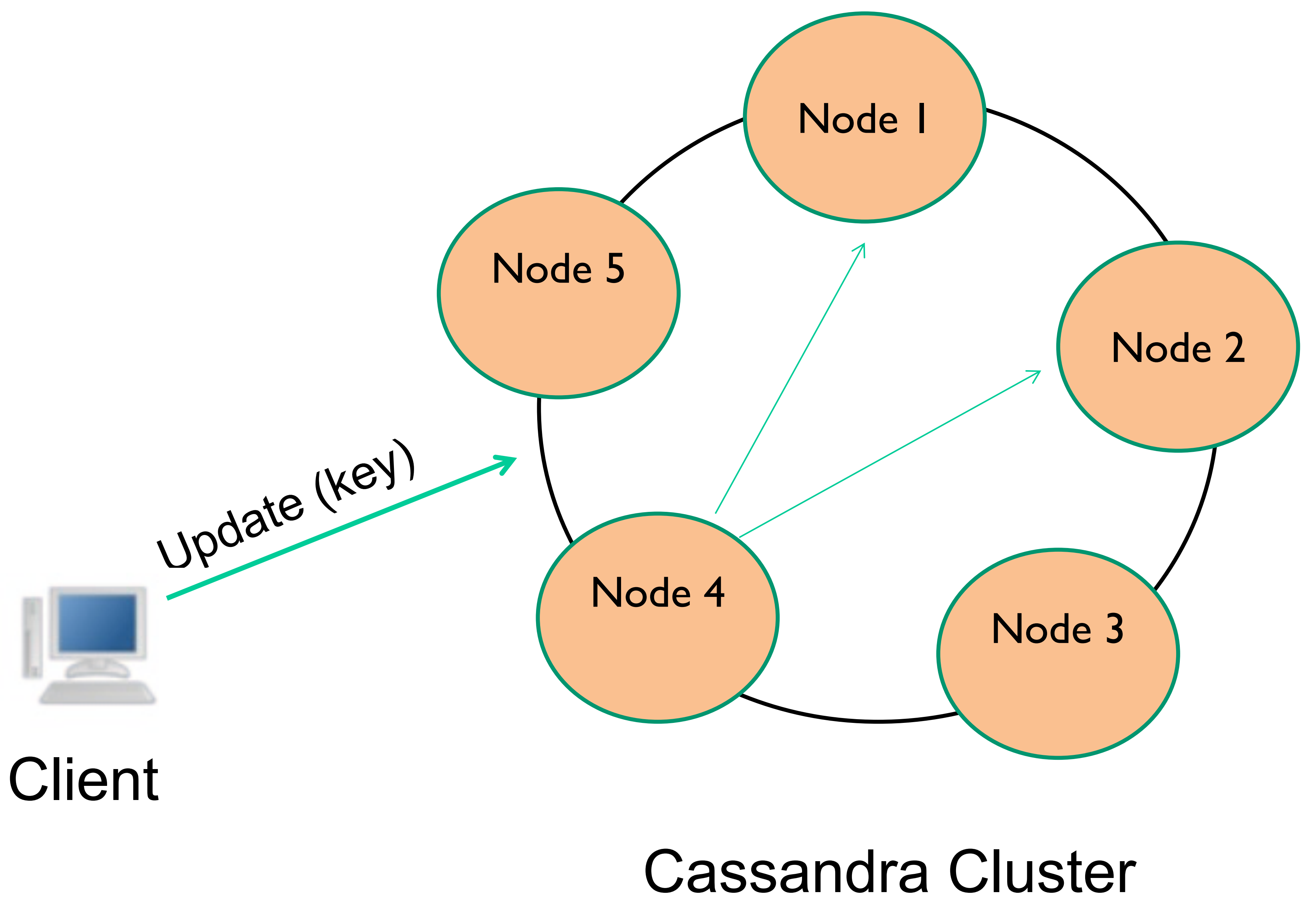
Outline

- Background
- Case Study
- Design
- Evaluation
- Conclusion

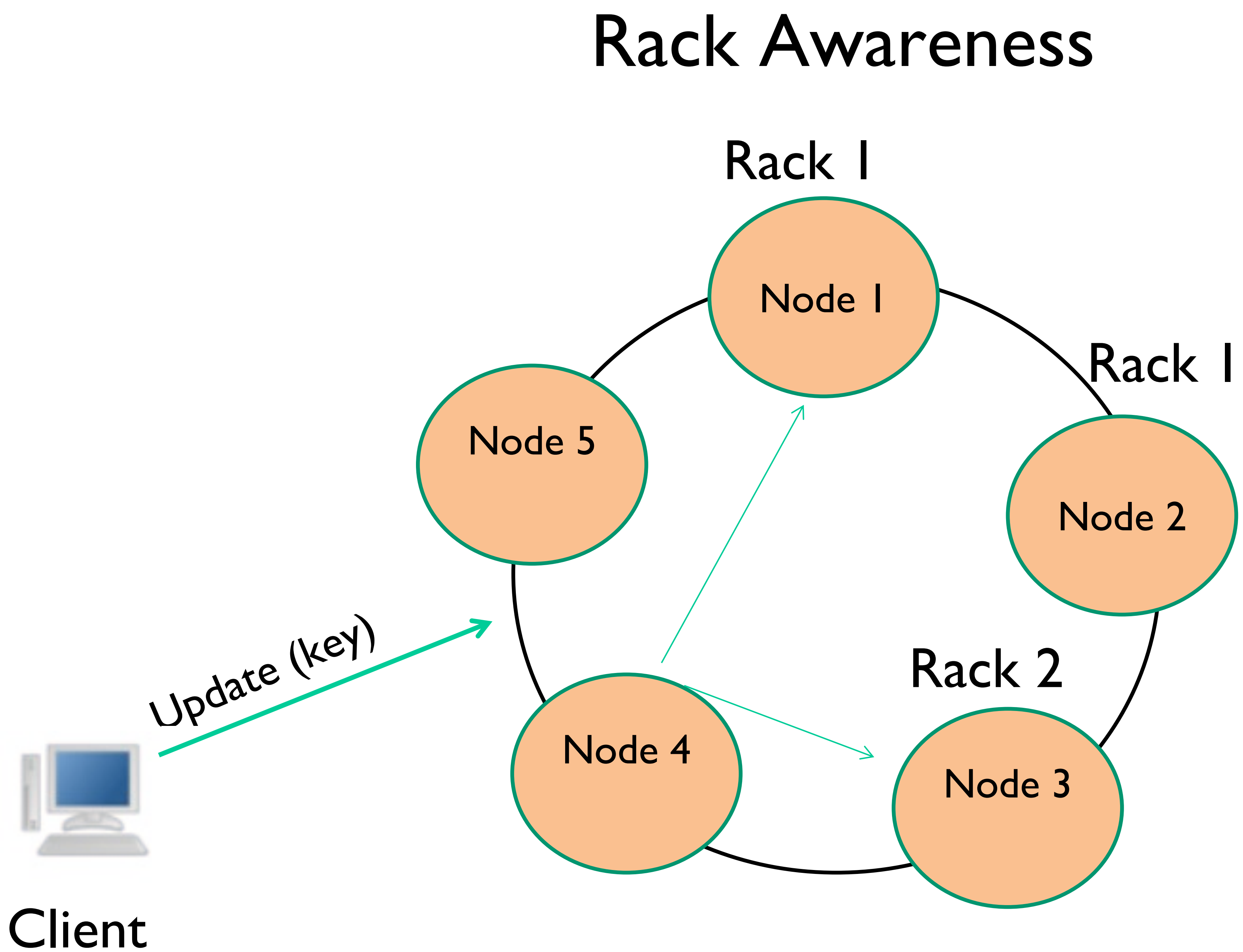
Case Study - Cassandra

- Distributed NoSQL database system deployed geographically
- Manages large amounts of structured data in commodity servers
- Provides highly available service and no single point of failure
- Typically focuses on availability and partition tolerance

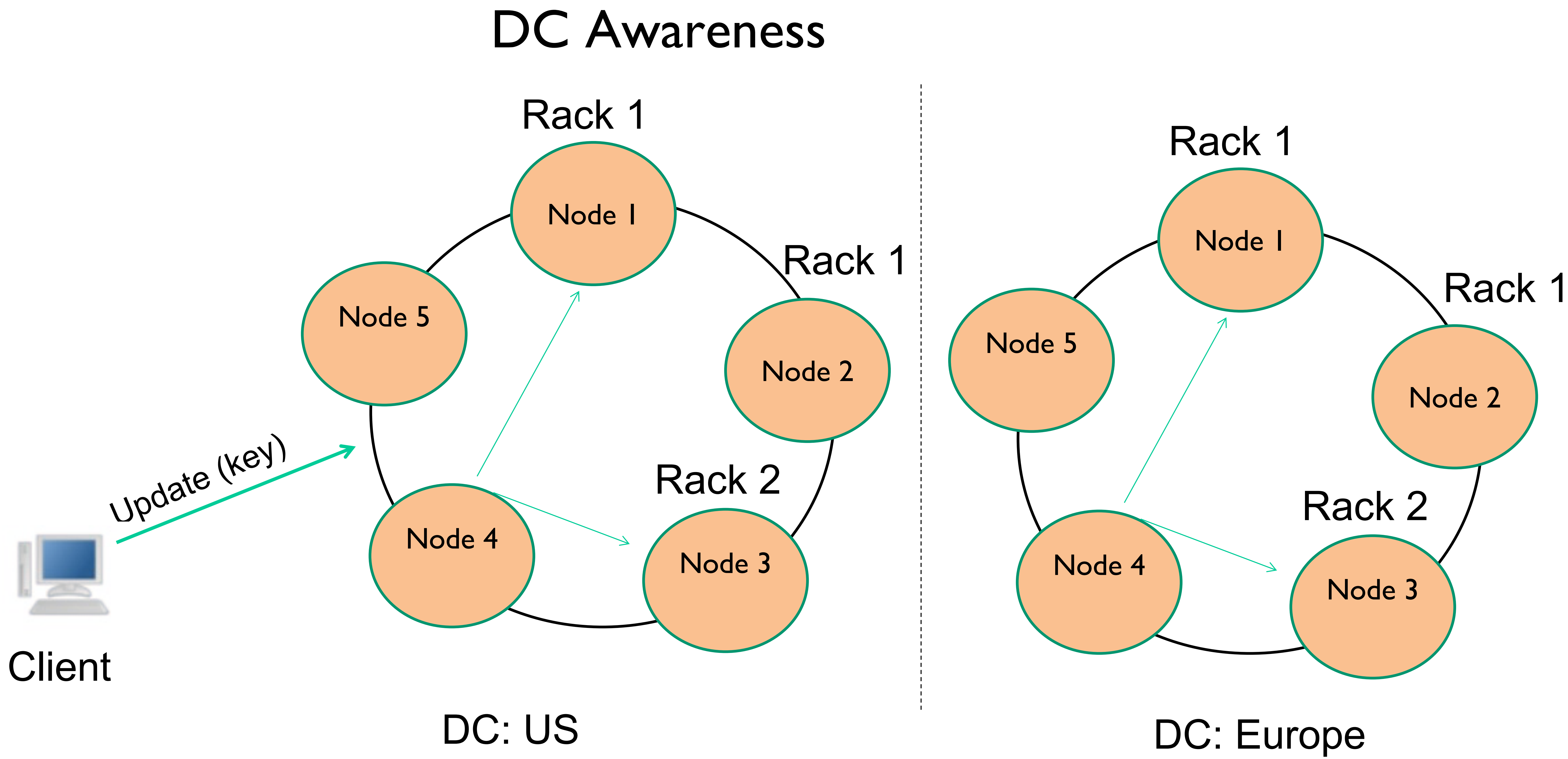
Cassandra – Replication



Cassandra – Replication



Cassandra – Replication



Cassandra's Snitch Monitoring

- Cassandra uses *Snitch* to monitor network topology and route requests across replicas
- Also provides capability to spread replicas across DCs to avoid correlated failures
- Snitch monitors (read) latencies to avoid non-responsive replicas
- Different types: Gossiping, MultiRegionSnitch

  - Gossiping uses rack and datacenter information to gossip across nodes and collect latency information

- Problem: No hardware heterogeneity awareness

Analysis Goal and Methodology

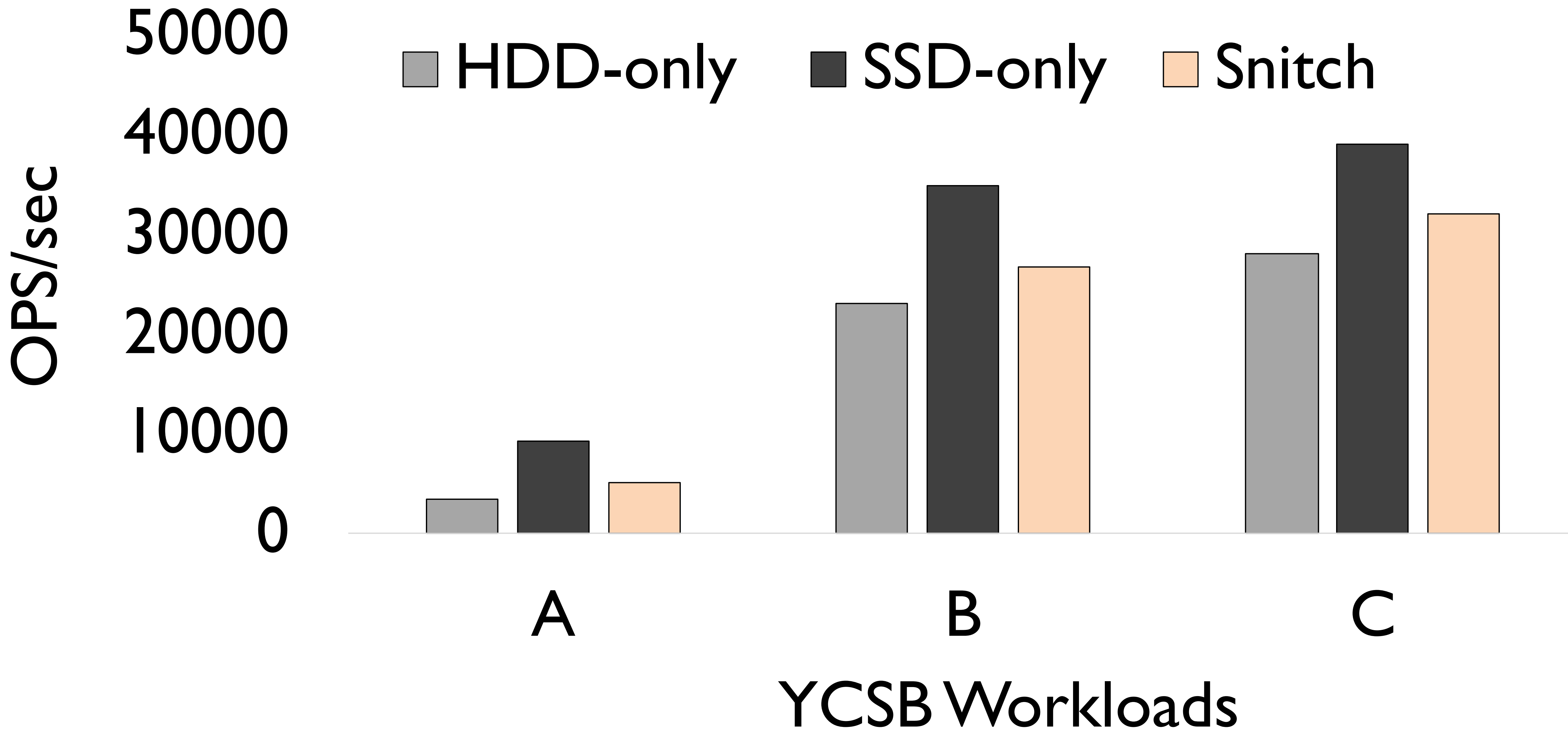
- Goal: Highlight the lack of heterogeneity awareness
- Replica Configuration

  - SSD Replica: Sequential storage b/w - 600MB/s, rand b/w: 180 MB/s
  - HDD replica: Sequential storage b/w - 120MB/s, rand b/w: 10 MB/s

- Network latency across replicas same (for this analysis)
- Workload – YCSB benchmark

  - workload A (50% read and writes)
  - workload B (95% reads)
  - workload C (100% reads)

Impact of Storage Heterogeneity Awareness



- Significant performance impact over optimal SSD-only configuration
- Snitch: Lack of awareness to storage hardware heterogeneity

Outline

- Background
- Case Study
- Design
- Evaluation
- Conclusion

Our Design: MicroMon

- Monitoring and inference framework for geo-distributed applications
- Performs **micro-metrics** monitoring at the host and network-level
  - micro-metrics includes fine-grained software and hardware metrics
- Efficiently disseminates collected micro-metrics
- Ongoing - Distributed inference engines to guide application requests to the best replica

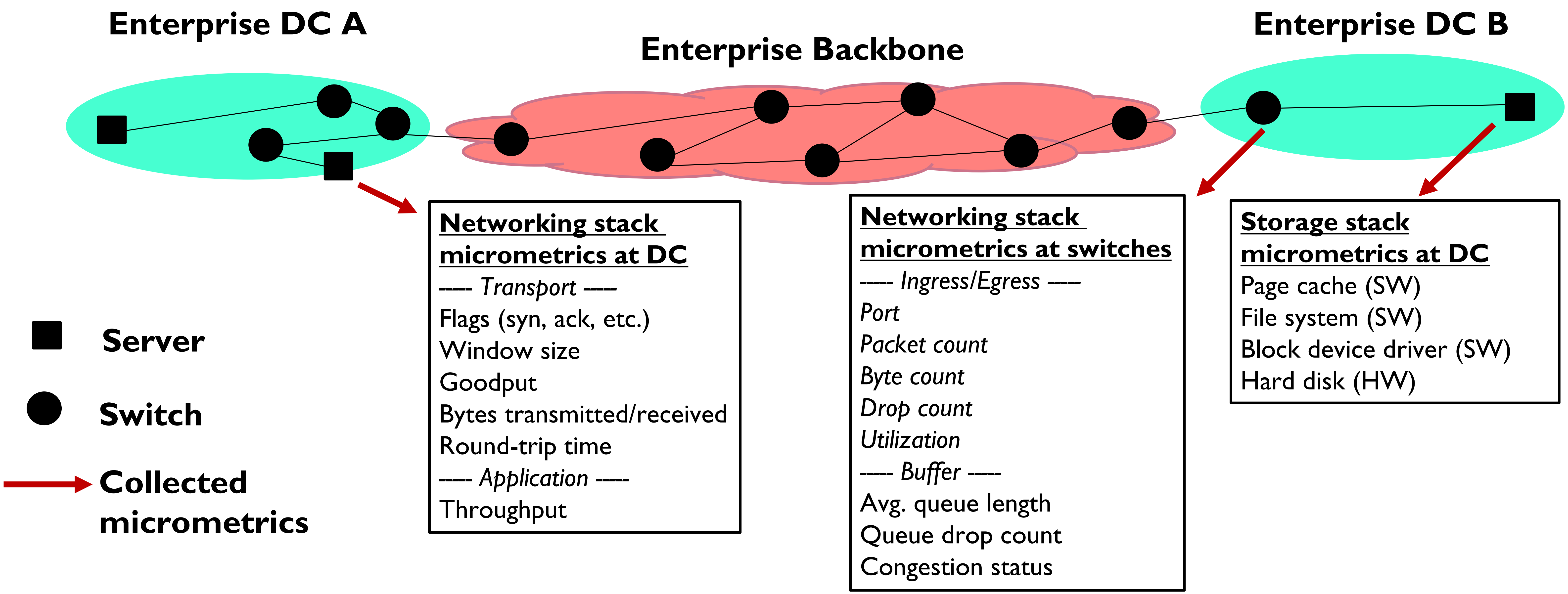
MicroMon Challenges

- **Selection Problem: What micrometrics to consider?**
- **Dissemination Problem: How to send all micrometrics?**
- **Inference Problem: How to quickly infer from micrometrics?**

Design - Micrometrics Selection

- Huge combinations of micrometrics across app, host OS, and network
- Micrometrics could vary for different application-level metrics
  e.g. micrometrics for latency different than those for throughput
- Our approach: Start with storage and network micrometrics
- Identify hardware and software micrometrics using resource usage
  - e.g. high storage usage -> monitor page cache, read/write latency

MicroMon High-level Design



Reducing Dissemination – Anomaly Reports

- Problem: Prohibitive cost of dissemination across thousands of nodes
  - cost increases with hardware and software components
- - e.g., SSD's SMART counters contain close to 32 counters

- Observation: OSes already expose anomalies (indirectly)
  - e.g. high I/O wait time of process -> higher page cache misses
  - e.g. sustained storage BW against max. hardware BW
  - e.g. network I/O queue wait time alludes to TCP congestion

- Proposed Idea: Instead of sending thousands of micrometrics to decision agent, only report OS perceived anomalies

Reducing Dissemination - Network Telemetry

- Network telemetry offers aggregated stats about state of the network
- Idea: co-design in-band network telemetry (INT) with end host OS
  - monitor packets at end host with anomaly reports as payload
  - get network anomaly reports using INT
- Pre-established anomaly thresholds reduce total aggregated stats further

INT header	INT payload
Network anomalies	End-host anomalies

Scalable Inference - Scoring-based Inference

- Simple scoring–based inference in Cassandra
  - replicas sorted and ranked by network latency
- Problem: for bandwidth sensitive applications, need higher weights for WAN-based micrometrics compared to host-level micrometrics
- Our approach:
  - we assign equal weights to all software and hardware micrometrics
  - use collected micrometrics to calculate a replica score
  - route request to replicas with higher scores
  - flexibility to assign higher weights for WAN-based micrometrics
- Ongoing: Designing a generic, self-adaptive inference engine

Outline

- Background
- Case Study
- Design
- Evaluation
- Conclusion

Evaluation Goals

Goals:

- Understand the impact of storage heterogeneity with Micromon
- Understand the impact of storage heterogeneity + network latency
- Analyze the page cache impact (see paper for details)

Analysis Methodology

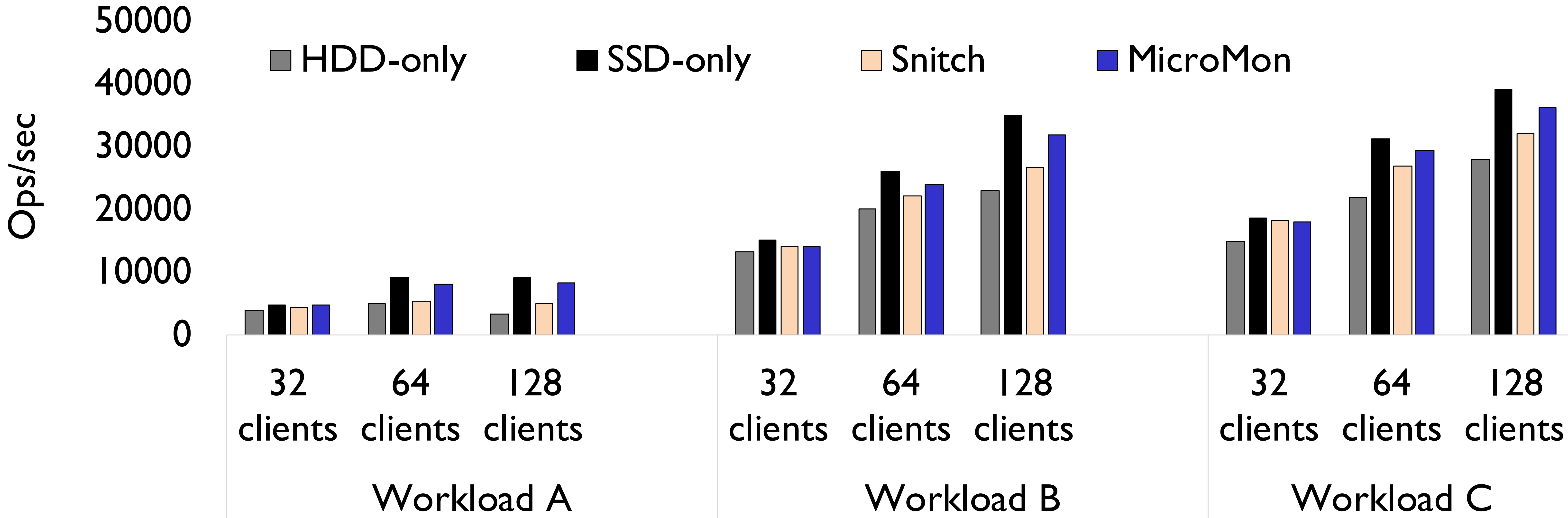
- Multiple DCs from CloudLab Infrastructure
  - three nodes located in UTAH, APT, and Emulab DCs

- Replica Configuration
  - UTAH replica: NVMe storage (seq bw: 600MB/s, rand bw: 180 MB/s)
  - APT replica: HDD (seq bw: 120 MB/s, rand bw: 10 MB/s)
  - Emulab master node: HDD (same as above)

- Network Latencies
  - 400us between UTAH (NVMe) replica and master node
  - 600us between APT (HDD) replica and master node

- Workload – YCSB benchmark
  - workload A (50% read and writes)
  - workload B (95% reads)
  - workload C (100% reads)

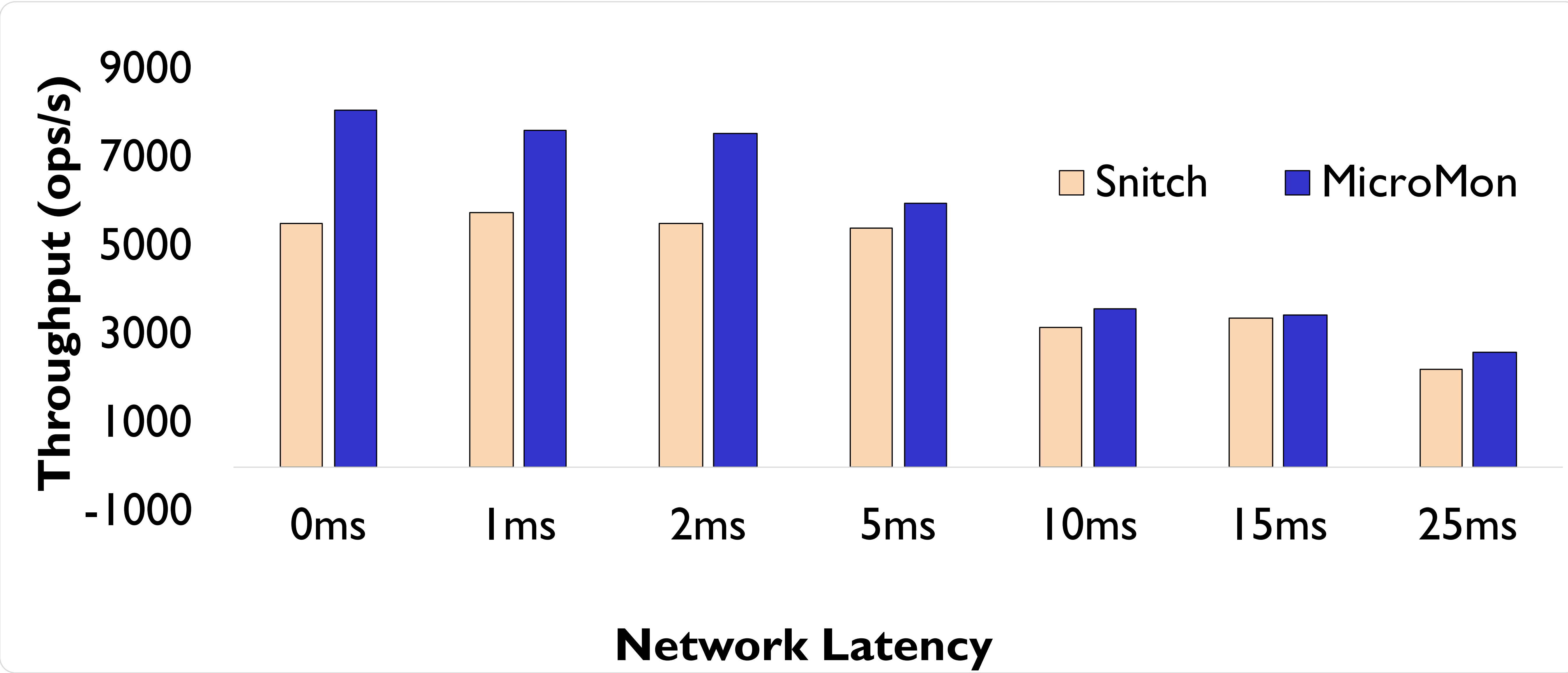
MicroMon's - Storage Heterogeneity



- Snitch lacks storage heterogeneity awareness
- MicroMon's storage heterogeneity awareness provides performance close to SSD-only (optimal) configuration
- Performance improves by up to 49% for large thread configuration

Storage Heterogeneity + Network Latency

- Introduce network latency for SSD-only node



- For high network latencies (e.g., beyond 10ms) SSD benefits reduce

Conclusion

- Datacenter systems are becoming more and more heterogeneous
- Deploying geo-distributed applications in heterogeneous datacenters requires redesign of monitoring mechanisms
- We propose MicroMon, a fine-grained micrometric monitoring, dissemination, and inference framework
- Our on-going work will focus on efficient dissemination and self-adaptive inference mechanisms

Thanks!

Questions?

Contact:

sudarsun.kannan@rutgers.edu

ram@cs.uoregon.edu