Towards Fair Sharing of Block Storage in a Multi-tenant Cloud

Xing Lin, Yun Mao*, Feifei Li, Robert Ricci

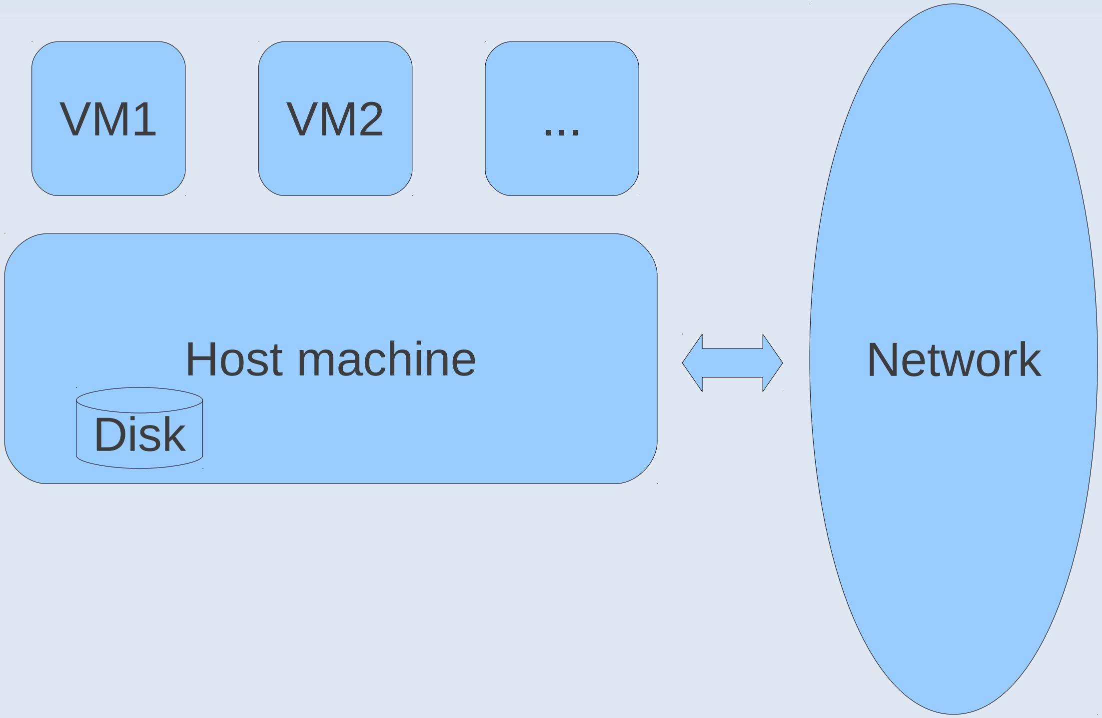
THE UNIVERSITY OF UTAH

* at&t

Cloud Computing

Key Idea: Resource Sharing

- Ecomonies of scale
- High utilization



Typical setup

Performance Unpredictability

Sharing results in interference

- Listed as the Number 5 obstacle for Cloud Computing (Above the Cloud: a Berkeley View of Cloud Computing)
- CPU and memory sharing work well in practice
- A dedicated session for network performance yesterday
- Here, we are looking into disk I/O sharing

Disk I/O Sharing

Disk I/O sharing is problematic

- Interference between random and sequential workloads
- Conflicts between read and write workloads

Can we build a cloud storage system with more predictable performance?

Interference Analysis - Workloads

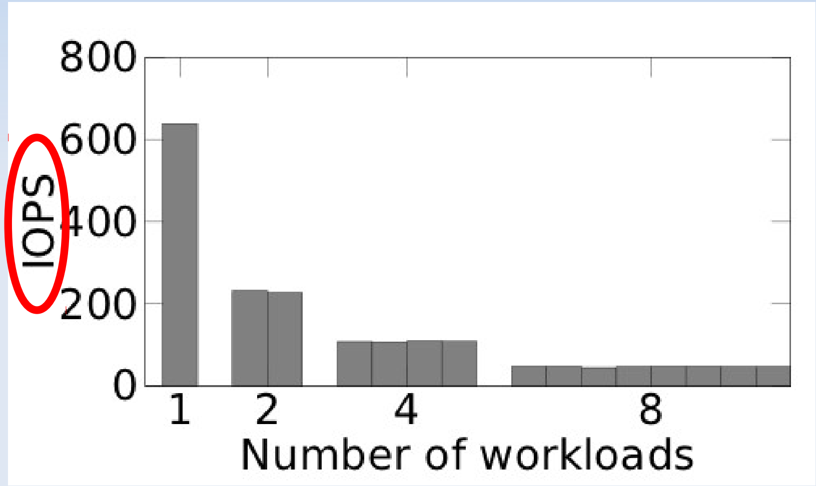
- Use FIO to investigate interference between:
  - Random Read(RR)
  - Sequential Read(SR)
  - Random Write(RW)
  - Sequential Write(SW)
- Real-world application
  - TPC-H

Interference Analysis - Setup

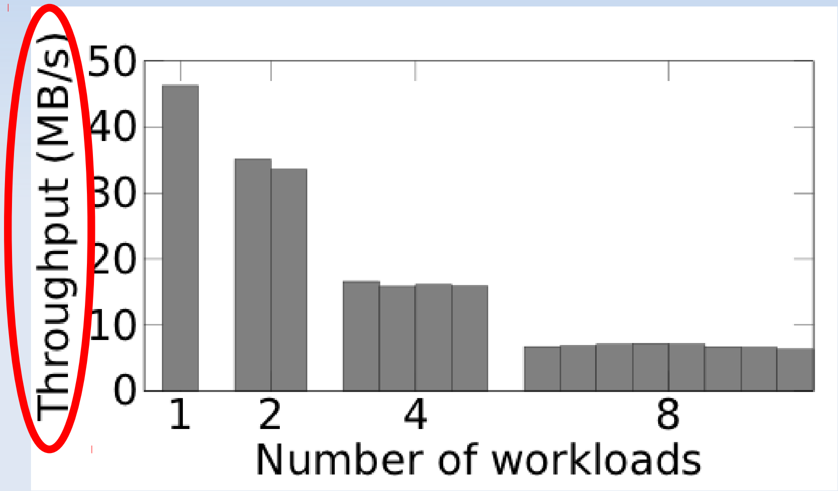
- Disk: Seagate Cheetah 10,000 RPM 146 GB SCSI disk(pc3000 in Emulab)
- FIO benchmarks
  - 10 GB partitions
  - Direct IO
  - Block size: 4 KB
  - IO depth: 32
  - Runtime: 120 s
  - Metrics: IOPS for random workloads and throughput for sequential workloads

Interference Analysis Result - I

Co-locating same type of workloads



Random Read



Sequential Write

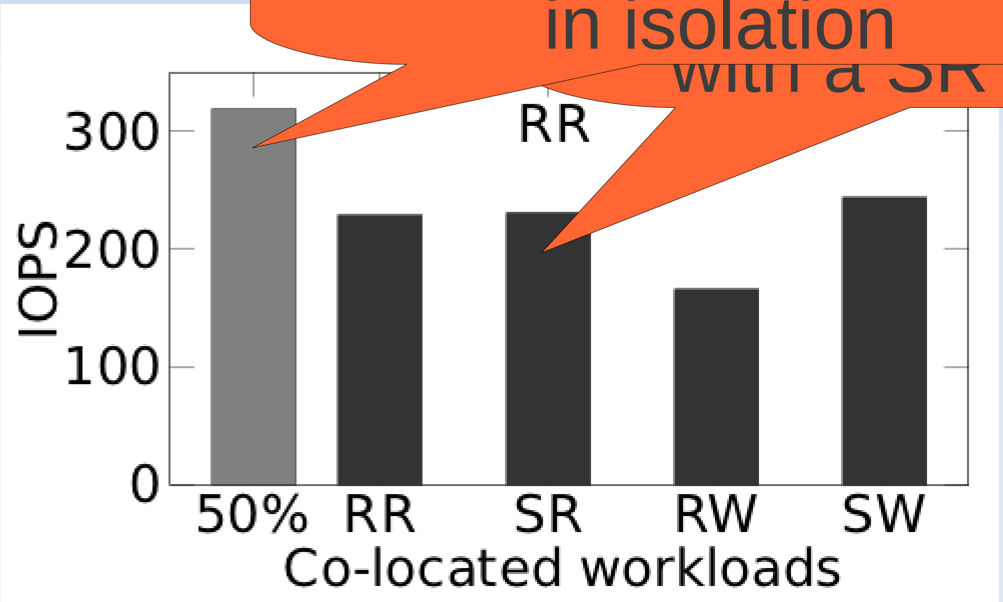
Observation1: When co-locating the same type of workloads, each workload gets a fair share in performance and system resources.

Interference Analysis Results - II

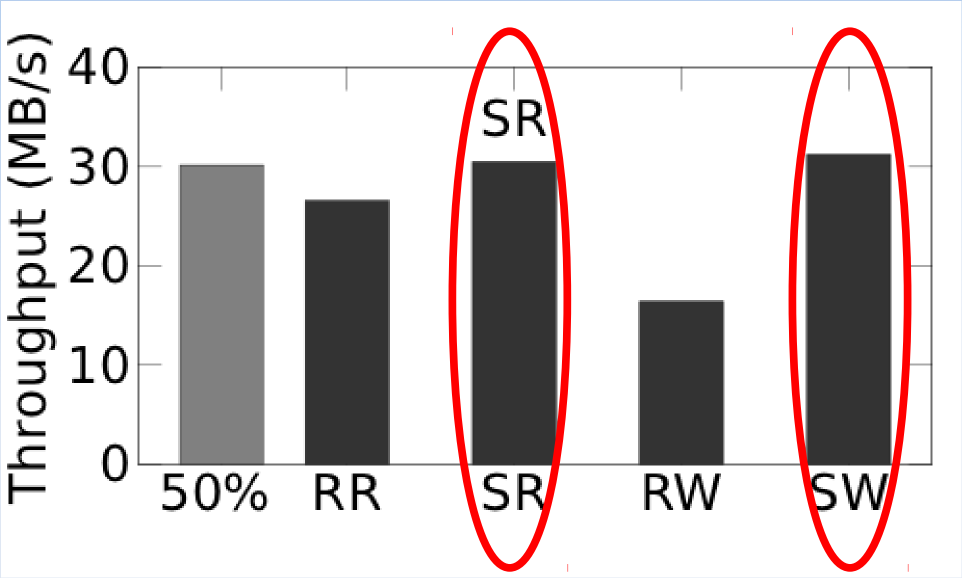
Co-locating different ty

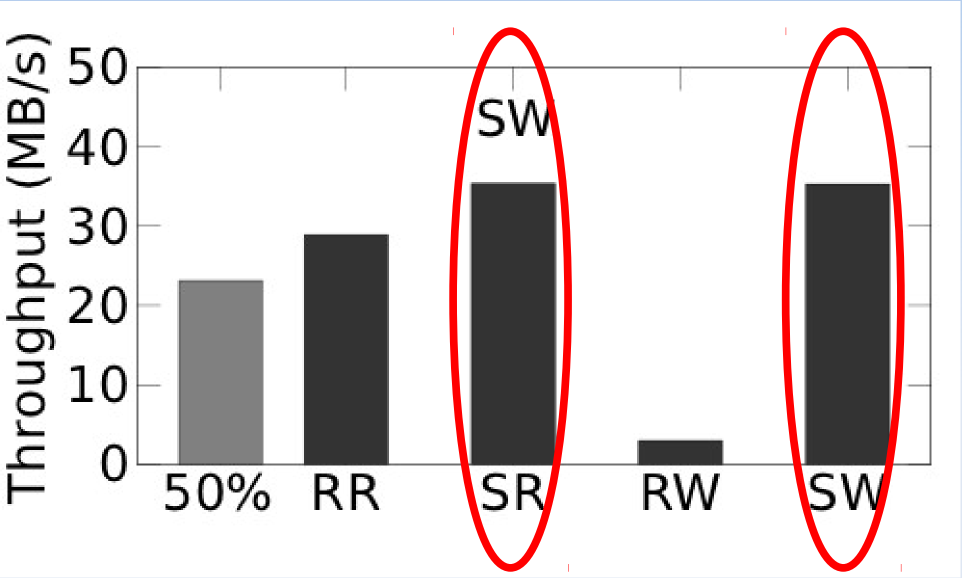
50% of the performance of a RR workload when run in isolation

f a RR
s run
with a SR workload



Interference Analysis Results - II





Sequential workloads

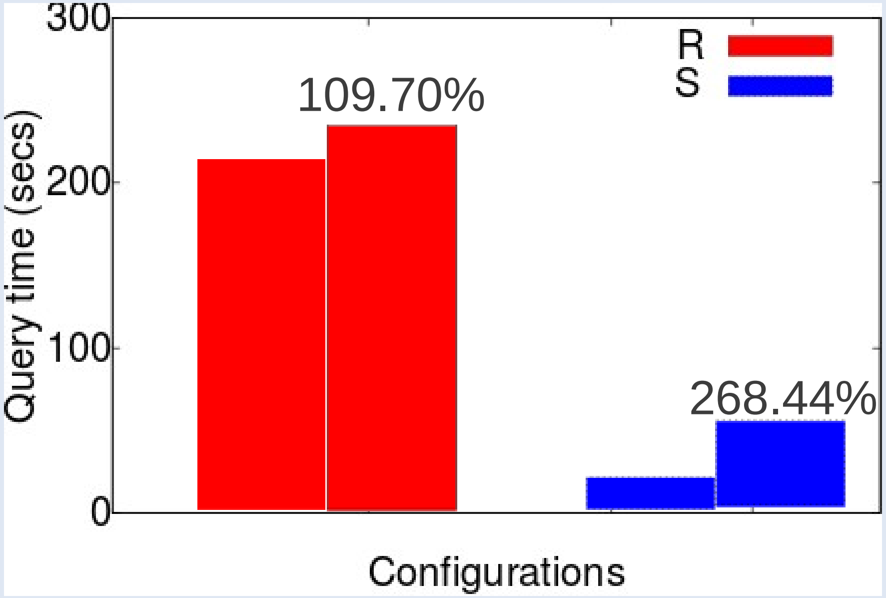
Observation2: Random workloads are destructive to sequential workloads.

Interference Analysis Results - II

Observation3: Random write workload is destructive for all other types of workloads.

Interference Analysis Result - III

- Real-world application: TPC-H
  - 21 TPC-H queries(random read)
  - sequential scan of 9 tables(sequential read)

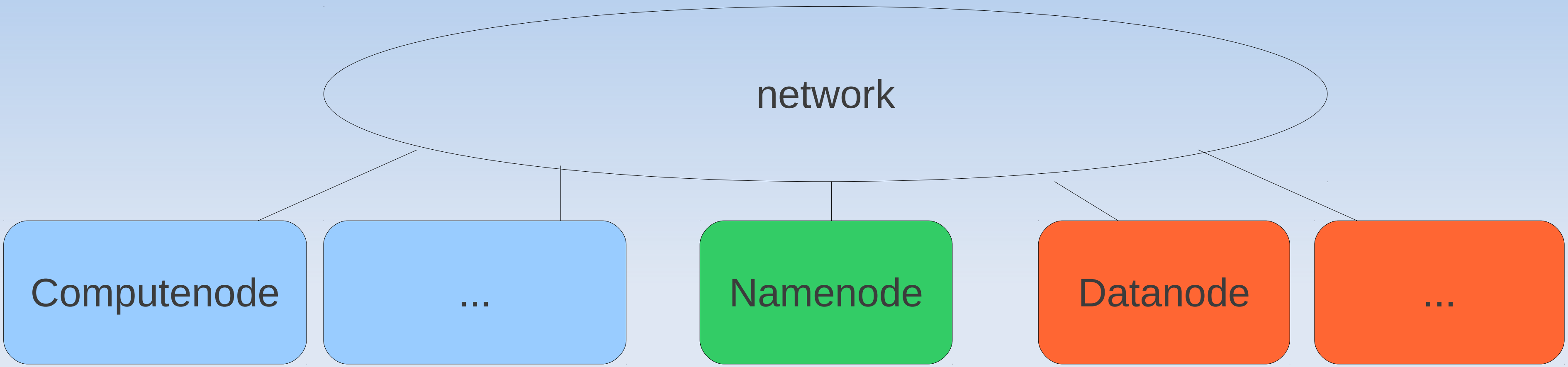


FAST – Fair Assignment for Storage Tenants

Goal: want to build a block storage system, similar to Amazon EBS, with more predictable performance

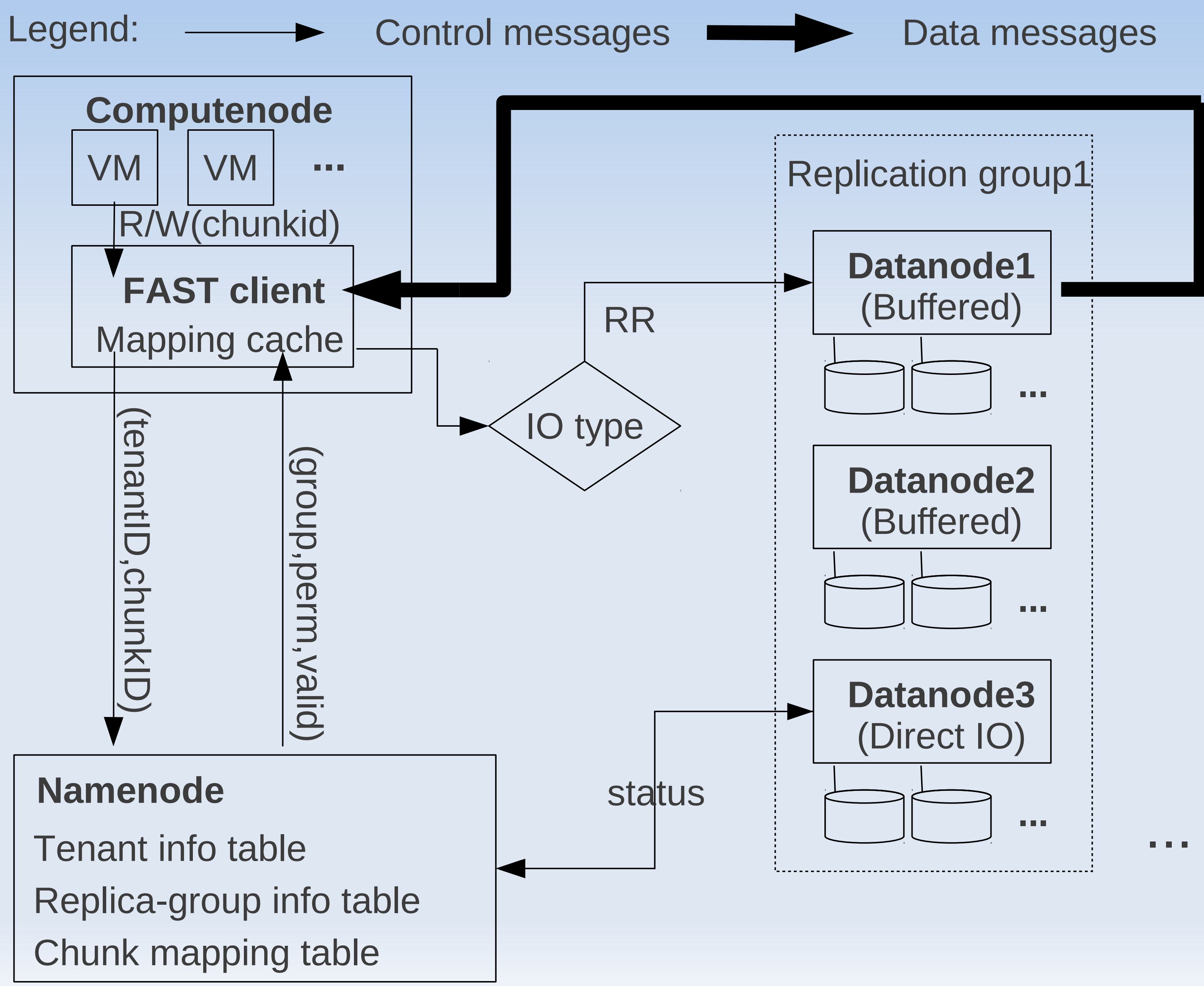
- Assumptions
  - Inexpensive commodity components: replication
  - Exclusive ownership of a virtual volume
  - No assumption about workloads within VM

FAST – System Design

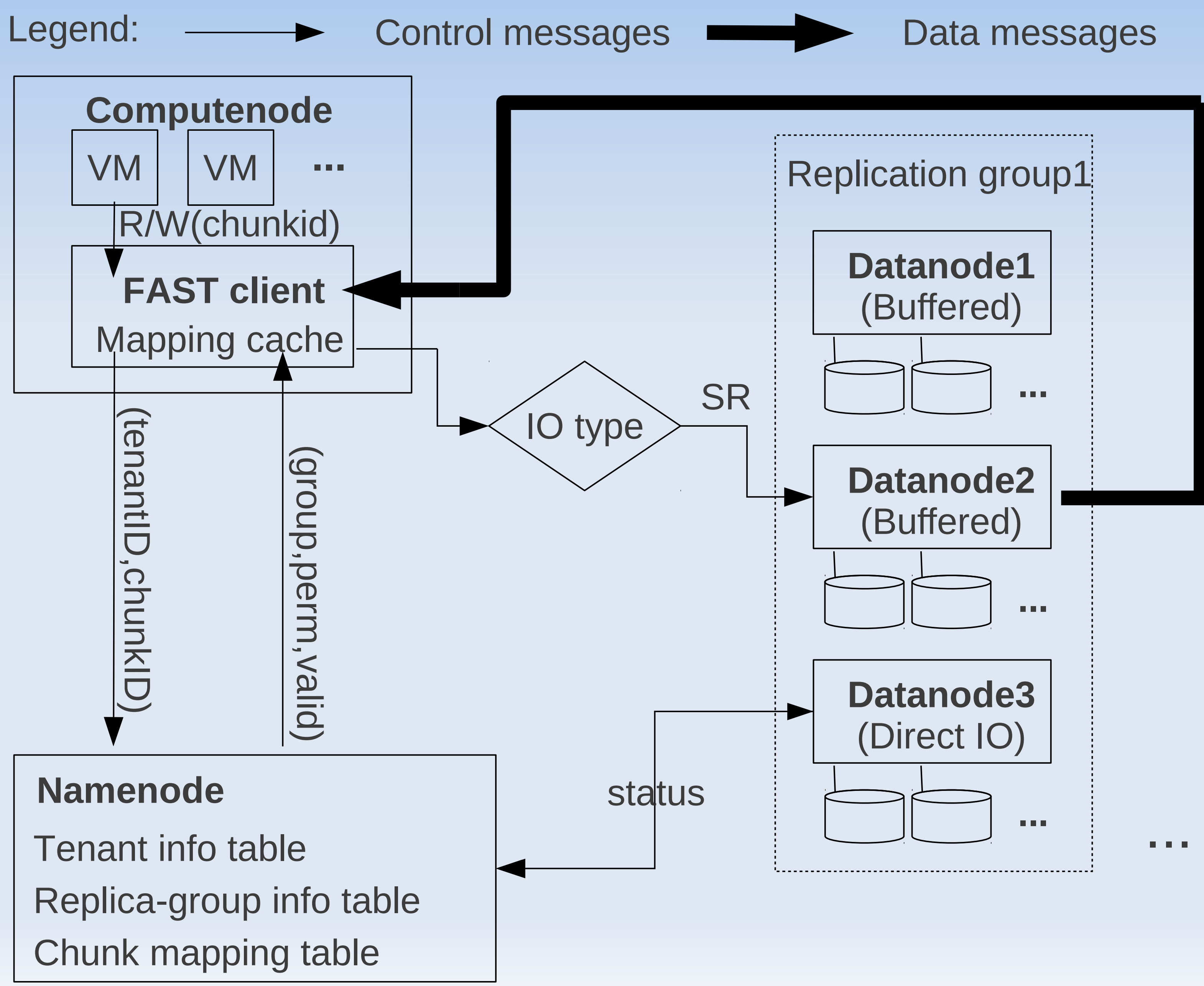


- System Design:
  - Directs random reads and sequential reads to different replicas
  - Log-structure to convert random writes into sequential

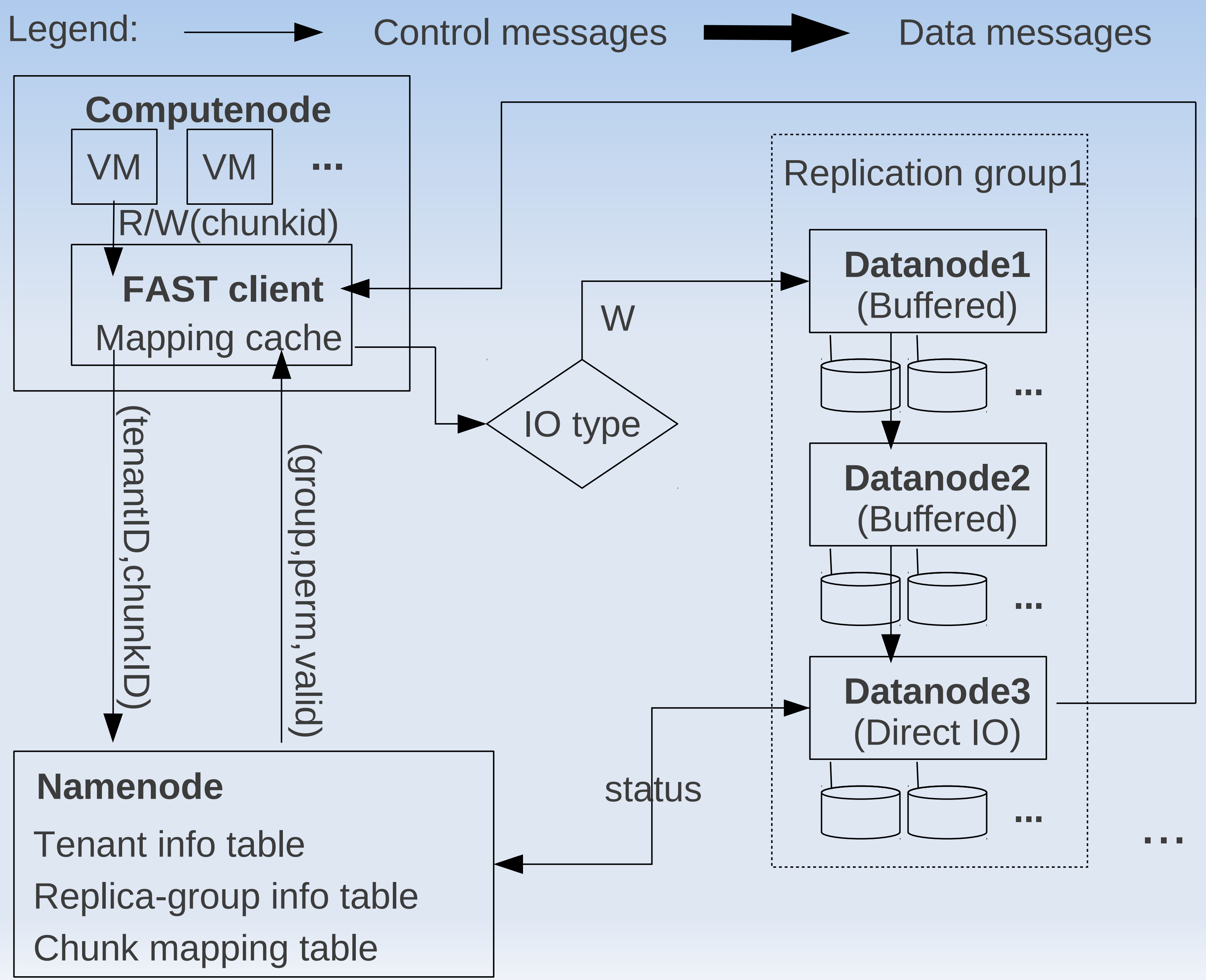
FAST – Architecture



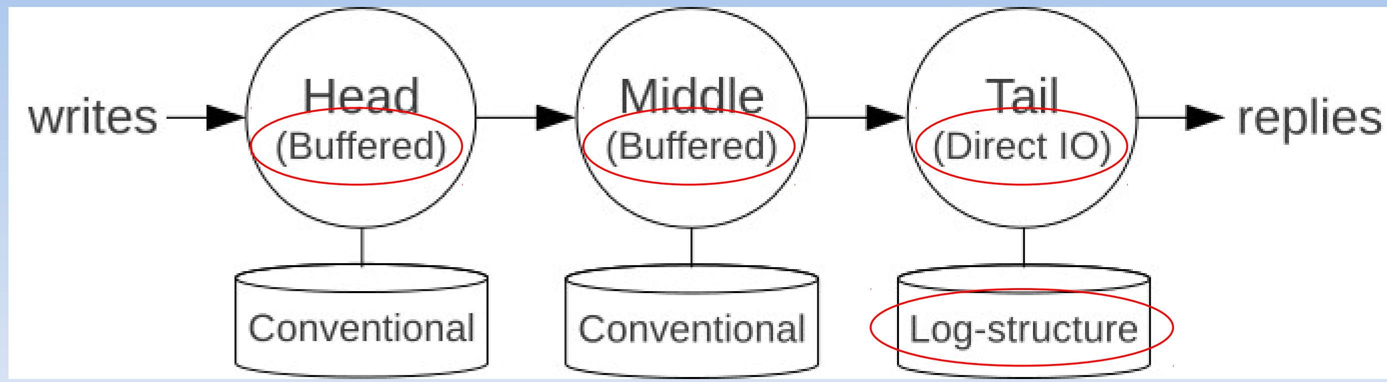
FAST – Architecture



FAST – Architecture



FAST – Disk Layout and Strategy



Chain Replication:
Disk Layout and Write Policies

- Default-with-steal strategy
  - By default, random reads go to head node and sequential reads go to middle node.
  - Allows idle or lightly-loaded replicas to steal ”requests” from other replicas

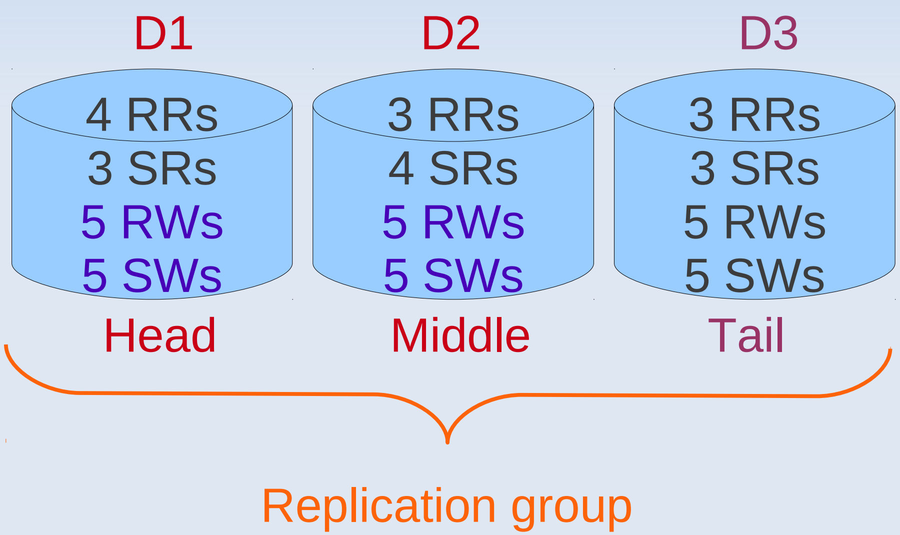
Initial Results – Simulation Setup

- Workloads:
  - One replication group
  - 30 tenants, each running one workload
  - 10 random read of 16 MB each
  - 10 sequential read of 19 MB each
  - 5 random write of 20 MB each
  - 5 sequential write of 20 MB each
- Workload assignment
  - Baseline: round-robin
  - FAST: workload type-aware

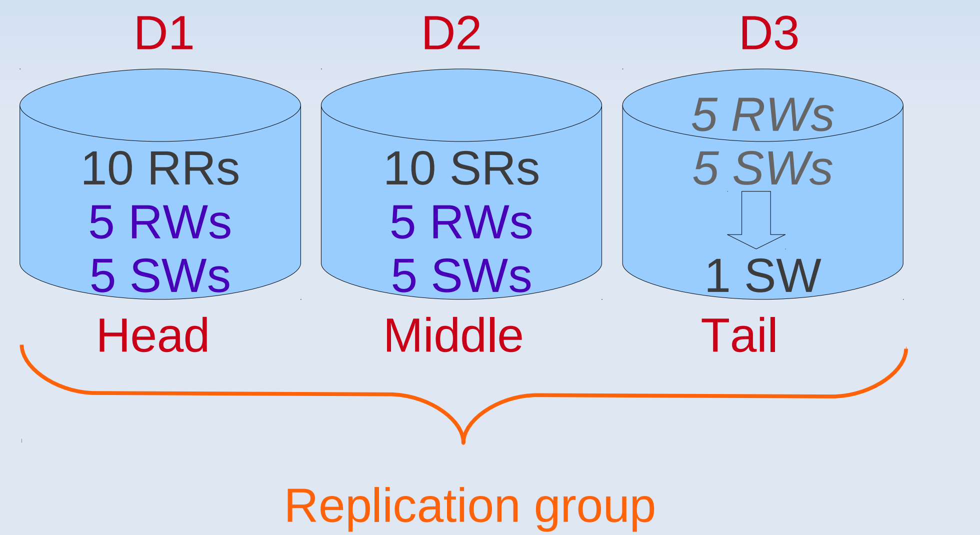
Initial Results - Assignment

Workloads: 10 RRs, 10 SRs, 5 RWs and 5 SWs

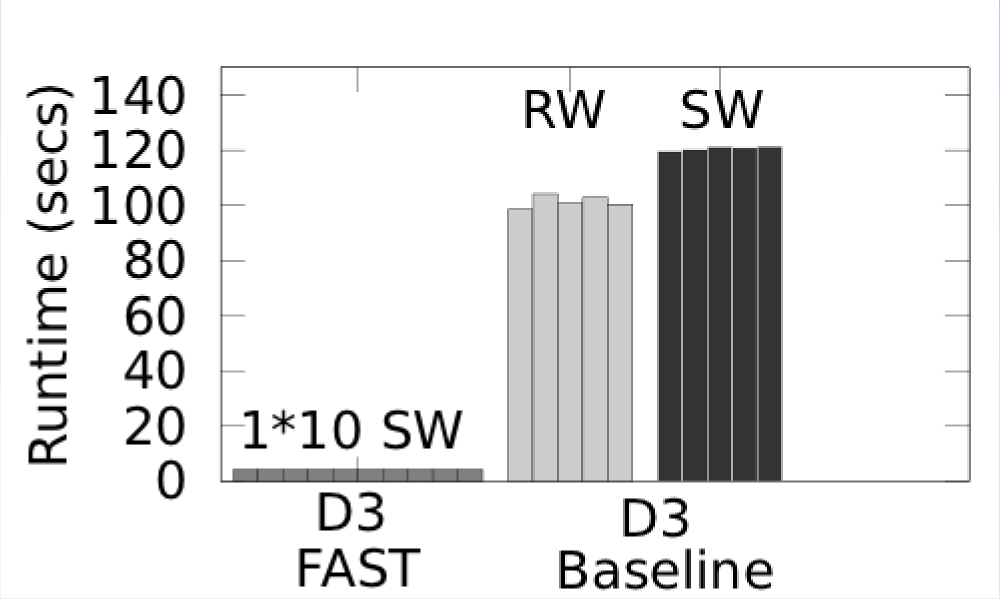
Baseline: (round-robin)



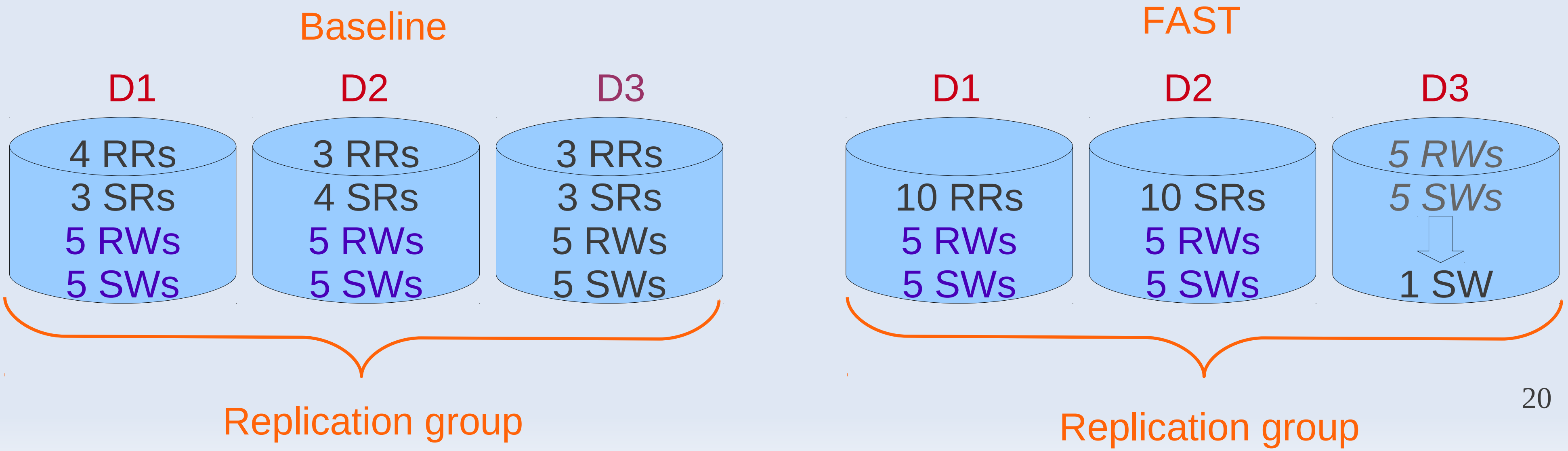
FAST



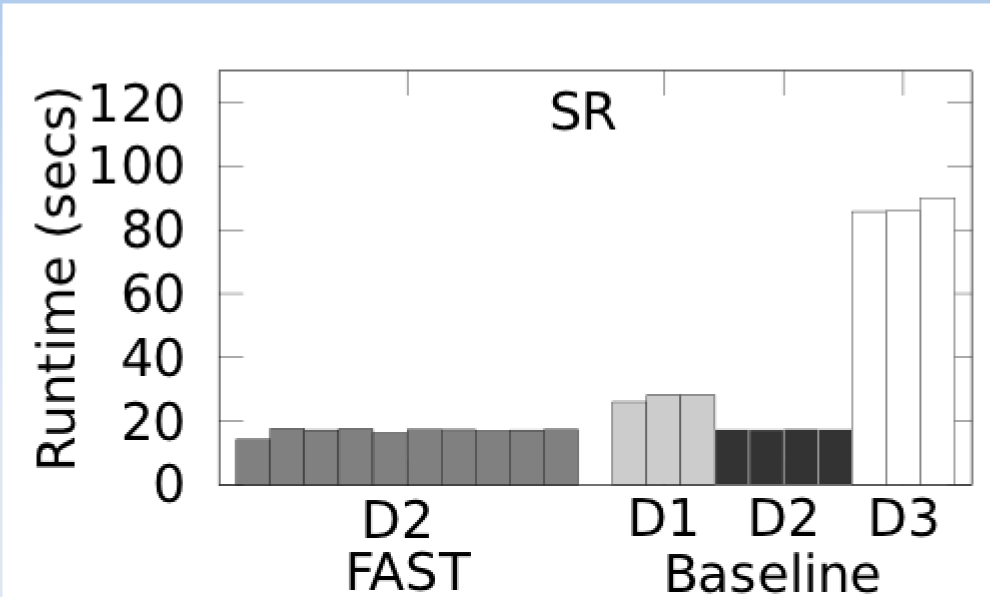
Initial Results - Evaluations



Result1:
Write workloads in FAST get much better performance



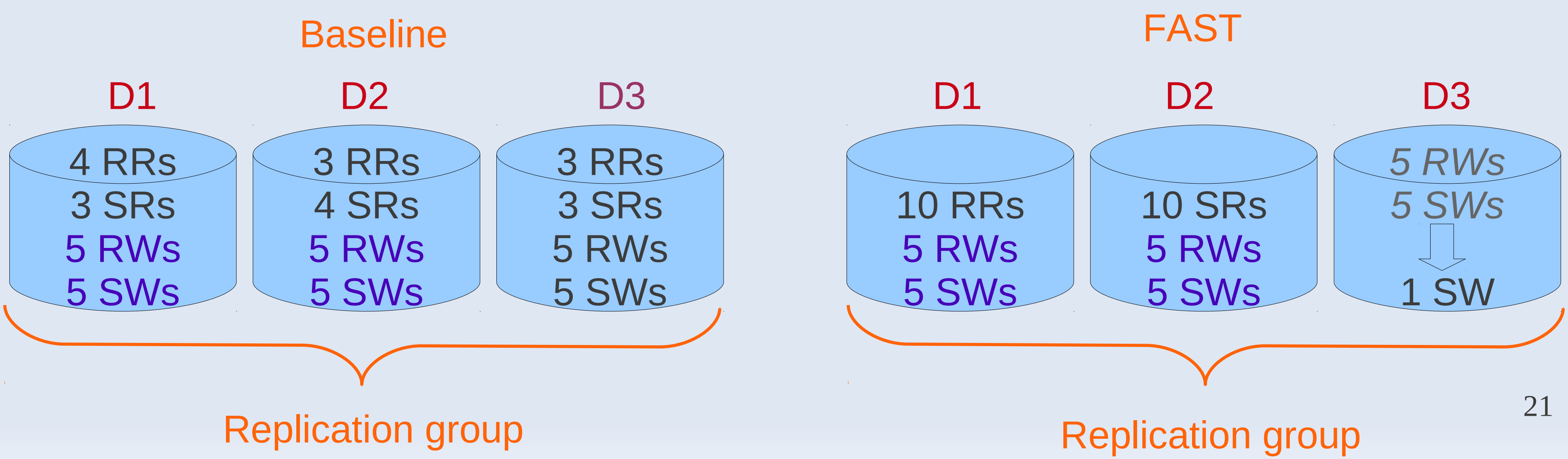
Initial Results - Evaluations



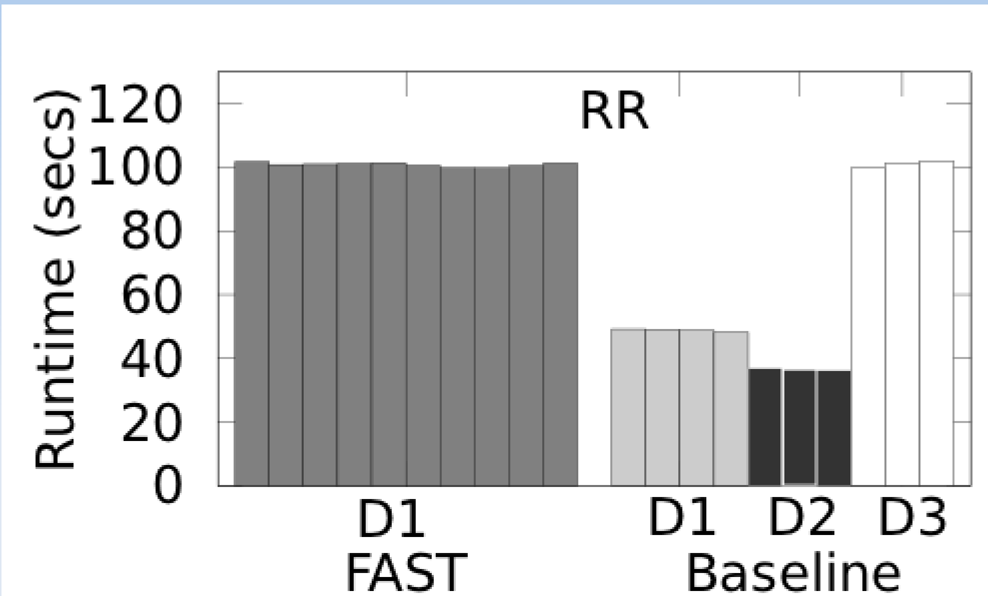
Result2:

a). All SRs in FAST get similar performance

b). SRs in FAST get comparable or better performance than the baseline



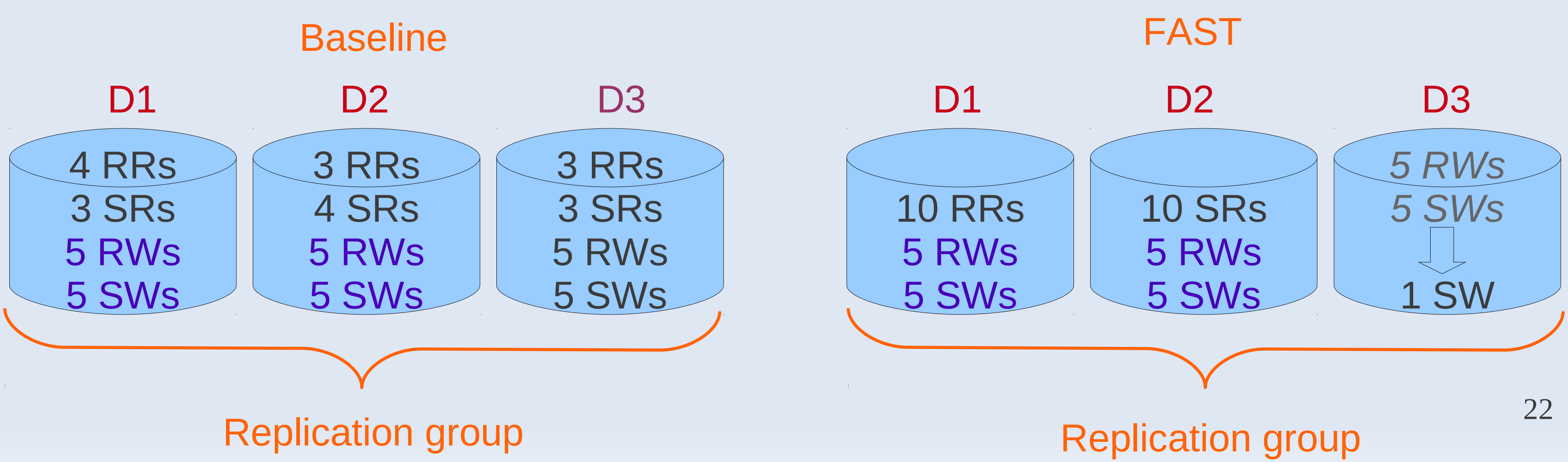
Initial Results - Evaluations



Result3:

a). All RRs in FAST get similar performance

b). RRs get worse performance in FAST



Future Work

- Modeling of effects of co-locating same type of workloads but with different I/O request characteristics
- Failure handling for datanode and namenode
- Load balancing among replication groups
- Tradeoff of chunk size
- System implementation

Conclusion

- Directs random and sequential reads to different replicas
- Introduce different write policies and disk layouts for chain replication

Thank you!

Questions?

Related Works and Contributions

- Related works
  - QoS-based resource allocation
    - Stonehege, Argon and Aqua
  - Support for latency control
    - SMART, BVT and pClock
  - Proporitional share + limit and reservation
    - mClock

These work typically abstract the storage device to a single block device and rely on the lower layer to deal with replications.

IOPS – 1

From disk specification:

- Average (rotational) latency: 3.0 ms
- Average read seek time: 4.7 ms
- Average write seek time: 5.3 ms

For the whole disk:

- Theoretical read IOPS = 1000/(3+4.7) = 129.87
- Theoretical write IOPS = 1000/(3+5.3) = 120.48
- Measured read IOPS = 123
- Measured write IOPS = 222

IOPS – 2

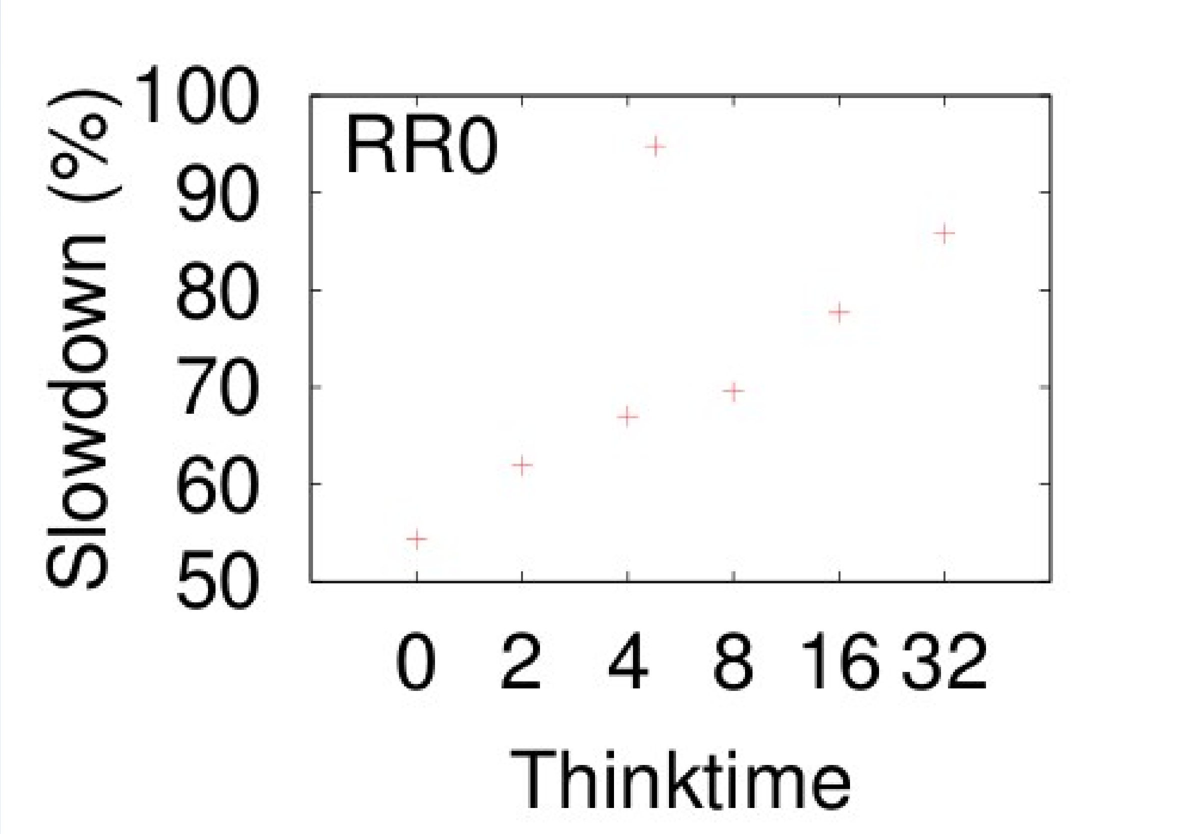
From disk specification:

- Average (rotational) latency: 3.0 ms
- Average read seek time: 4.7 ms
- Average write seek time: 5.3 ms

For a 10GB partition:

- Theoretical read IOPS = 1000/(3+4.7*10G/146.8G) = 301.19
- Theoretical write IOPS = 1000/(3+5.3*10G/146.8G) = 297.53
- Measured read IOPS = 198
- Measured write IOPS = 339

RR with different think times



SR with different block size

	Throughput
Isolation:	
4k-SR:	60.538 MB/s
256k-SR:	73.755 MB/s
concurrent:	
4k-SR:	31.222 MB/s
256k-SR:	35.651 MB/s

	Throughput
Isolation:	
4k-SR:	60.538 MB/s
1m-SR:	73.635 MB/s
concurrent:	
4k-SR:	28.037 MB/s
1m-SR:	38.942 MB/s