Shift into High Gear: Dramatically Improve Hadoop and NoSQL

M. C. Srivas, CTO/Co-founder

My Background

- Search
  - map-reduce, bigtable
- Chief Architect
  - now Netapp
- AFS
  - ran AFS team
  - now IBM

Google

SPINNAKER NETWORKS

NetApp™

Distributed File Systems

Andrew file system

VERITE · DEVOIR · VAILLANCE

MAPR™ TECHNOLOGIES

MapR History

06 — '07 — '09 — '11 — '12 — '13

06
Google publishes MapReduce paper

(struggling with Nutch) Copies MR paper, calls it Hadoop

Uses Hadoop

07
LinkedIn uses Hadoop

Uses Hadoop

'09
MapR founded

DOW drops from 14,300 to 6,800

'11
Launches ultra-reliable

'12
MapR partners with Google

amazon web services

Breaks all world records!

'13
MapR M7
Fastest NoSQL on the planet

2500 nodes Largest non-Web installation

MAPR TECHNOLOGIES

MapR Distribution for Apache Hadoop

Tools | Web 2.0 | Enterprise Apps | Vertical Apps

Analytics → Operations → OLTP

MAPR TECHNOLOGIES

Management

Hardening, Testing, Integrating, Innovating, Contributing as part of the Apache Open Source Community

NoSQL

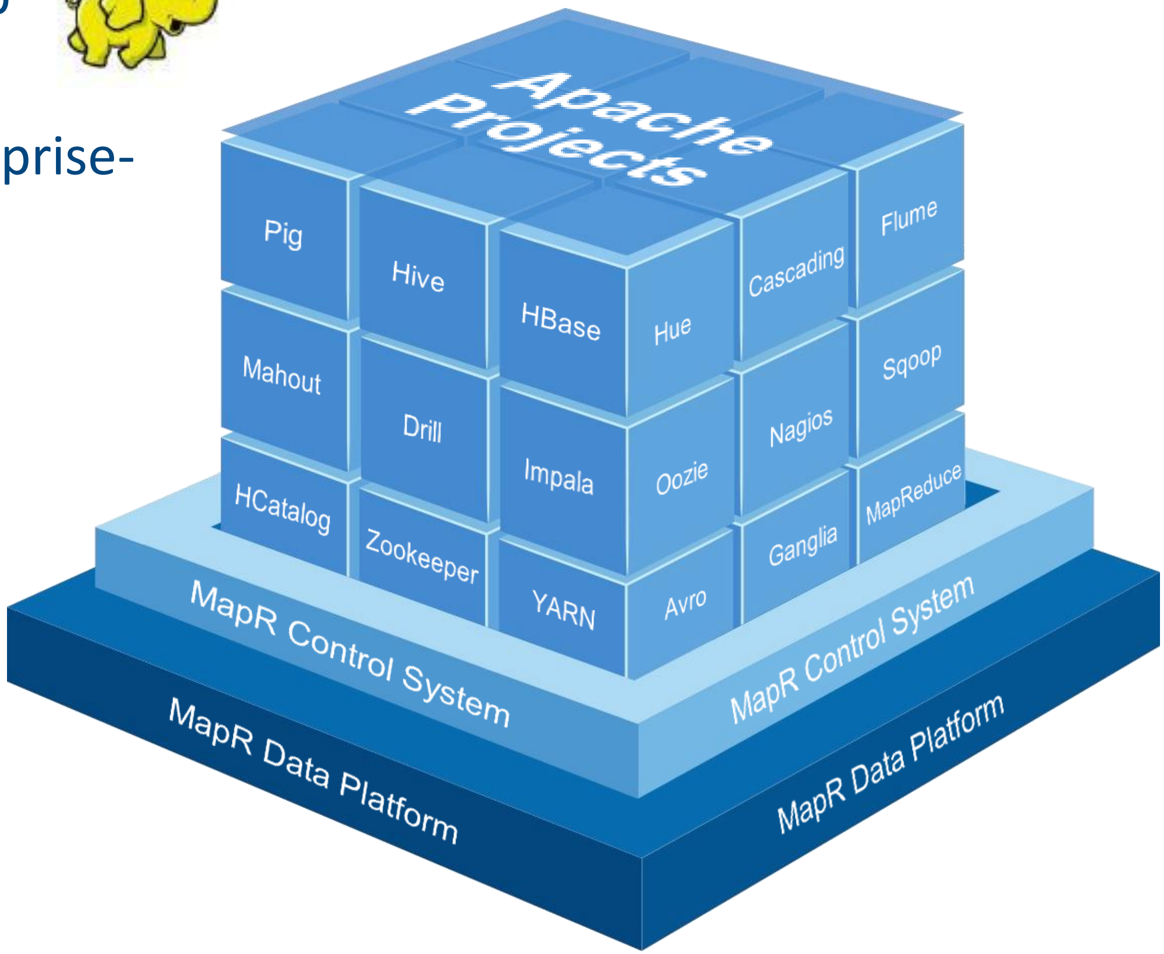
Data Platform

Batch → Interactive → Real Time

- 99.999% HA
- Data Protection
- Disaster Recovery
- Enterprise Integration
- Scalability & Performance
- Multi-Tenancy

MapR Distribution for Apache Hadoop

- 100% Apache Hadoop
- With significant enterprise-grade enhancements
- Comprehensive management
- Industry-standard interfaces
- Higher performance



MAPR TECHNOLOGIES

The Cloud Leaders Pick MapR

amazon webservices™

Amazon EMR is the largest Hadoop provider in revenue and # of clusters

Google

Google chose MapR to provide Hadoop on Google Compute Engine

openstack™

MapR partnership with Canonical and Mirantis provides advantages for OpenStack deployments

ubuntu

What Makes MapR so Reliable?

How to Make a Cluster Reliable

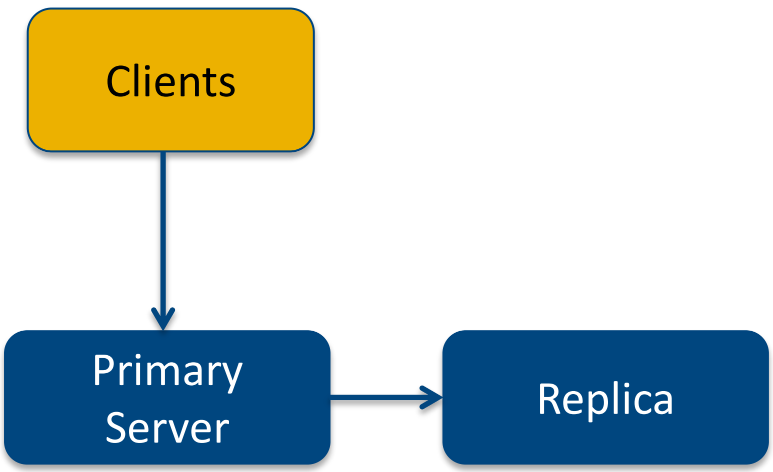
1. Make the storage reliable
   - Recover from disk and node failures
2. Make services reliable
   - Services need to checkpoint their state rapidly
   - Restart failed service, possibly on another node
   - Move check-pointed state to restarted service, using (1) above
3. Do it fast
   - Instant-on … (1) and (2) must happen very, very fast
   - Without maintenance windows
     - No compactions (e.g., Cassandra, Apache HBase)
     - No "anti-entropy" that periodically wipes out the cluster (e.g., Cassandra)

Reliability with Commodity Hardware

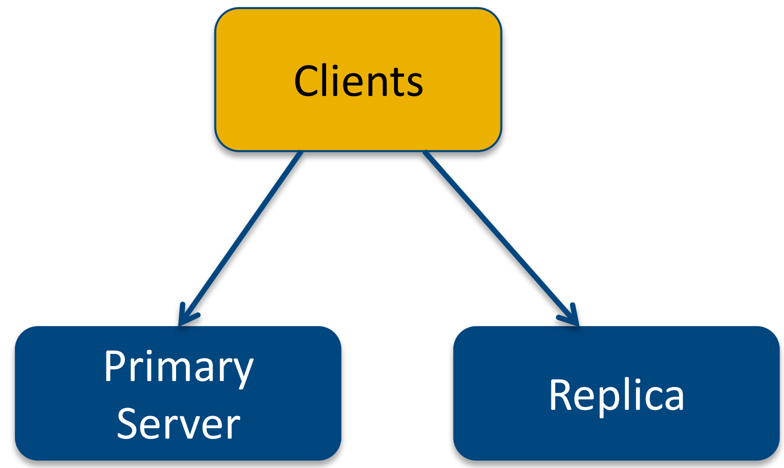
- No NVRAM
- Cannot assume special connectivity
  – No separate data paths for "online" vs. replica traffic
- Cannot even assume more than 1 drive per node
  – No RAID possible
- Use replication, but …
  – Cannot assume peers have equal drive sizes
  – Drive on first machine is 10x larger than drive on other?
- No choice but to replicate for reliability

Reliability via Replication

Replication is easy, right?
All we have to do is send the same bits to the master and replica



Normal replication, primary forwards

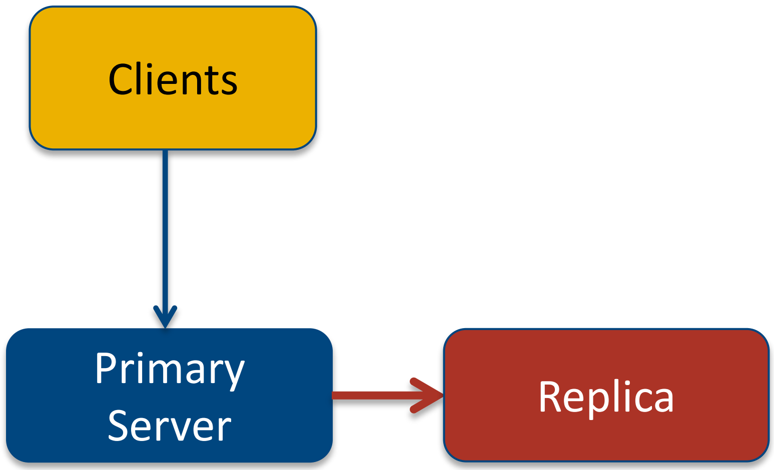


Cassandra-style replication

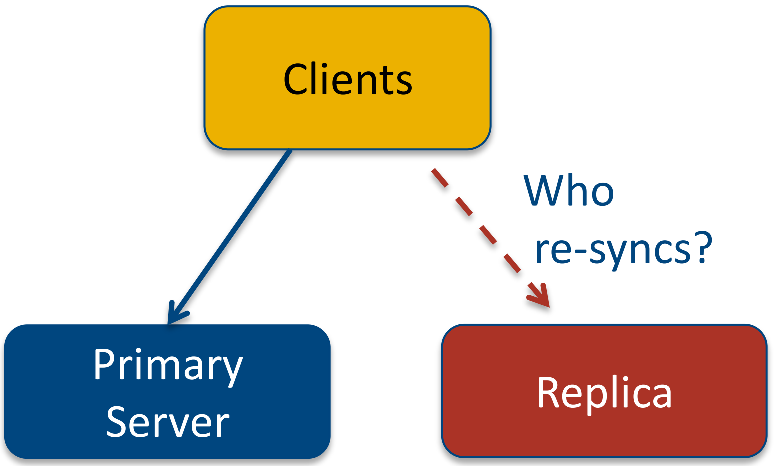
But Crashes Occur…

When the replica comes back, it is stale

- It must be brought up-to-date
- Until then, exposed to failure



Primary re-syncs replica



Replica remains stale until "anti-entropy" process kicked off by administrator

Unless it's HDFS …

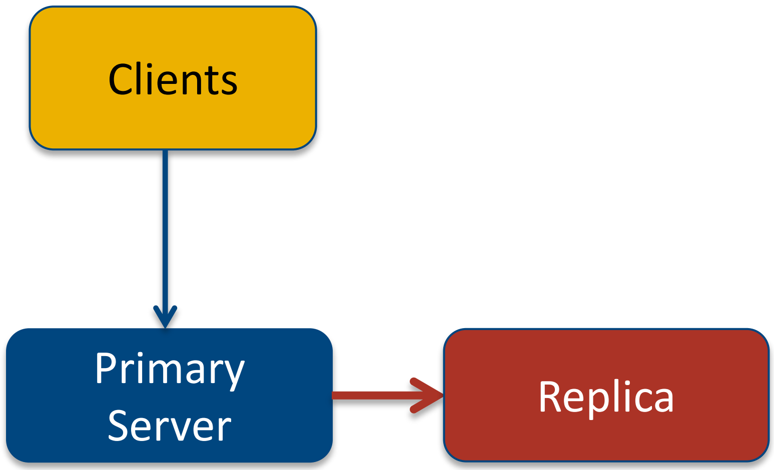
- HDFS solves the problem a third way
- **Makes everything read-only**
  - static data, trivial to re-sync
- Single writer, no reads allowed while writing
- File close is the transaction that allows readers to see data
  - Unclosed files are lost
  - Cannot write any further to closed file

HDFS design goal

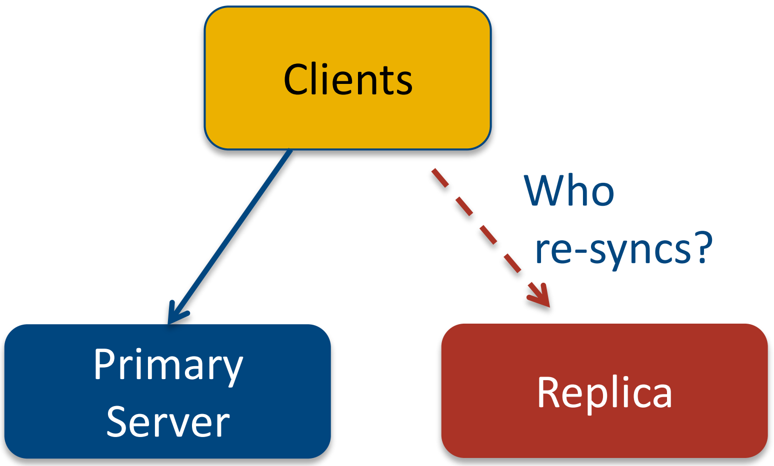
- Single writer, no reads allowed while writing
- File close is the transaction that allows readers to see data
  - Unclosed files are lost
  - Cannot write any further to closed file
- Realtime not possible with HDFS
  - To make data visible, must close file immediately after writing
  - Too many files is a serious problem with HDFS (a well documented limitation)
- HDFS therefore cannot do NFS, ever
  - No “close” in NFS … can lose data any time

To support normal apps, need RW

- Full read/write, “update-in-place” support
- Issue: re-sync the replica when it comes back



Primary re-syncs replica



Replica remains stale until "anti-entropy" process kicked off by administrator

How Long to Re-sync?

- 24 TB / server
  - @ 1000MB/s = 7 hours
  - Practical terms, @ 200MB/s = 35 hours

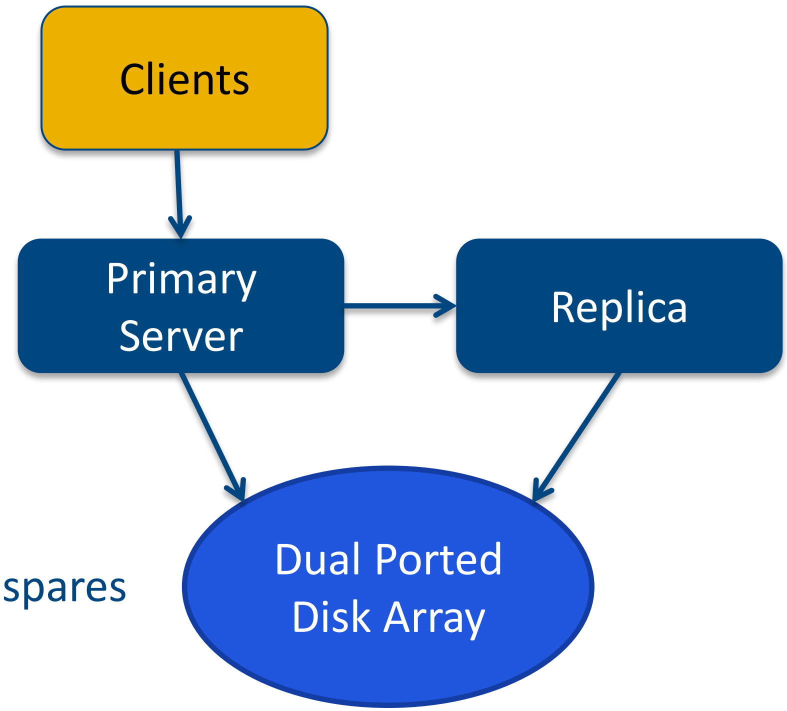
- Did you say you want to do this online?
  - Throttle re-sync rate to 1/10th
  - 350 hours to re-sync (= 15 days)

- What is your Mean Time To Data Loss (MTTDL)?
  - How long before a double disk failure?
  - A triple disk failure?

Traditional Solutions

Use dual-ported disk to side-step this problem

Servers use NVRAM



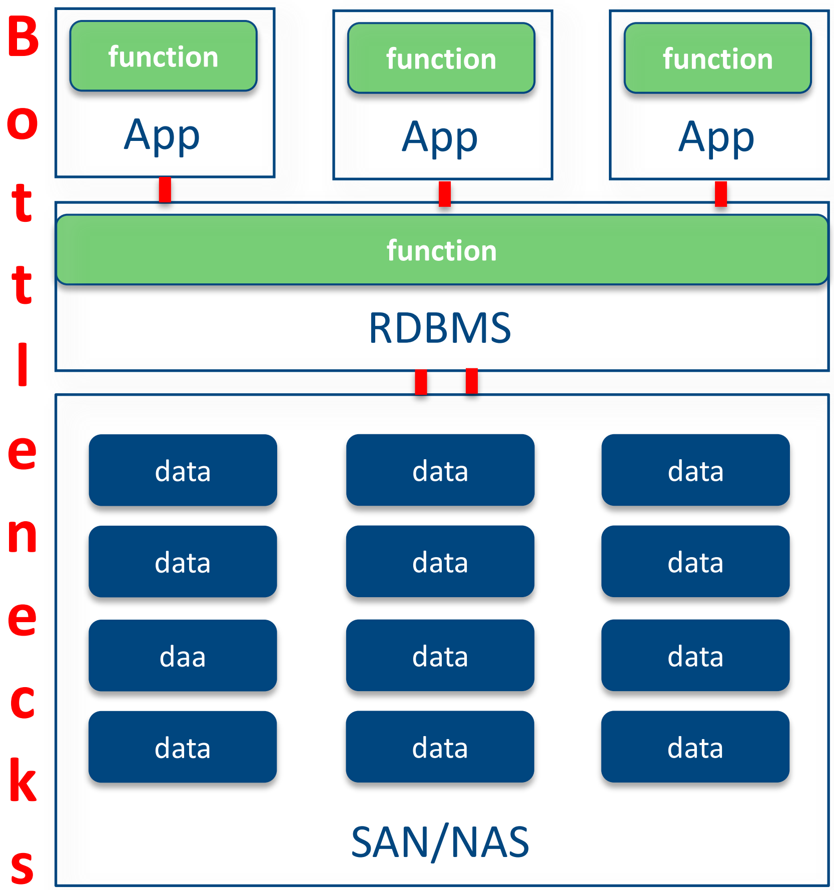
Raid-6 with idle spares

~~COMMODITY HARDWARE~~ ~~LARGE SCALE CLUSTERING~~

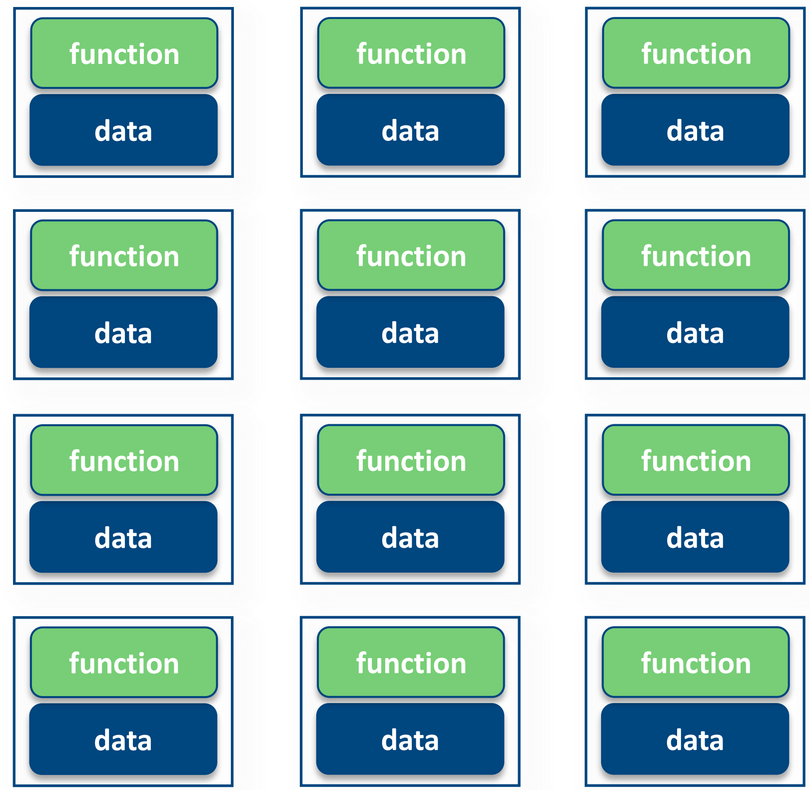
Large Purchase Contracts, 5-year spare-parts plan

Forget Performance Too?

Traditional Architecture

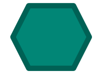


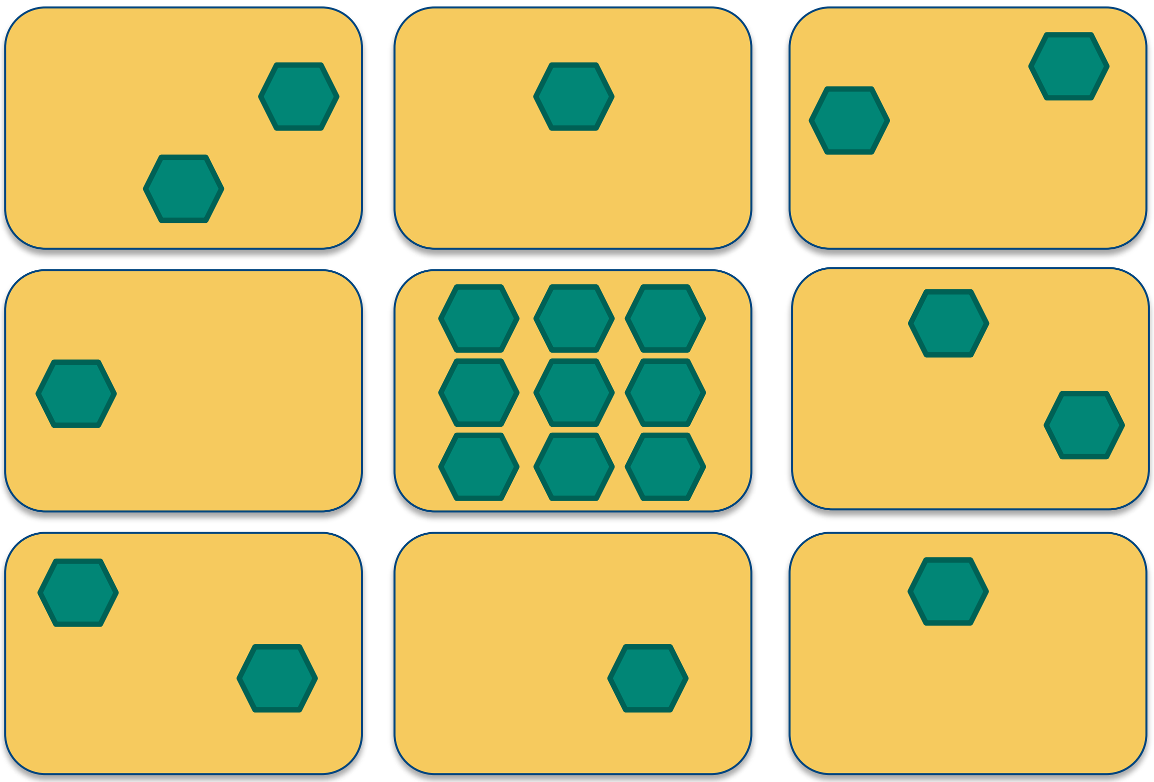
Hadoop



Geographically dispersed also?

What MapR Does

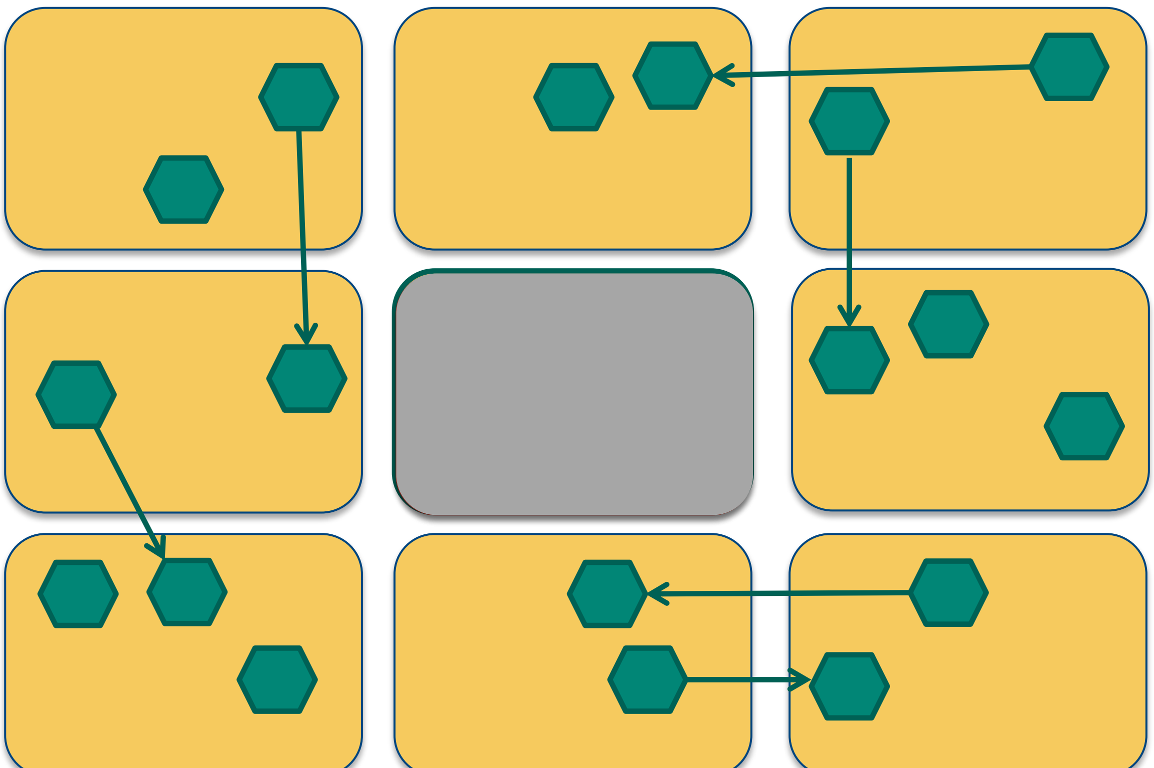
- Chop the data on each node to few 1000s of pieces
  - Pieces are called ***containers*** 
- Spread replicas of each container across the cluster



Why Does It Improve Things?

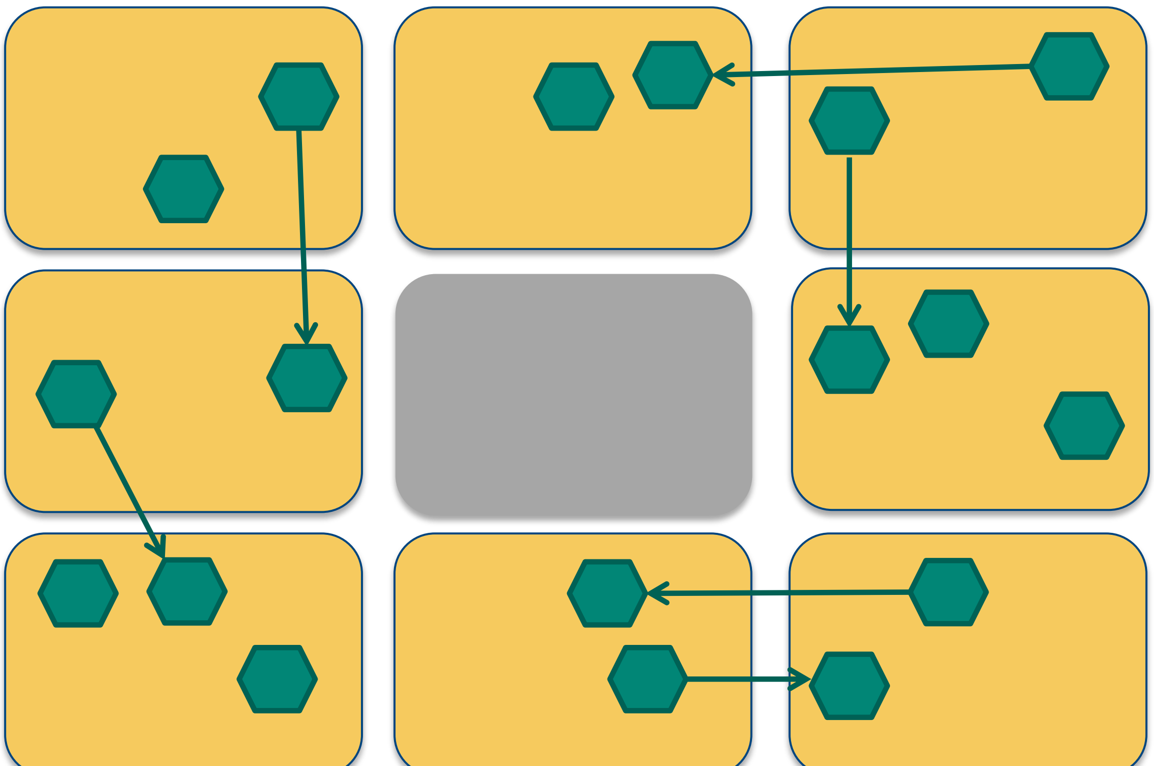
MapR Replication Example

- 100-node cluster
- Each node holds 1/100th of every node's data
- When a server dies, entire cluster re-syncs the dead node's data



MapR Re-sync Speed

- 99 nodes re-sync'ing in parallel
  - 99x number of drives
  - 99x number of Ethernet ports
- Each is re-sync'ing 1/100th of the data
- Net speed up is about 100x
  - 3.5 hours vs. 350
- **MTTDL is 100x better**



MAPR TECHNOLOGIES

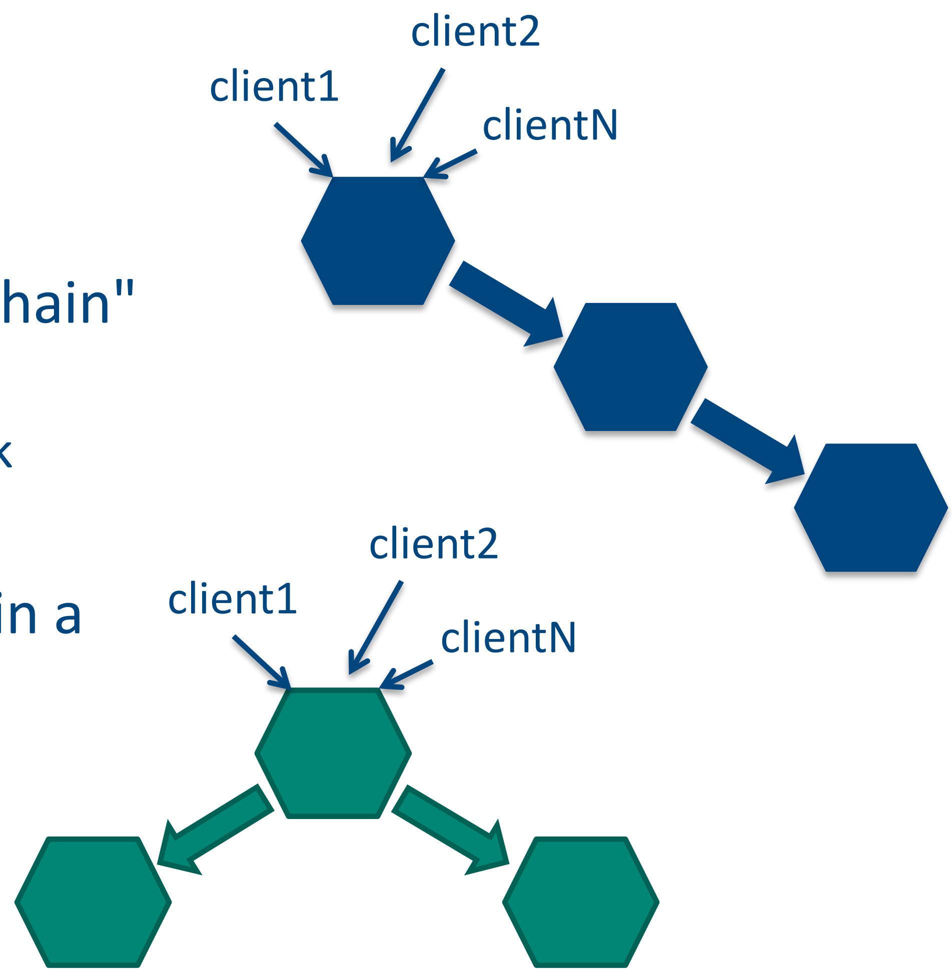
MapR Reliability

- Mean Time To Data Loss (MTTDL) is far better
  – Improves as cluster size increases
- Does not require idle spare drives
  – Rest of cluster has sufficient spare capacity to absorb one node's data
  – On a 100-node cluster, 1 node's data == 1% of cluster capacity
- Utilizes all resources
  – no wasted "master-slave" nodes
  – no wasted idle spare drives … all spindles put to use
- Better reliability with less resources
  – on commodity hardware!

Why Is This So Difficult?

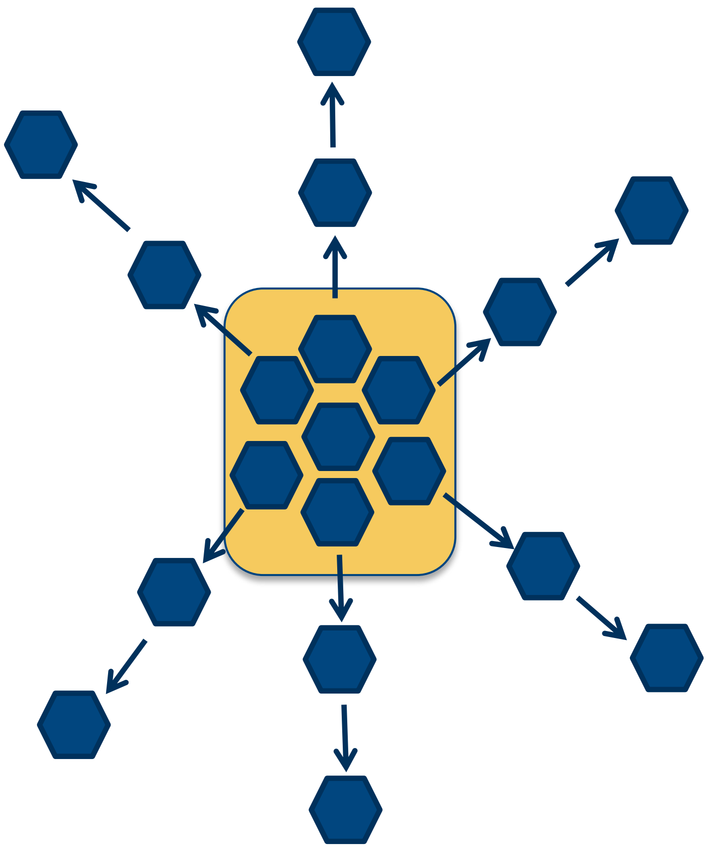
MapR's Read-write Replication

- Writes are synchronous
  - Visible immediately
- Data is replicated in a "chain" fashion
  - Utilizes full-duplex network
- Meta-data is replicated in a "star" manner
  - Response time better



Container Balancing

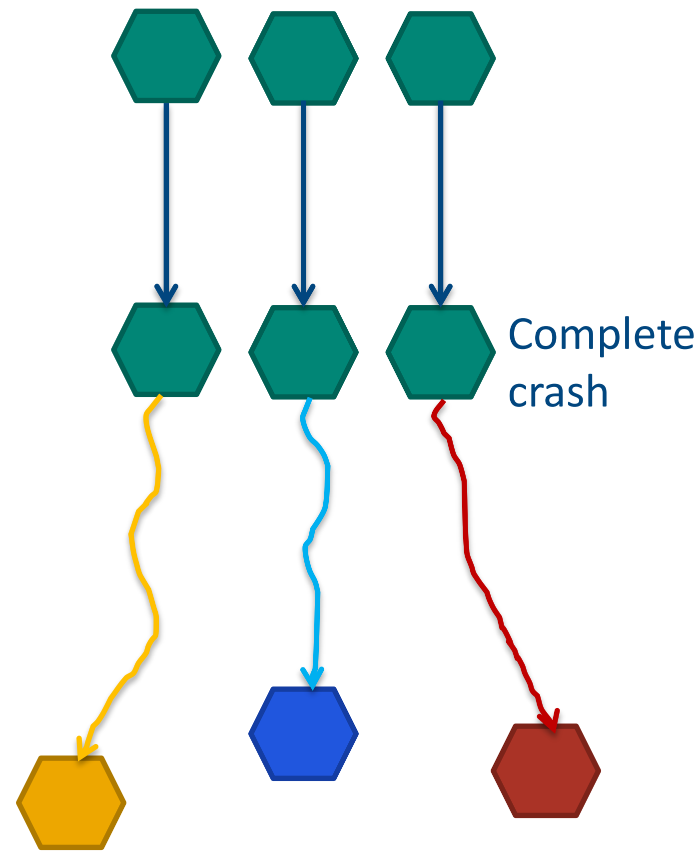
- Servers keep a bunch of containers "ready to go”
- Writes get distributed around the cluster



- As data size increases, writes spread more (like dropping a pebble in a pond)
- Larger pebbles spread the ripples farther
- Space balanced by moving idle containers

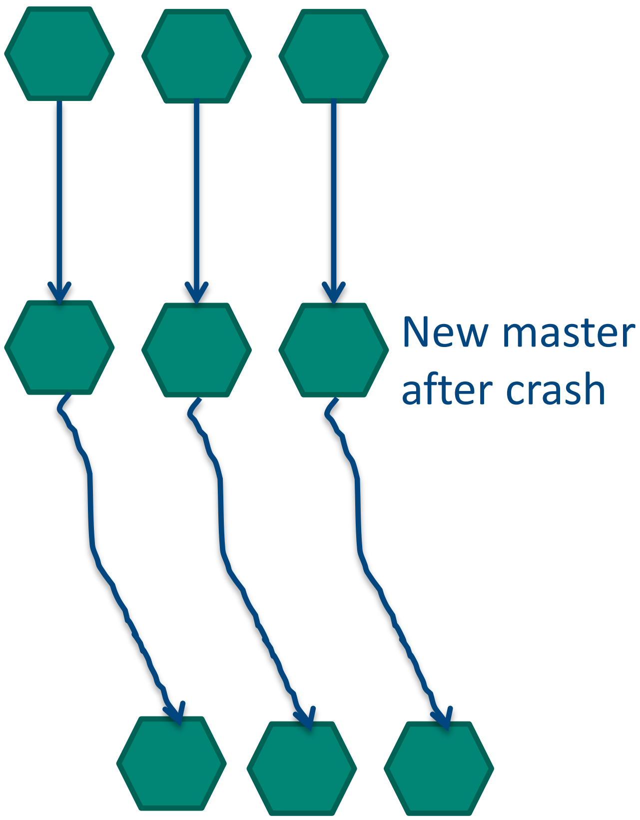
MapR Container Re-sync

- MapR is 100% random write
  – uses distributed transactions
- On a complete crash, all replicas diverge from each other
- On recovery, which one should be master?



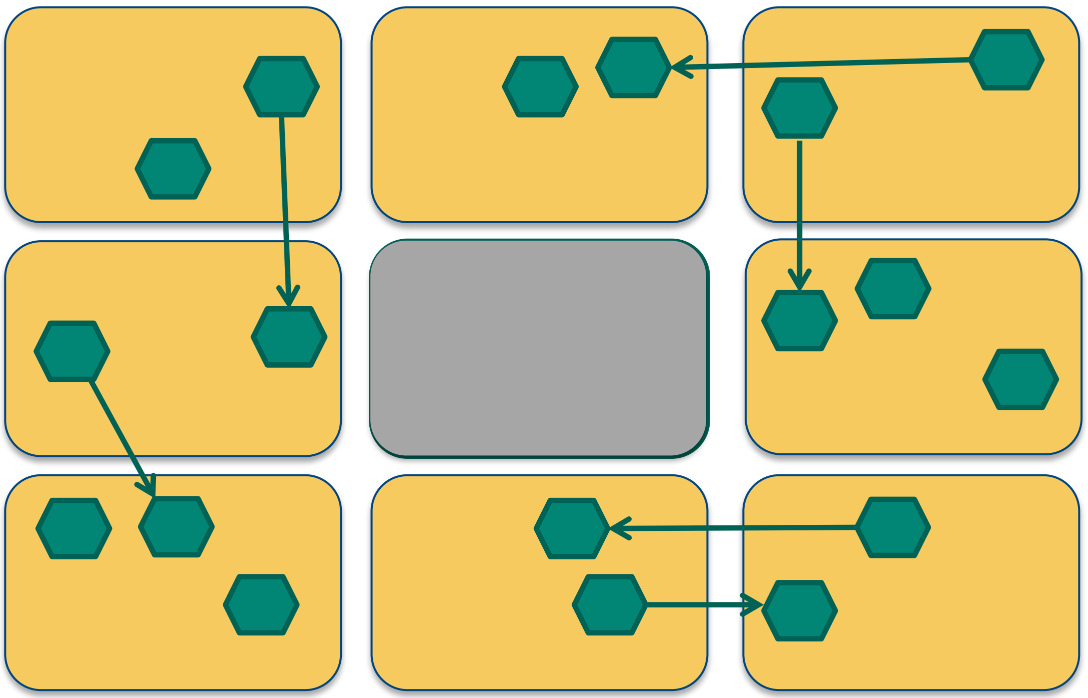
MapR Container Re-sync

- MapR can detect exactly where replicas diverged
  - even at 2000 MB/s update rate
- Re-sync means
  - roll-back each to divergence point
  - roll-forward to converge with chosen master
- Done while online
  - with very little impact on normal operations



MapR Does Automatic Re-sync Throttling

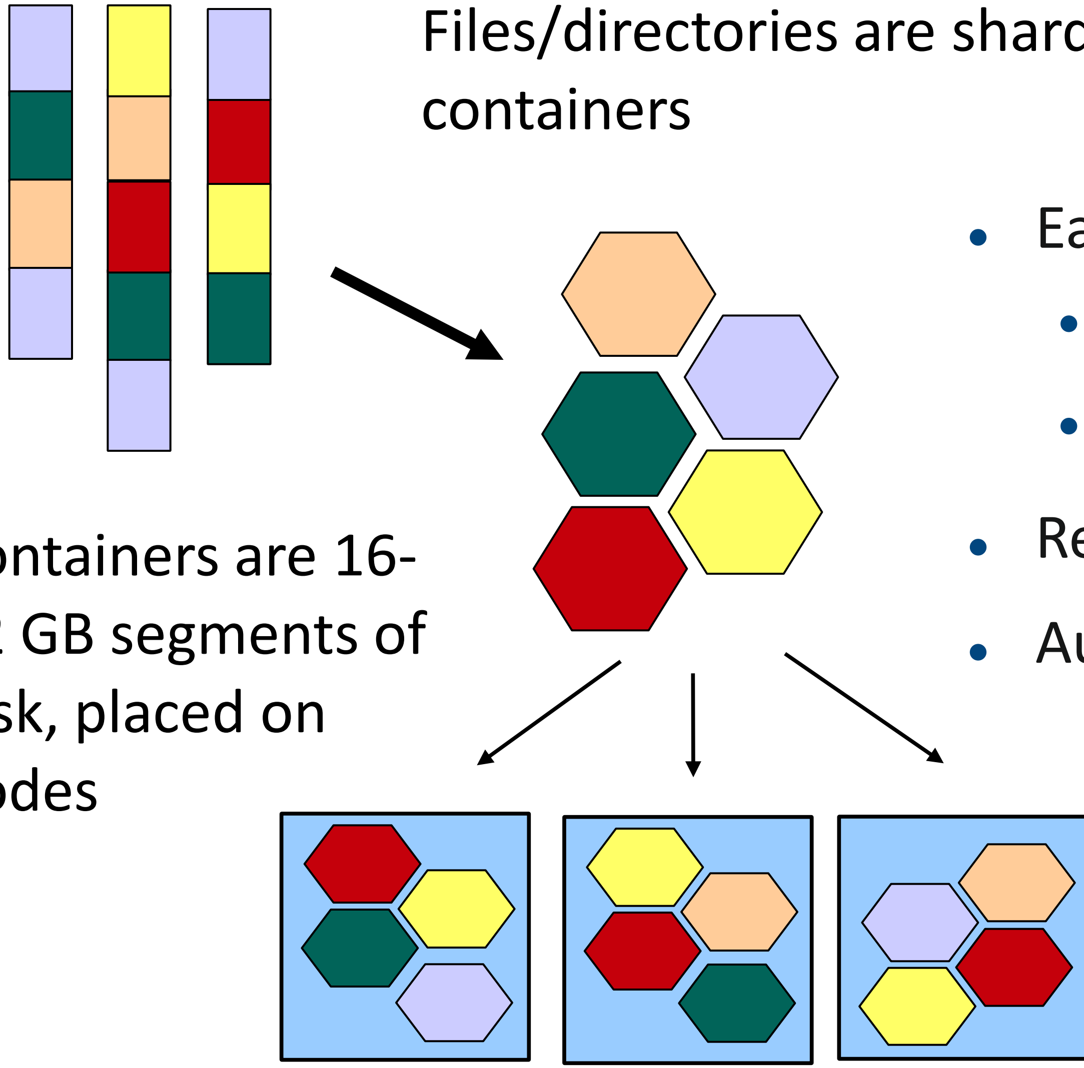
- Re-sync traffic is "secondary"
- Each node continuously measures RTT to all its peers
- More throttle to slower peers
  – Idle system runs at full speed
- All automatically



But Where Do Containers Fit In?

MapR's Containers

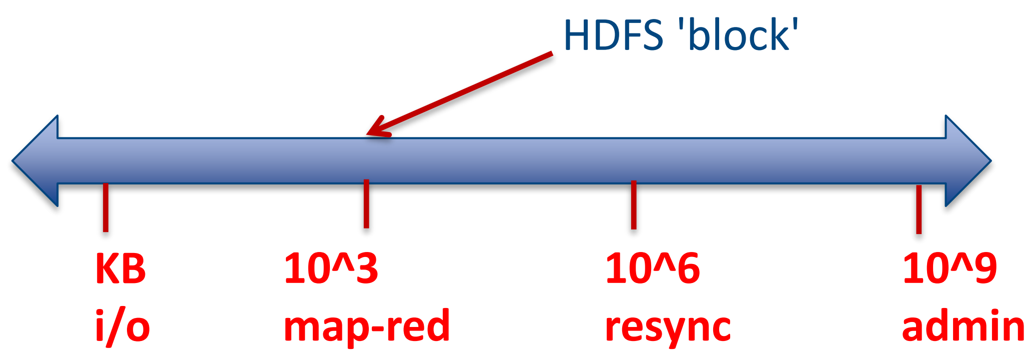
Files/directories are sharded and placed into containers



- Each container contains
  - Directories & files
  - Data blocks
- Replicated on servers
- Automatically managed

Containers are 16-32 GB segments of disk, placed on nodes

MapR's Architectural Params

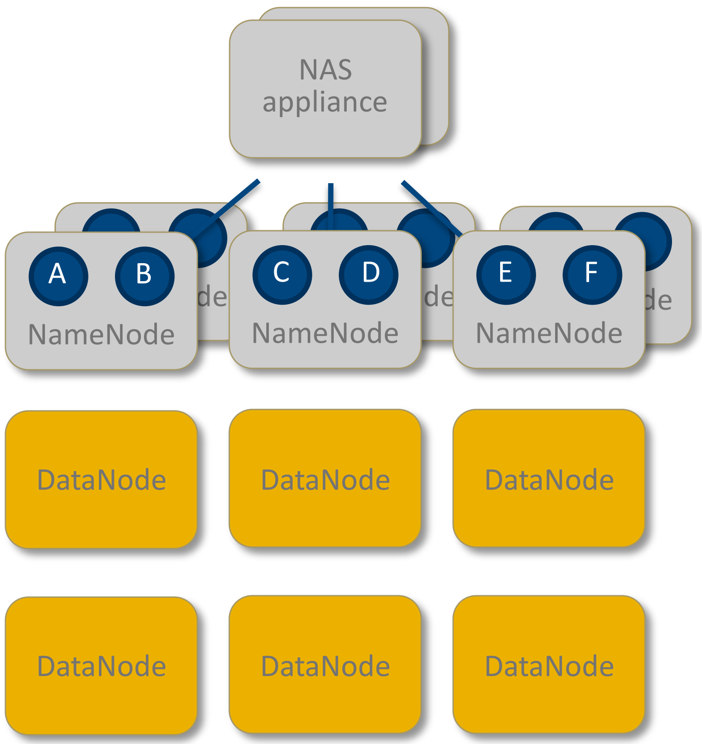


- Unit of I/O
  - 4K/8K (8K in MapR)
- Unit of Chunking (a map-reduce *split*)
  - 10-100's of megabytes
- Unit of Resync (a replica)
  - 10-100's of gigabytes
  - container in MapR
  - automatically managed
- Unit of Administration (snap, repl, mirror, quota, backup)
  - 1 gigabyte - 1000's of terabytes
  - **volume in MapR**
  - what data is affected by my missing blocks?

Where & How Does MapR Exploit This Unique Advantage?

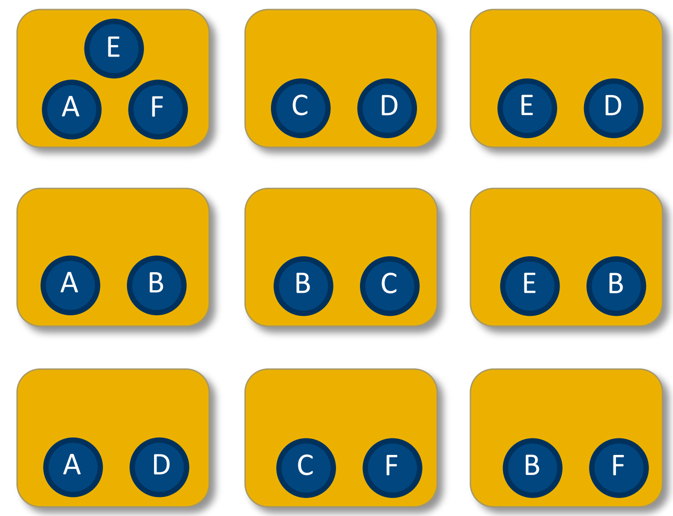
MapR's No NameNode HA™ Architecture

HDFS Federation



- Multiple single points of failure
- Limited to 50-200 million files
- Performance bottleneck
- Commercial NAS required

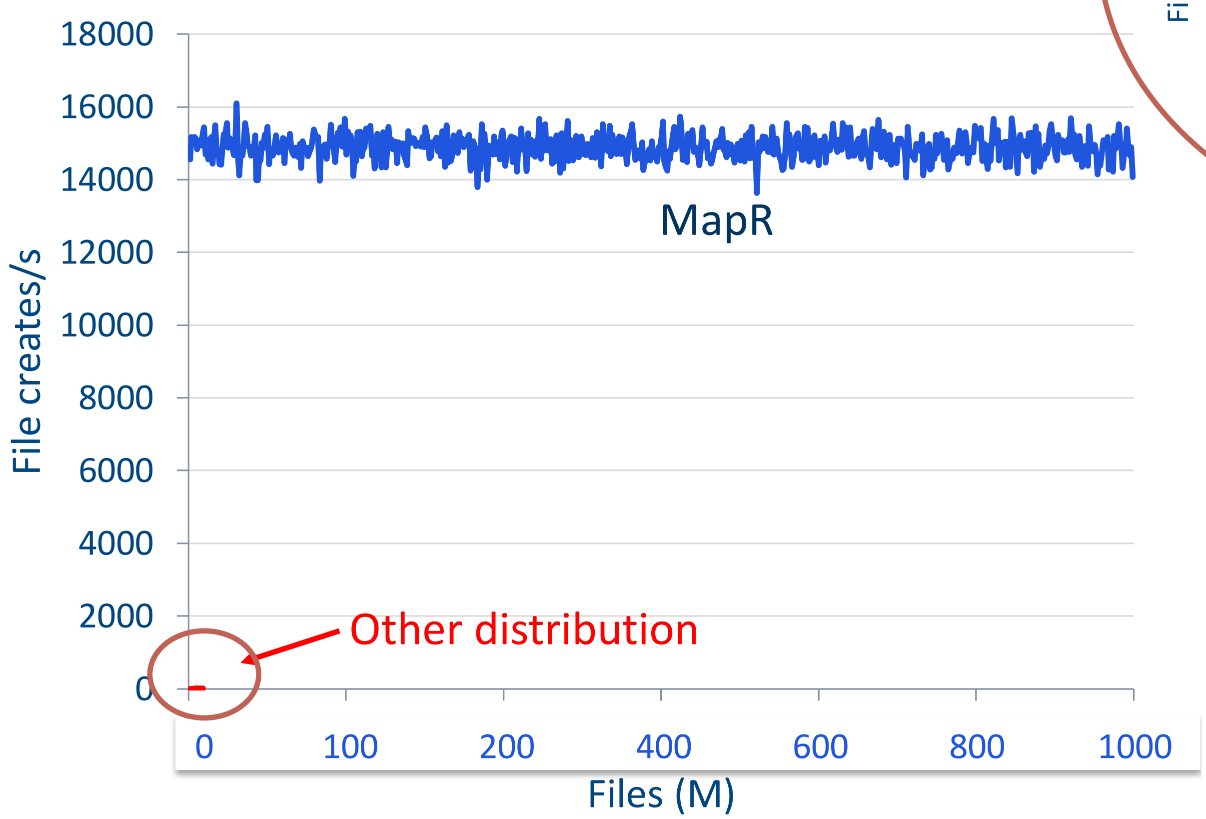
MapR (Distributed Metadata)

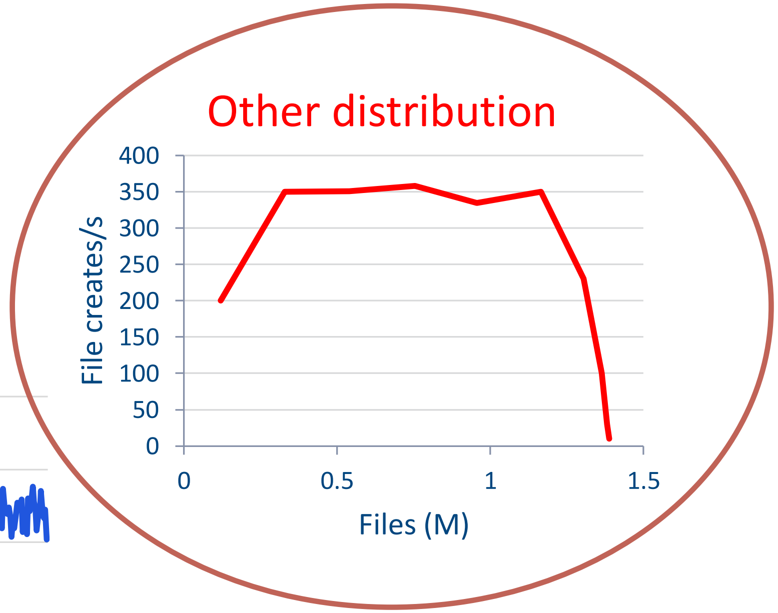


- HA w/automatic failover
- Instant cluster restart
- Up to 1T files (> 5000x advantage)
- 10-20x higher performance
- 100% commodity hardware

Relative Performance and Scale

	MapR	Other	Advantage
Rate (creates/s)	14-16K	335-360	40x
Scale (files)	6B	1.3M	4615x





Benchmark: 100-byte files

Hardware: 10 nodes

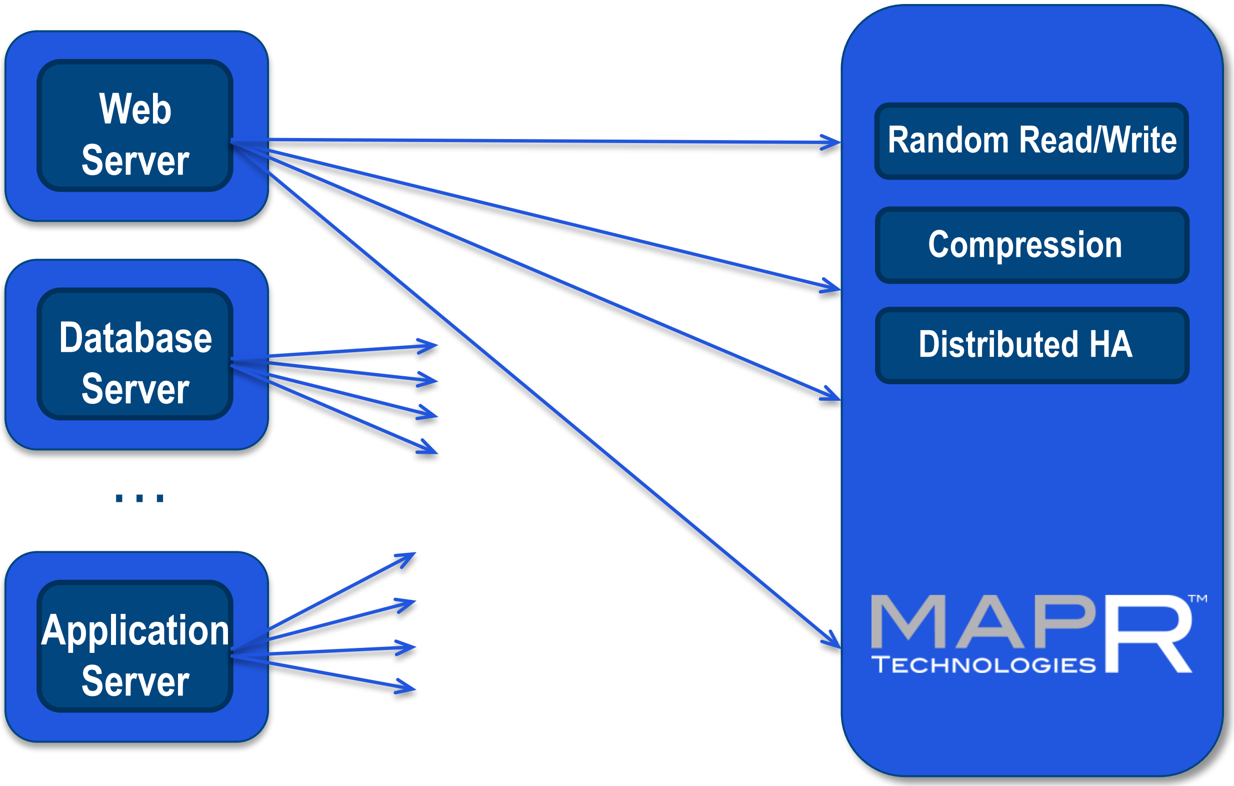
- 2 x 4 cores
- 24 GB RAM
- 12 x 1 TB 7200 RPM
- 1 GigE NIC

Where & How Does MapR Exploit This Unique Advantage?

MapR's NFS Allows Direct Deposit

Connectors not needed

No extra scripts or clusters to deploy and maintain



Where & How Does MapR Exploit This Unique Advantage?

MapR Volumes & Snapshots

100K volumes are OK, create as many as desired!

Volumes dramatically simplify data management

- Replication control
- Mirroring
- Snapshots
- Data placement control
- Ultra-strong security
  - Certificates (ie, like https)
  - Kerberos v5

Where & How Does MapR Exploit This Unique Advantage?

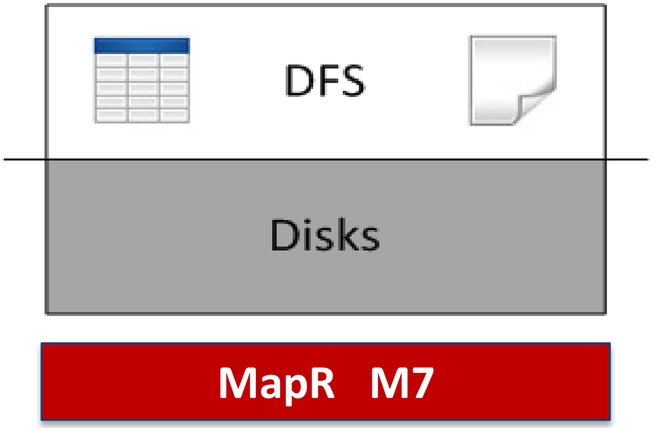
MapR M7 Tables

- Binary compatible with Apache HBase
  - no recompilation needed to access M7 tables
  - just set CLASSPATH

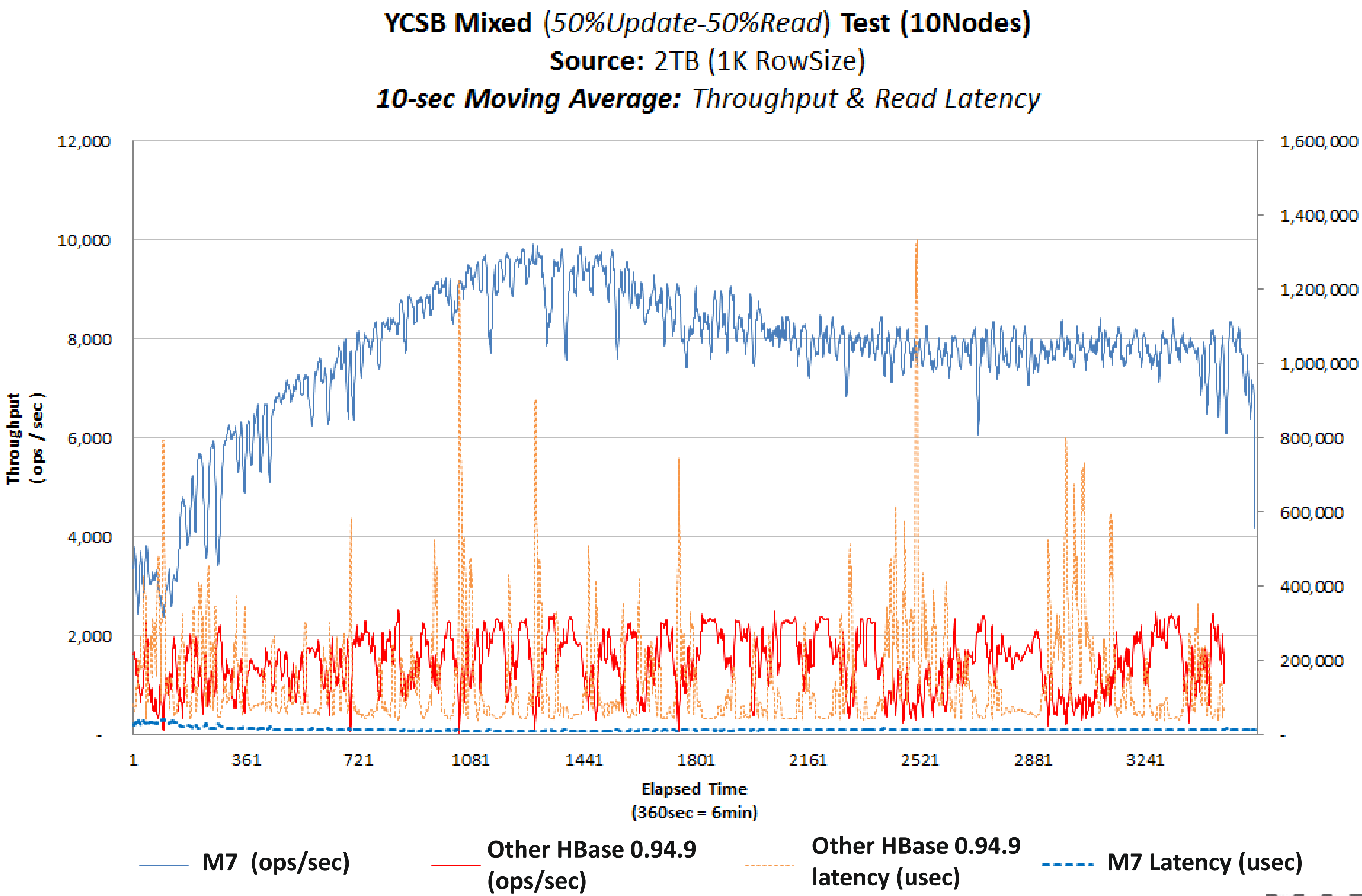
- M7 tables accessed via pathname
  - openTable("hello") … uses HBase
  - openTable("/hello") … uses M7
  - openTable("/user/srivas/hello") … uses M7

M7 Tables

- M7 tables integrated into storage
  – always available on every node
- Unlimited number of tables
- **No compactions**
  – update in place
- **Instant-on**
  – Zero recovery time
- **5-10x better performance**
- Consistent low latency
  – At 95th & 99th percentiles

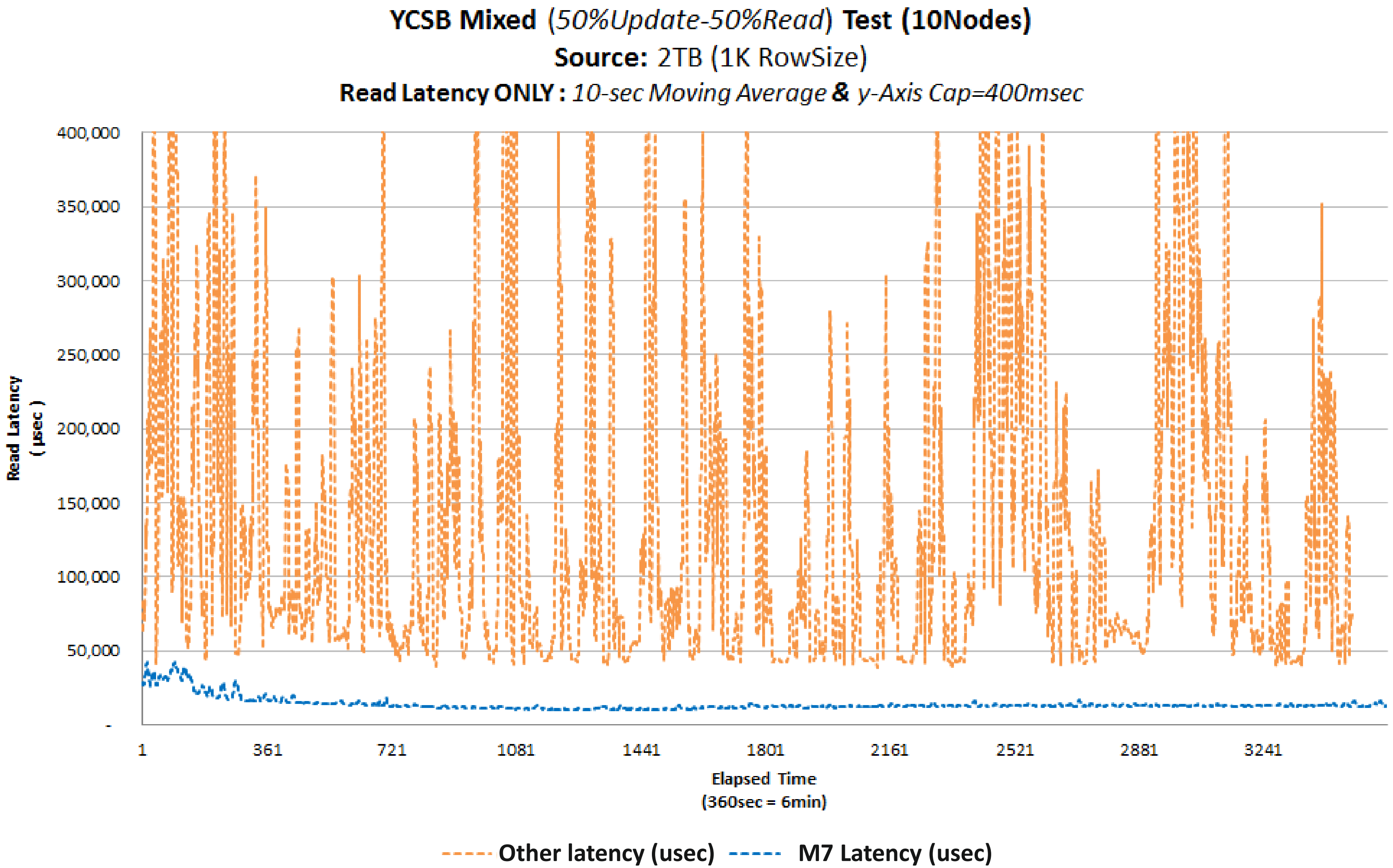


YCSB 50-50 Mix (throughput)



MAPR TECHNOLOGIES

YCSB 50-50 mix (latency)



MAPR TECHNOLOGIES

Where & How Does MapR Exploit This Unique Advantage?

MapR Makes Hadoop Truly HA

- **ALL** Hadoop components have High Availability
  – e.g. YARN
- ApplicationMaster (old JT) and TaskTracker record their state in MapR
- On node-failure, AM recovers its state from MapR
  – Works even if entire cluster restarted
- All jobs resume **from where they were**
  – Only from MapR
- Allows preemption
  – MapR can preempt any job, without losing its progress
  – ExpressLane™ feature in MapR exploits it

MapR Does MapReduce (Fast!)

TeraSort Record

1 TB in 54 seconds

1003 nodes

MinuteSort Record

1.5 TB in 59 seconds

2103 nodes

MapR Does MapReduce (Faster!)

TeraSort Record

1 TB in 54 seconds

1003 nodes

MinuteSort Record

1.65 ~~1.5~~ TB in 59 seconds

~~2103~~ nodes

300

Where & How Can YOUR CODE Exploit This Unique Advantage?

Exploit MapR's Unique Advantages

- **ALL** your code can easily be scale-out HA
  - Save service-state in MapR
  - Save data in MapR
- Use Zookeeper to notice service failure
- Restart anywhere, data+state will move there automatically
- *That's what we did!*

- **Only from MapR**: HA for Impala, Hive, Oozie, Storm, MySQL, SOLR/Lucene, HCatalog, Kafka, …

MapR: Unlimited Hadoop

Reliable Compute

Dependable Storage

Build cluster brick by brick, one node at a time

- Use commodity hardware at rock-bottom prices
- Get enterprise-class reliability: instant-restart, snapshots, mirrors, no single-point-of-failure, …
- Export via NFS, ODBC, Hadoop and other standard protocols

Thank you!

srivas@maprtech.com

www.mapr.com