



The Reliability Challenge with Commodity Hardware

M. C. Srivas
CTO & Founder



My Background

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

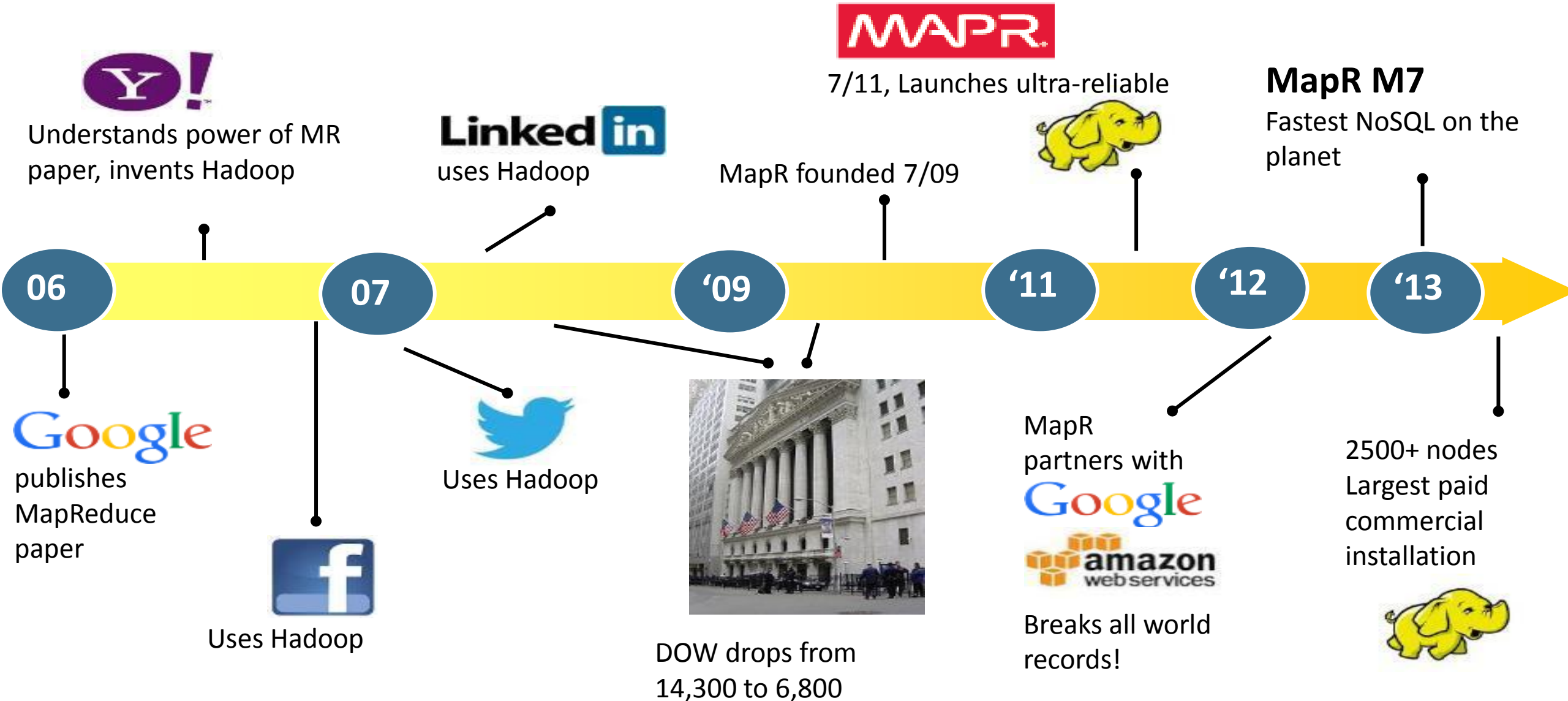


Distributed File Systems

Andrew file system



MapR History



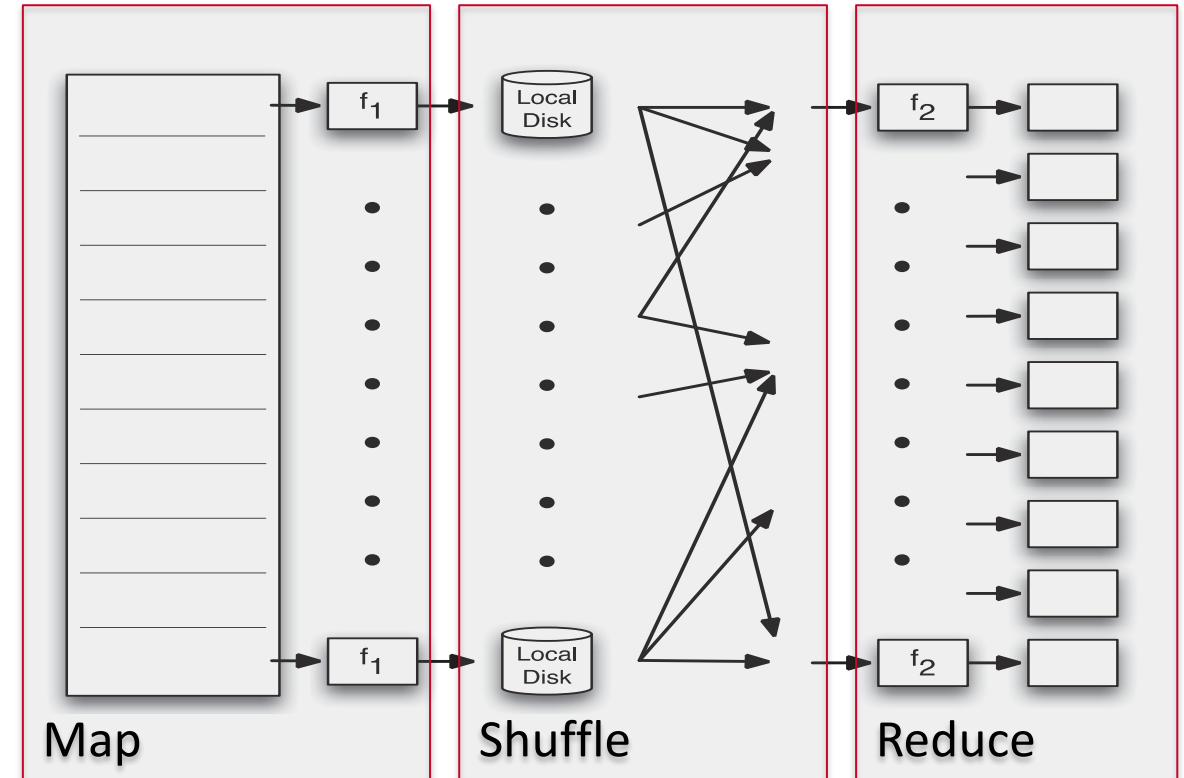
**Say BIG DATA
one more time ...**



NPR

What is Hadoop?

- Open Source Apache Project
- Hadoop Core includes:
 - Distributed File System
 - distributes data
 - Map/Reduce
 - distributes computation (near the data)
- Runs on commodity hardware

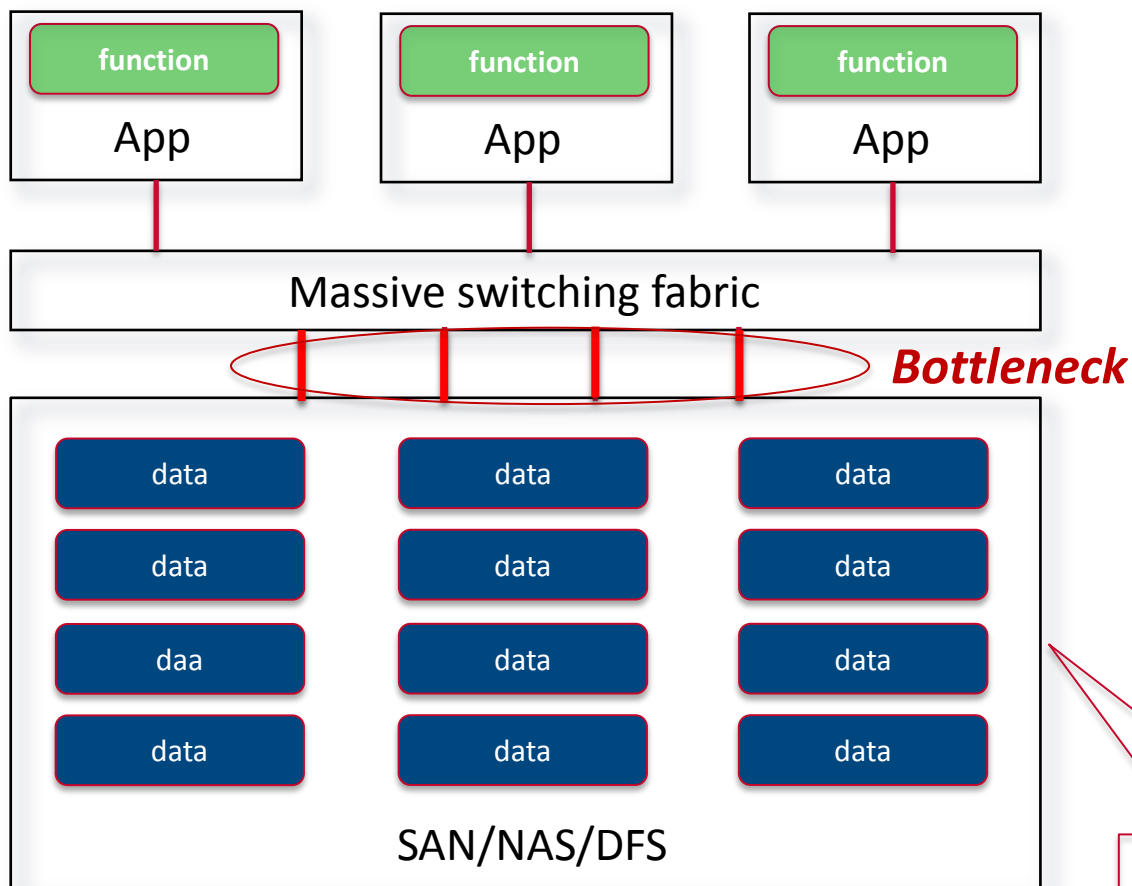


Why does Hadoop win?



Drowning in Data!

Traditional Architecture



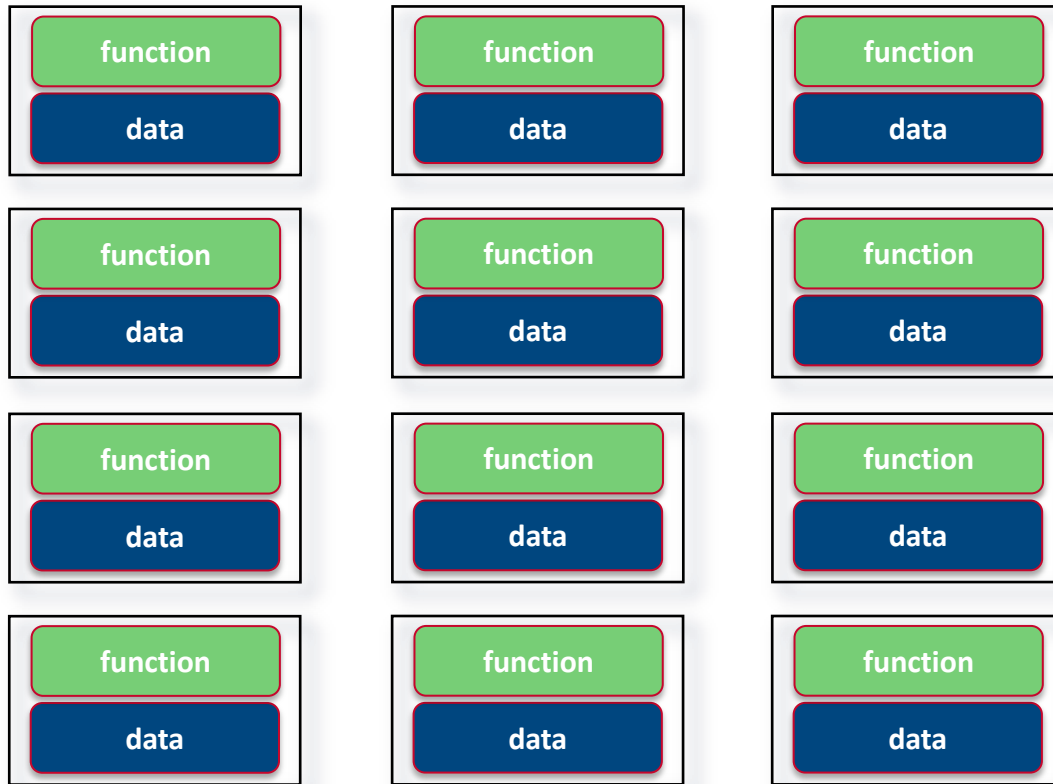
- Can it do 100 GB/s?
- 1 TB/s ?
- 10 TB/s?
- What happens when 100 more nodes are added?
- Another 10 PB of data is added?

Expensive!



Hadoop scales infinitely

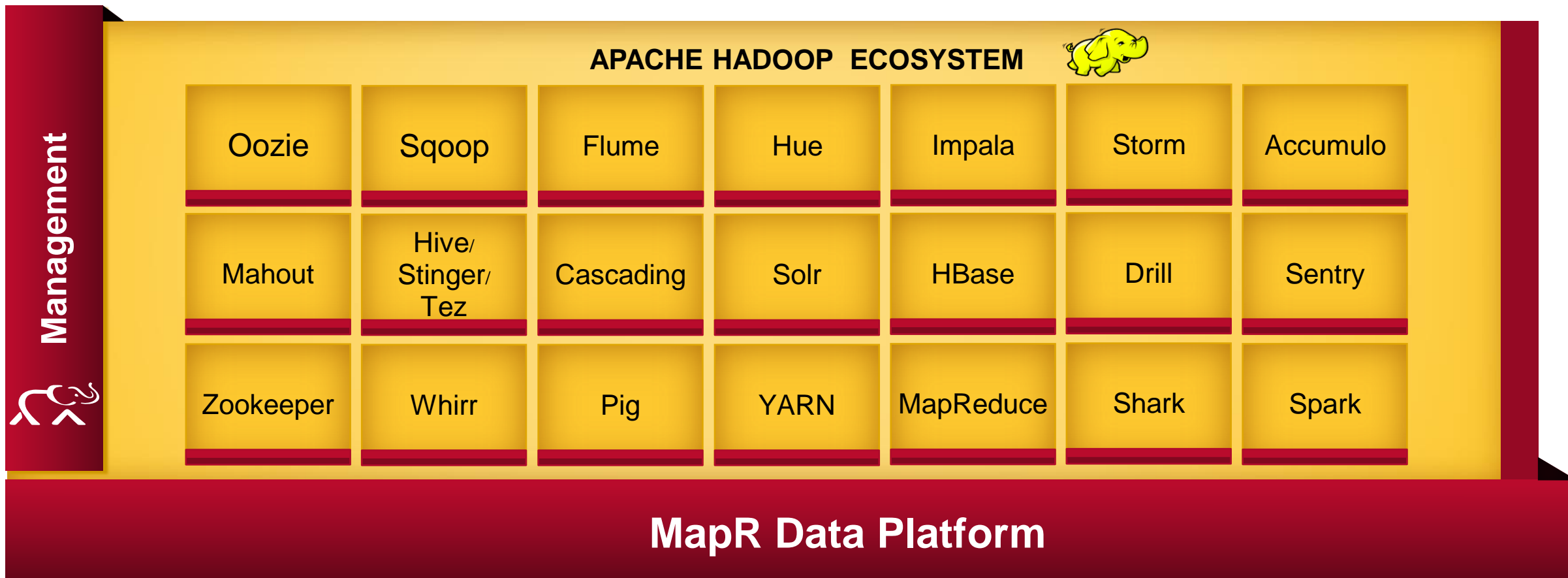
Hadoop



- Uses local drives
- Distributes data at time of *write*
- Runs compute on nodes where data is
- 10 drives per node
 - 1 GB/s per node
 - 100 GB/s on 100 nodes
 - 1 TB/s on 1000 nodes
- Scales automatically



Enterprise Hadoop from MapR



Enterprise-grade



Inter-operability



Multi-tenancy




Security



Operational





What shall we do
(with Big Data)?

I don't know ...
what do you
want to do?



“Simple algorithms and lots of data trump complex models”

Halevy, Norvig, and Pereira, *Google*
IEEE Intelligent Systems



Use Cases

- Web indexing
- Trending & clustering – new correlations
- Genome sequencing
- Surveillance & threat-detection
- Image recognition
- Risk Management
- Protein folding
- etc, etc, etc



What's Driving Hadoop Adoption?

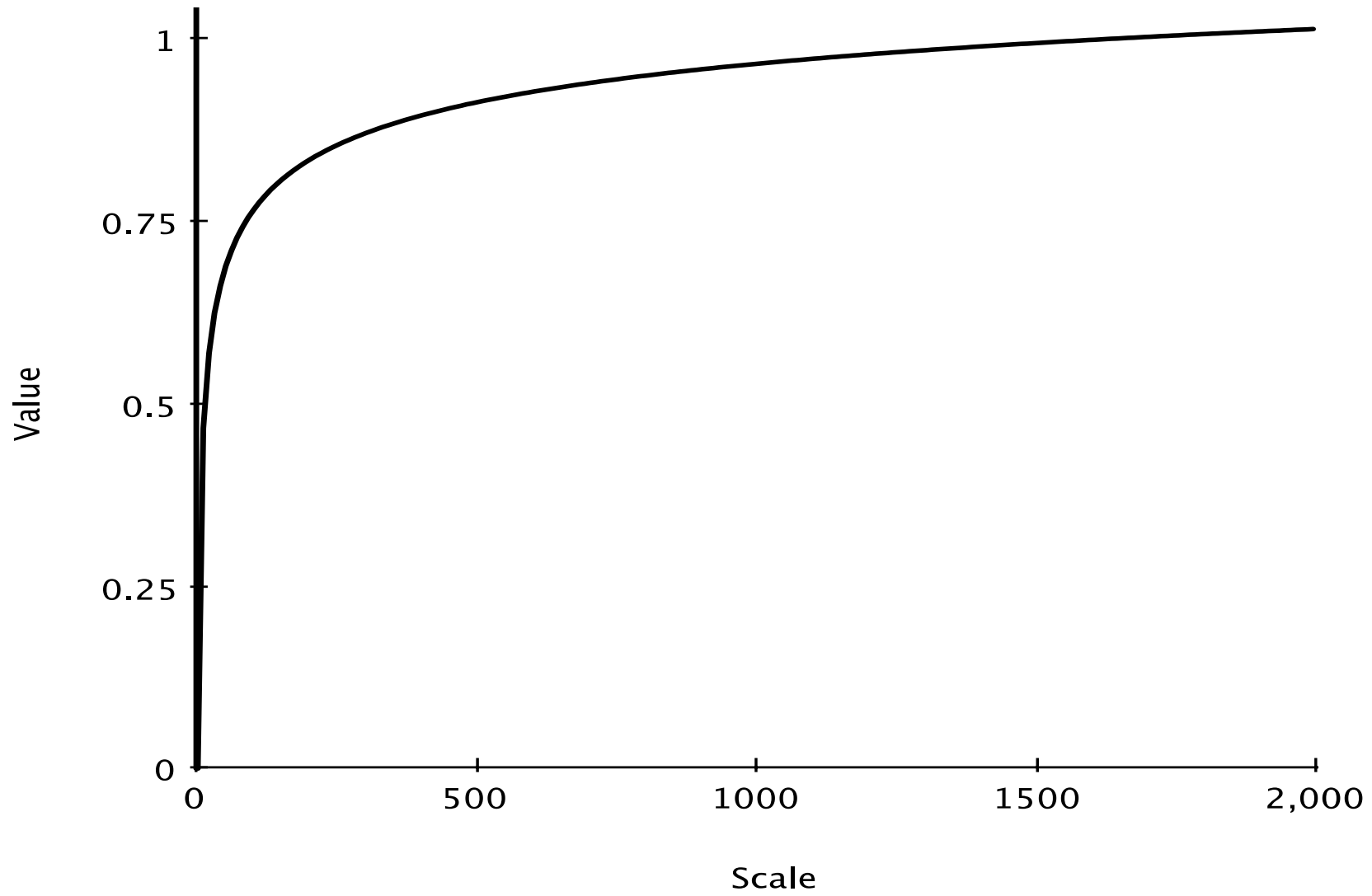
- No schema required
 - No need to anticipate what you will ask
 - Manipulate instead of just query
 - Flexible
 - Saves Time

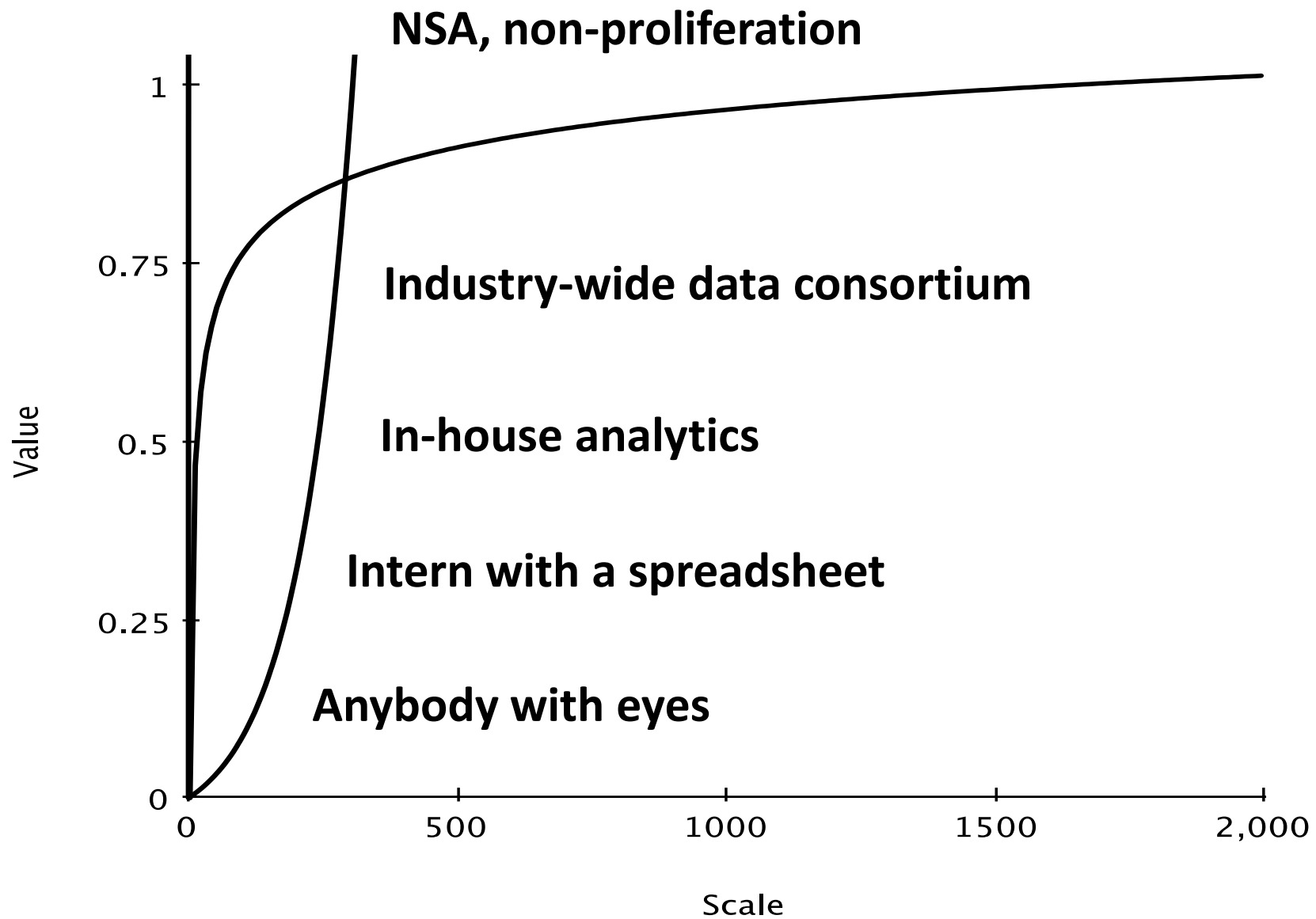


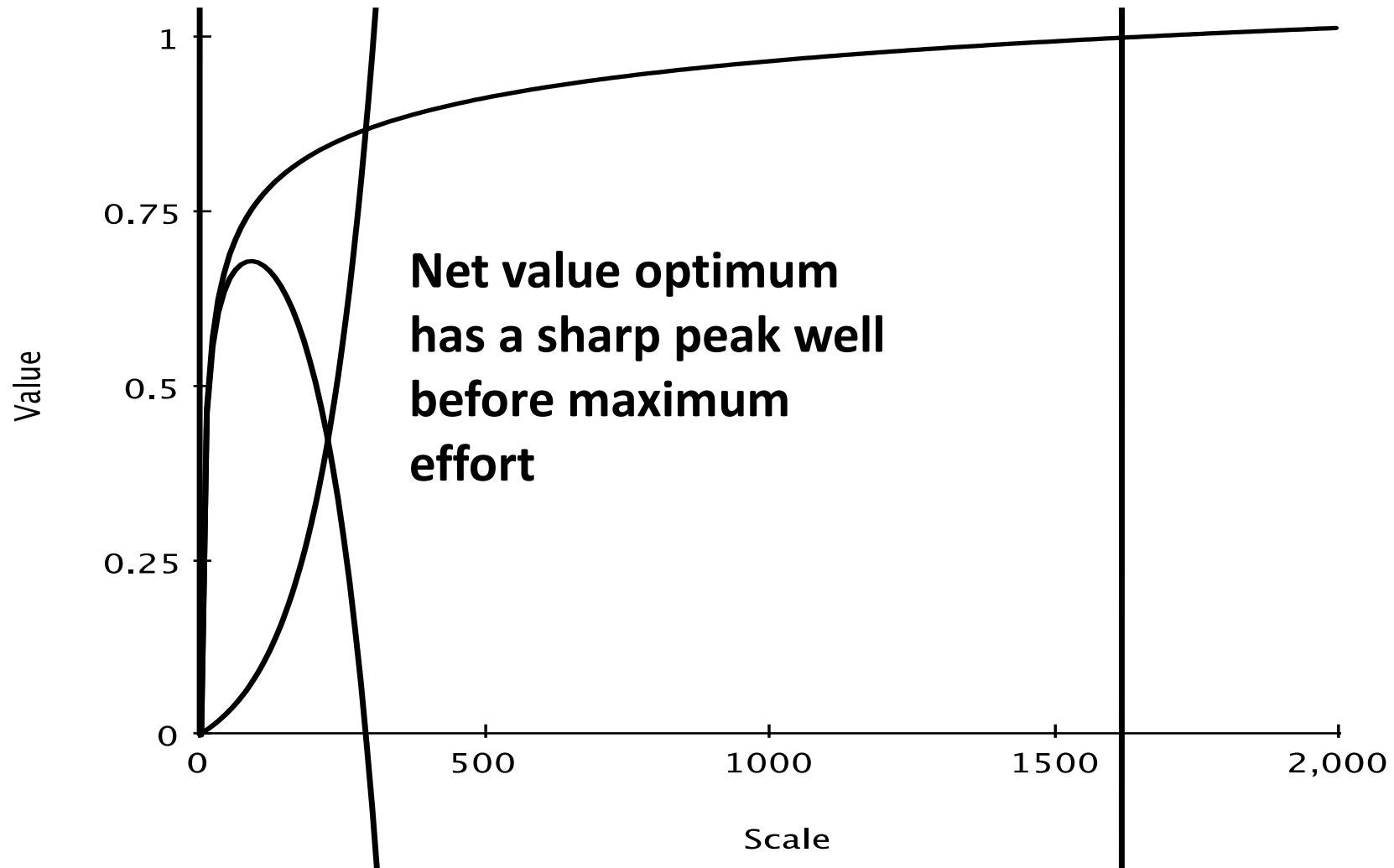
What's Driving Hadoop Adoption?

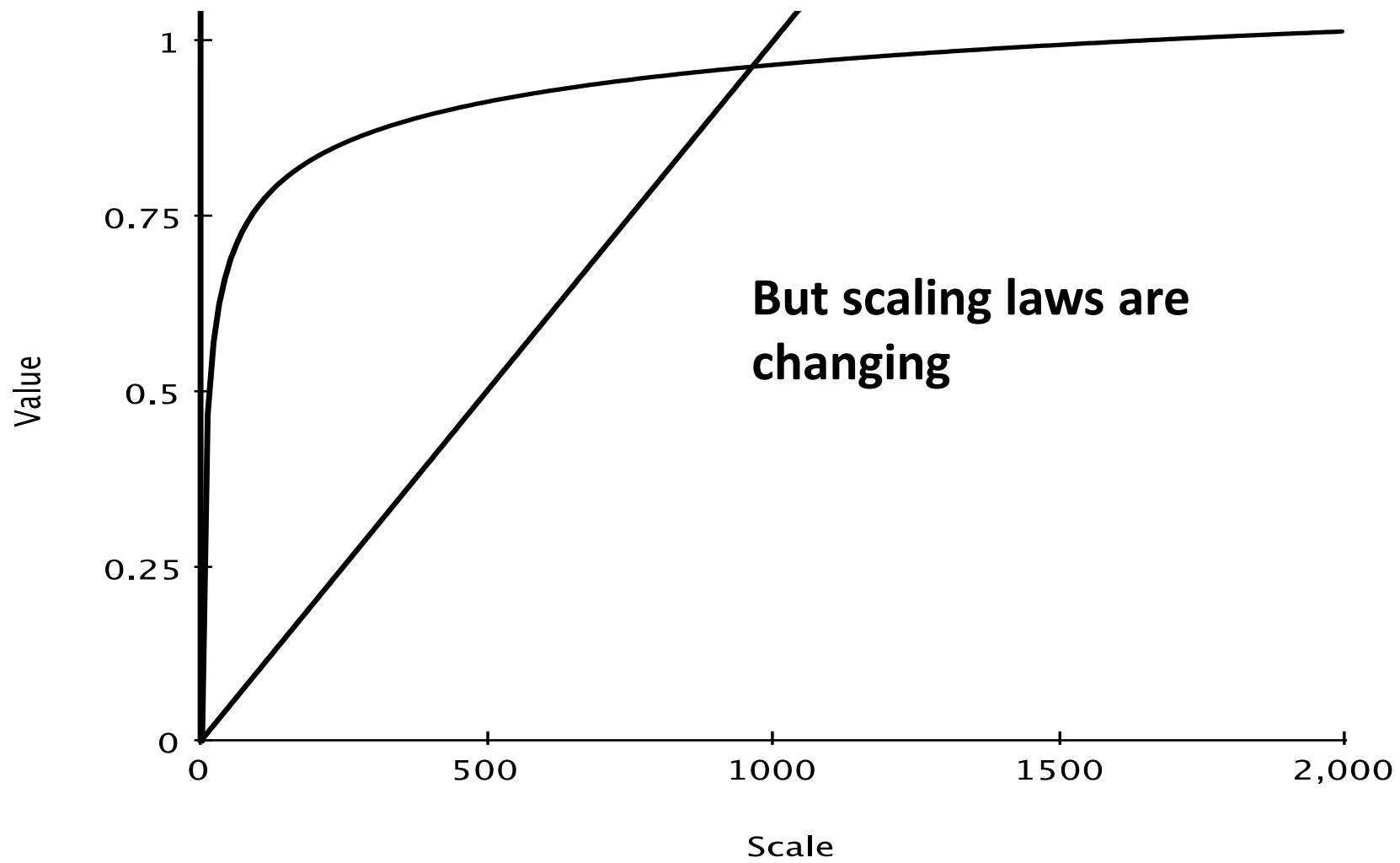
- Analytics scaling is all about the 80-20 rule
 - Big gains for little initial effort
 - Rapidly diminishing returns
- The key to net value is how costs scale
 - Old school – exponential scaling
 - Big data – linear scaling, low constant
- Cost/performance has changed radically
 - With the use of many commodity boxes

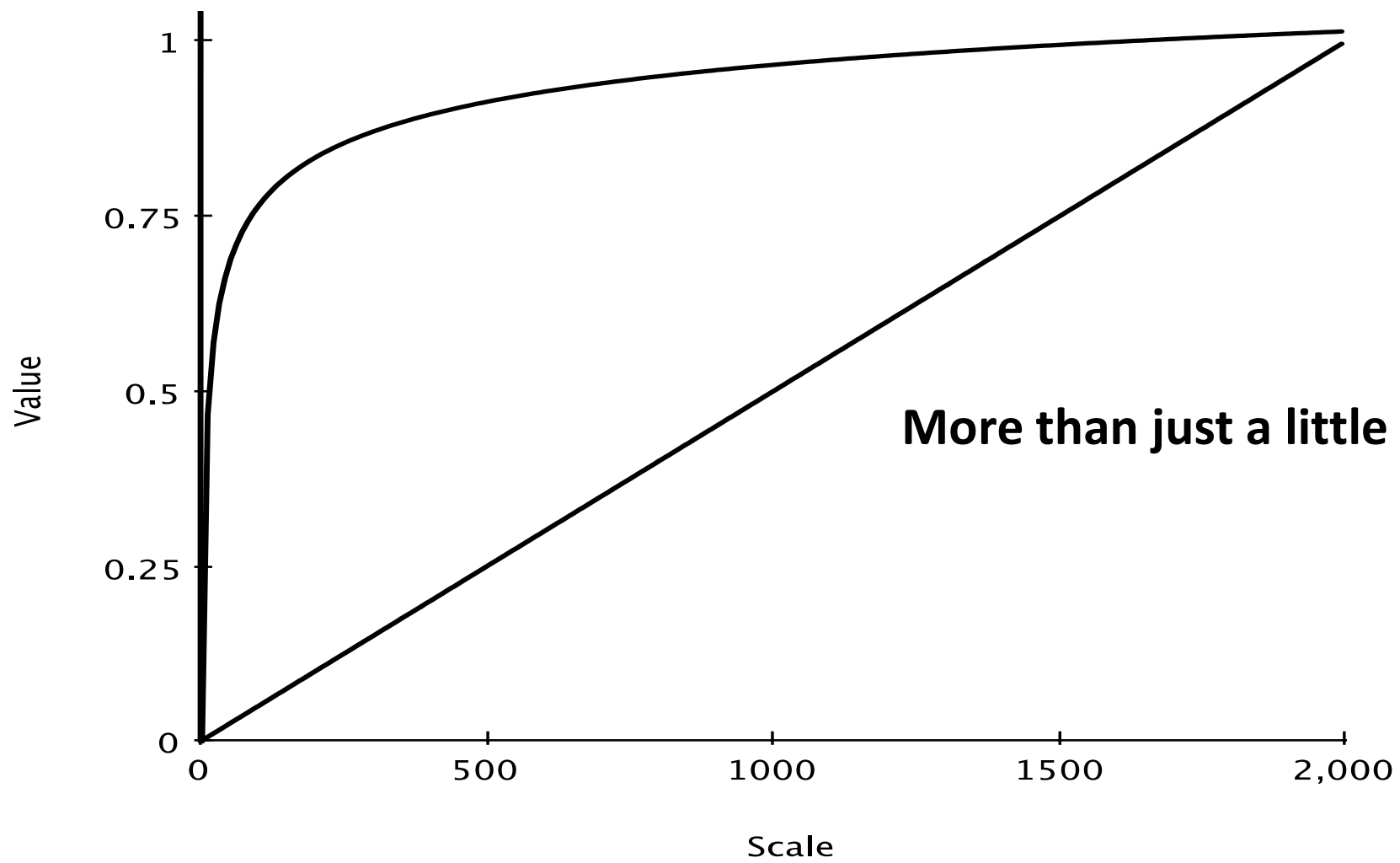


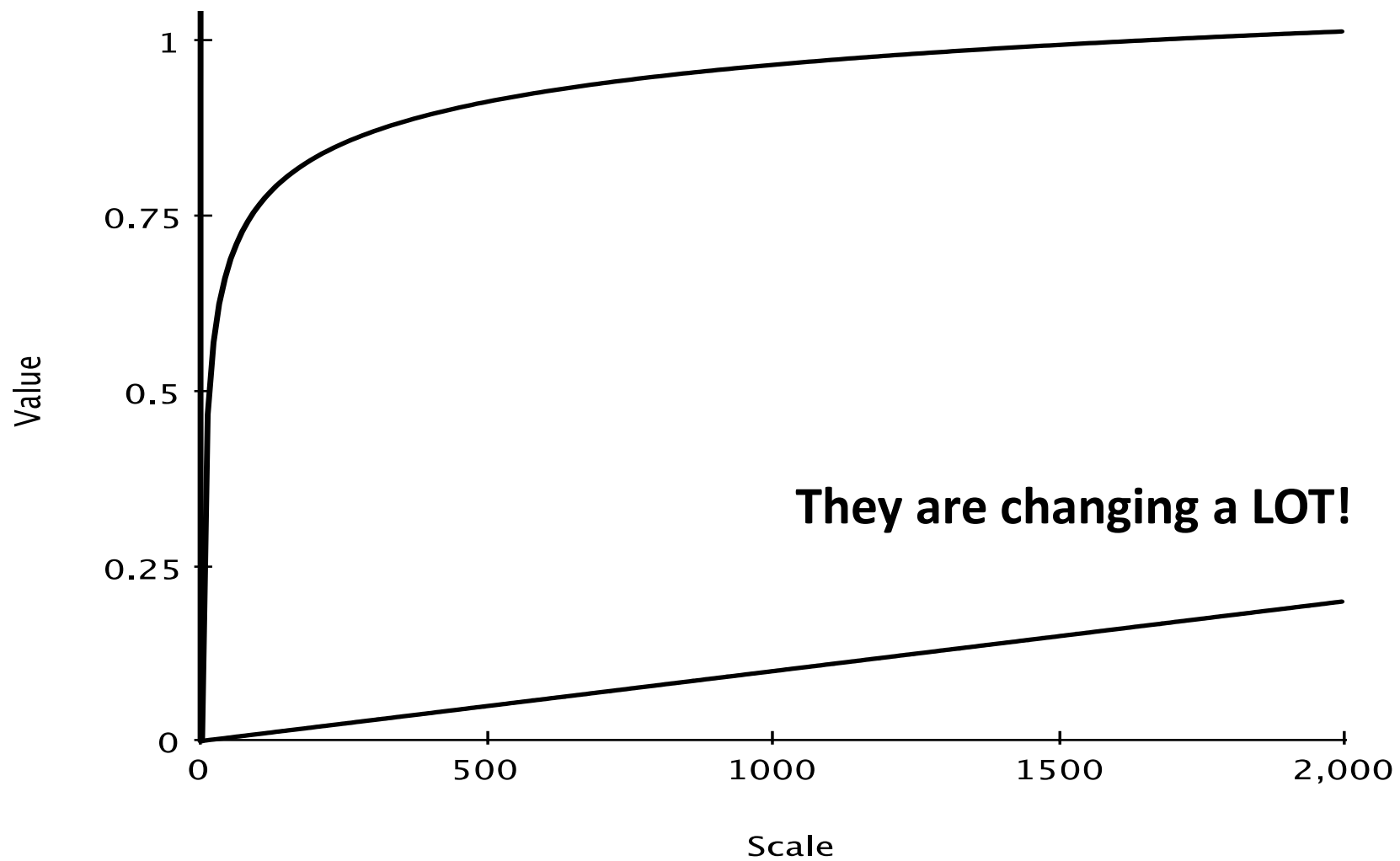












Hadoop Adoption

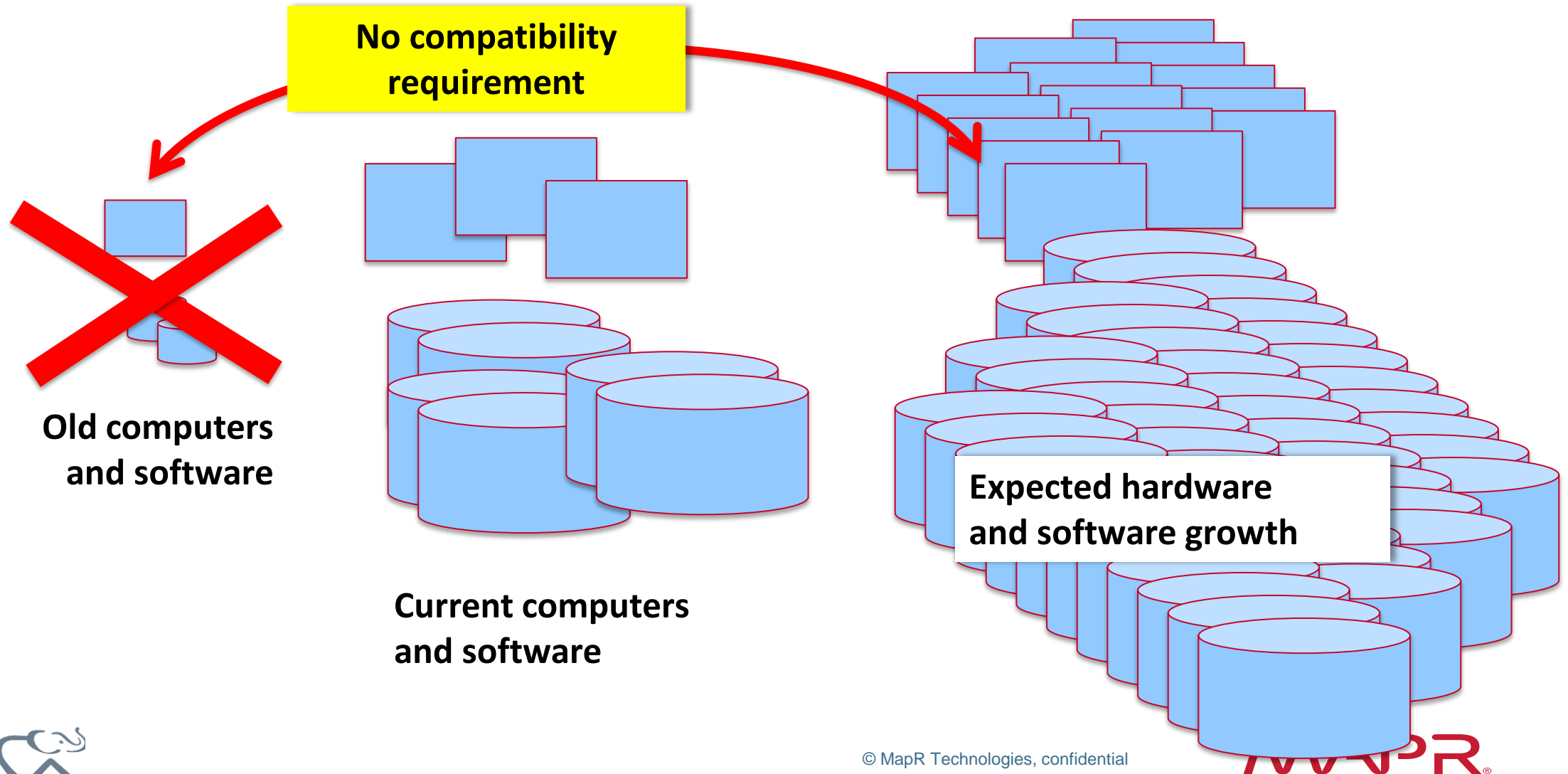
- From 2007 until now, Hadoop has exploded
- Strong adoption in web and startup community
- Weaker adoption in large-scale business
 - But lots of interest!



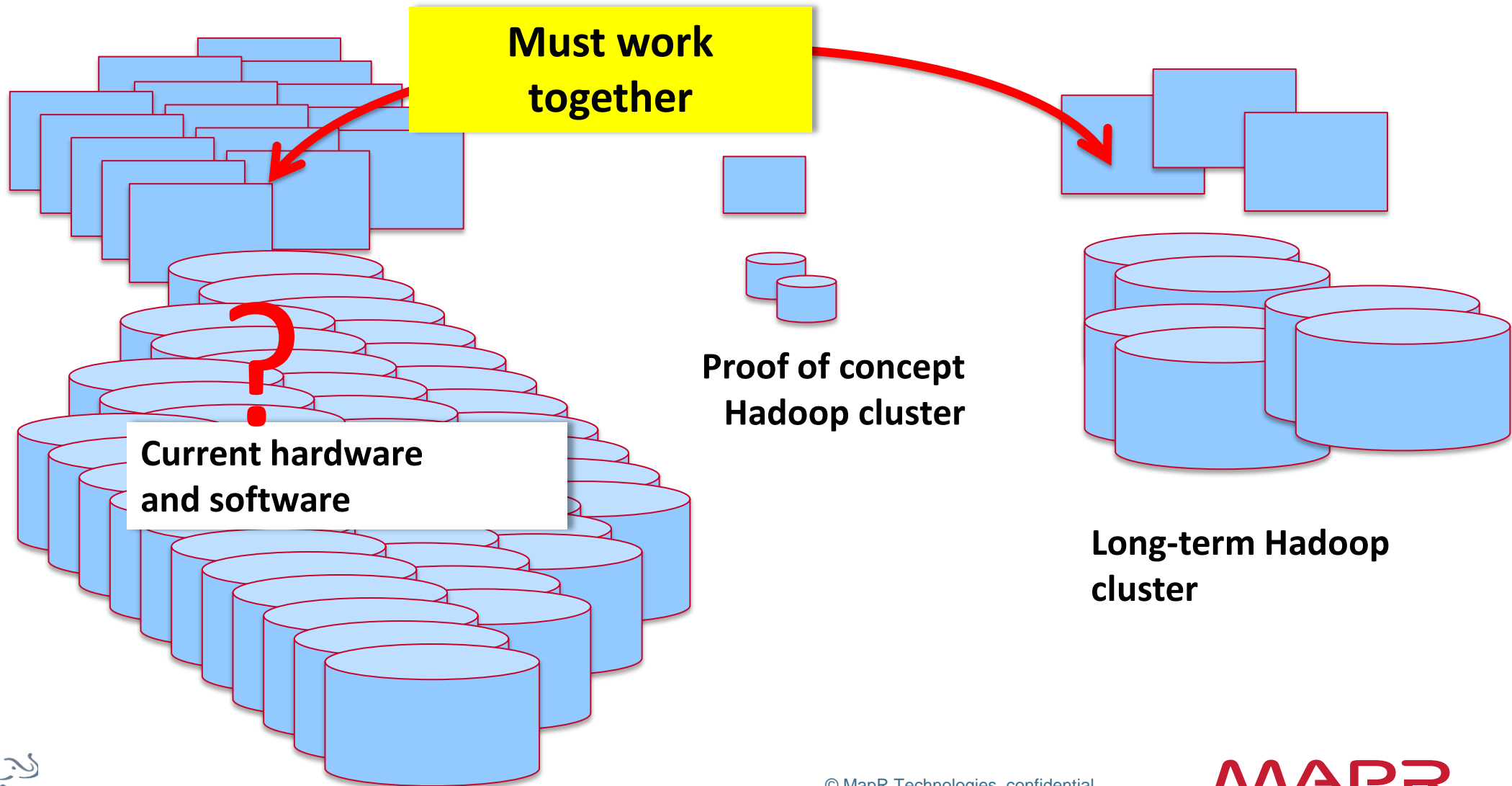
Why is this different?



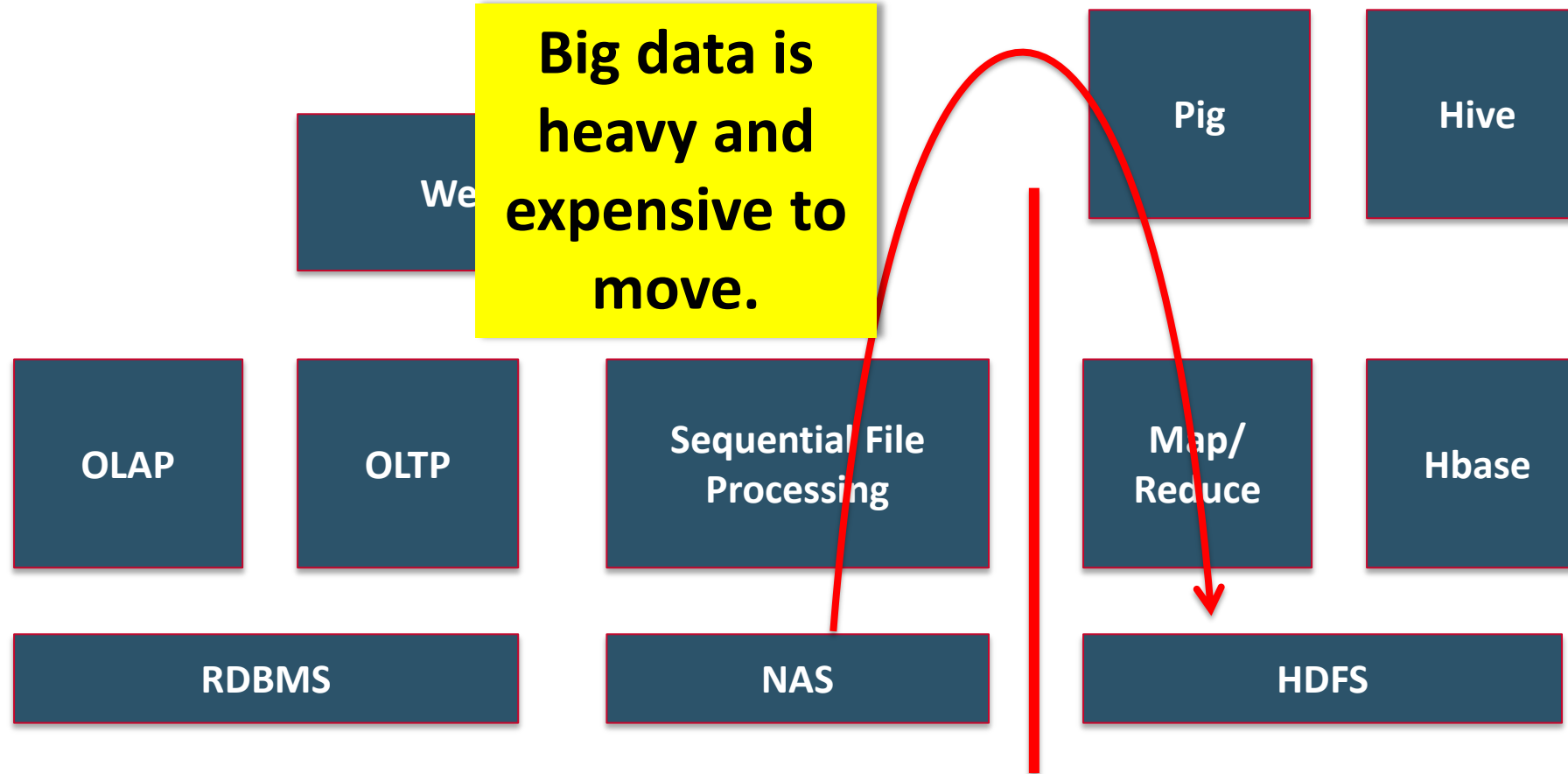
The startup technology picture



The large enterprise picture



Narrow Foundations



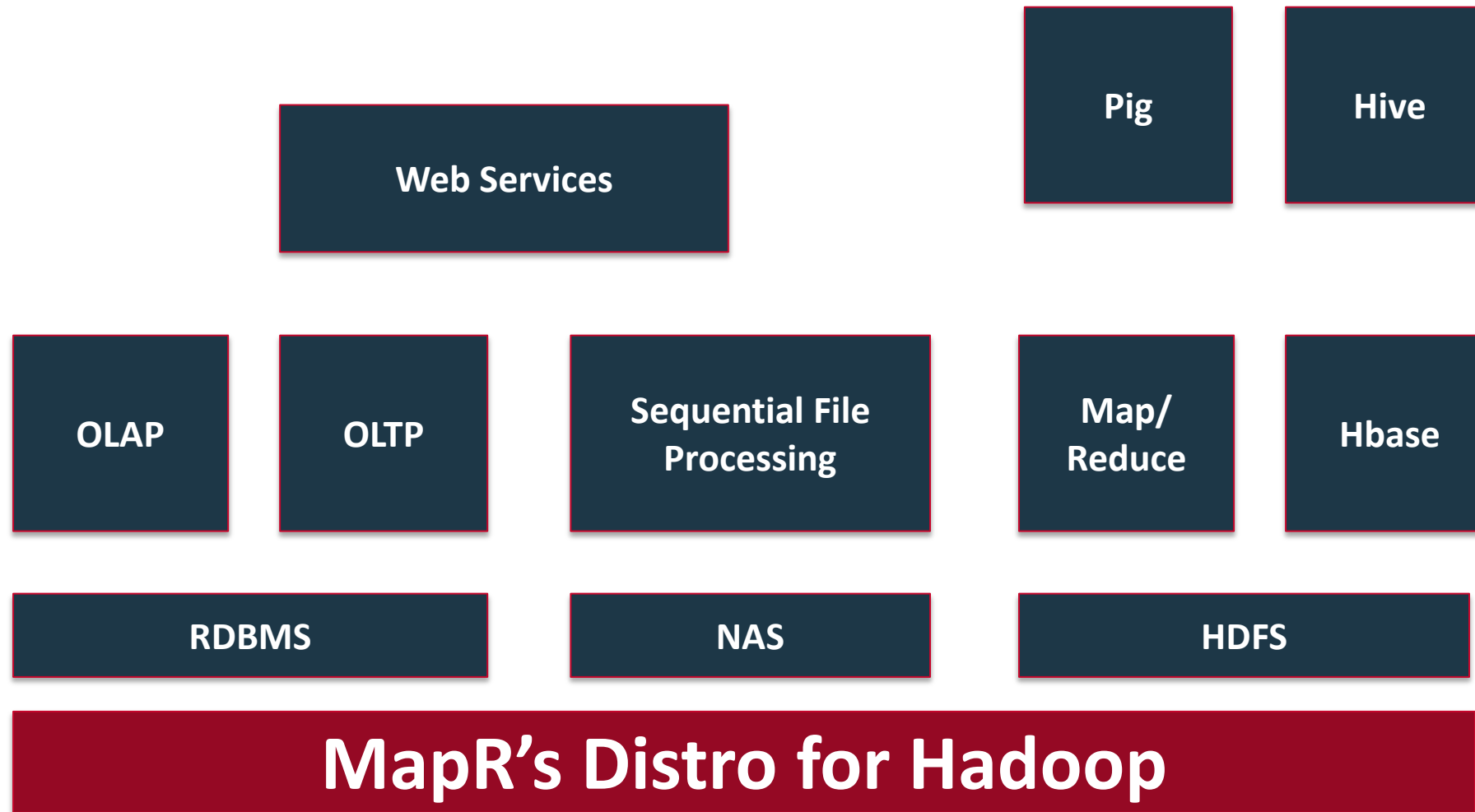


One Possible Answer

- Widen the foundation
- Use standard communication protocols
- Allow conventional processing to share with parallel processing



Broad Foundation





Why build a new storage system?



Technology Disruption in last 15 years

	<i>2000</i>	<i>2013</i>	
CPU Cores	2	128	100 x
DRAM	2G	1T	1000 x
Disk Size	32G	8T	250 x
# Spindles	250	4	1/100 x
Cluster Size	< 10	100 – 10,000	100 x



Why build a new storage system?

- “current” file system technology is 10 years old
 - Sun's ZFS, Spinnaker, Isilon, Panasas
 - ext3/ext4, xfs, reiserfs are even older ... ceph not even a contender
- Run a client on the server? NO WAY
- Cannot handle massive clustering
 - cluster too static
 - volumes too large or too small
 - replication added as an afterthought



The Cloud Leaders Pick MapR



Elastic Map-Reduce (EMR) is the largest Hadoop provider in revenue and # of clusters



Google chose MapR to provide Hadoop on the Google Compute Engine



What Makes MapR so Reliable?



Reliability with Commodity Hardware

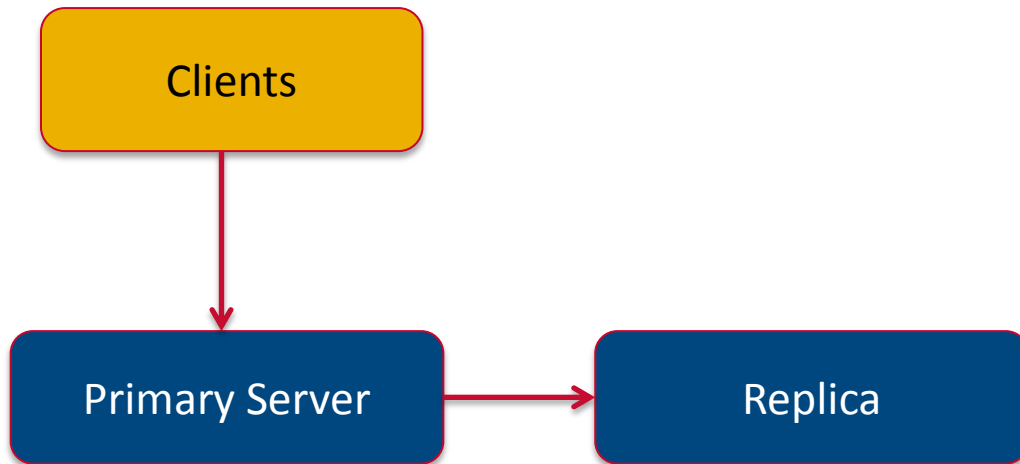
- No NVRAM
- Cannot assume special connectivity
 - No separate data paths for "online" vs. replica traffic
- Cannot even assume more than 1 drive per node
 - No RAID possible
- Use replication, but ...
 - What if one node is 10x larger than the other?



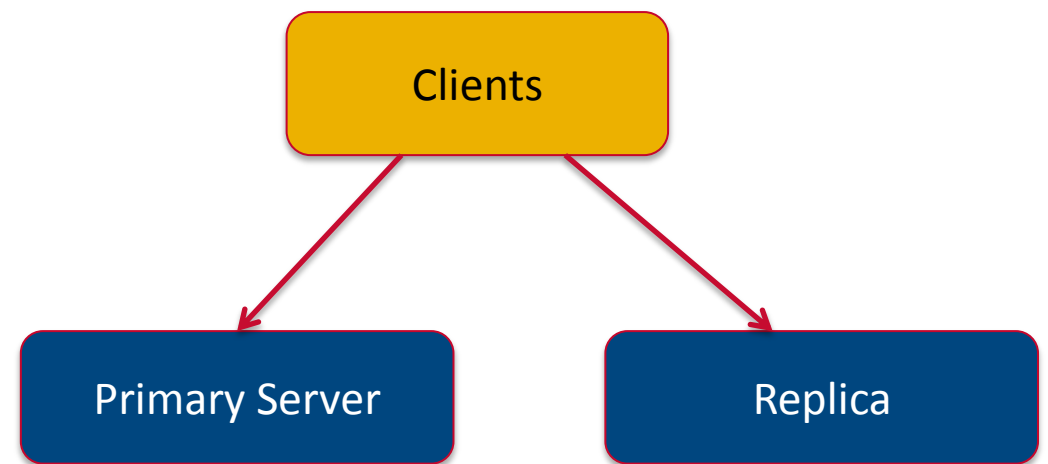
Reliability via Replication

Replication is easy, right?

All we have to do is send the same bits to the master and replica



Normal replication, primary forwards



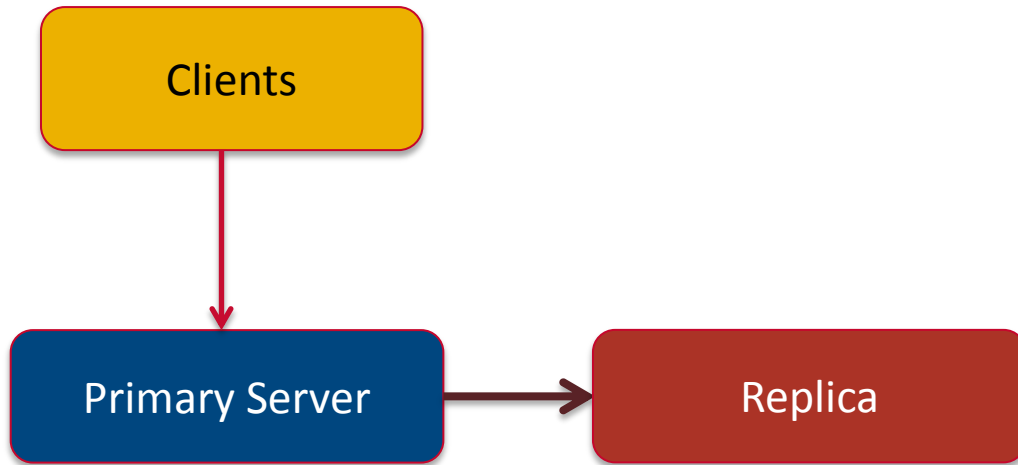
Cassandra-style replication



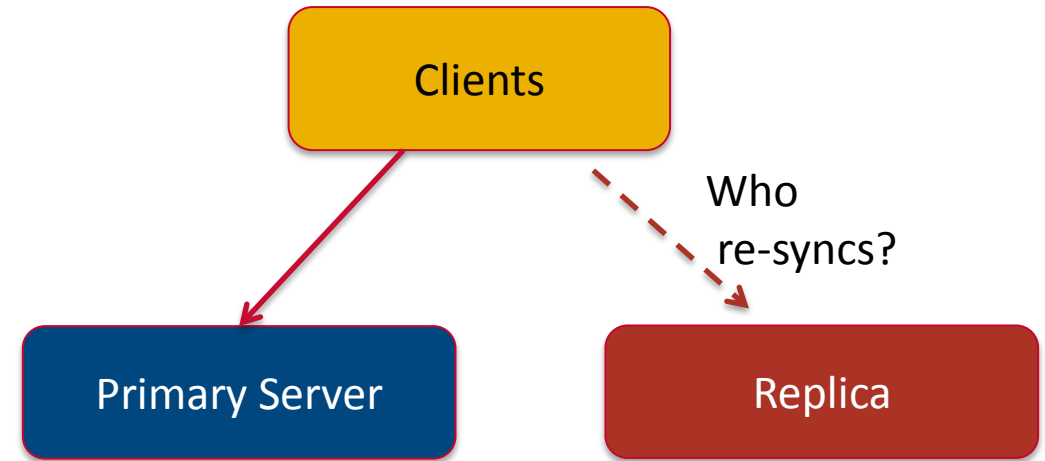
But Crashes Occur...

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 Apache HDFS ...

- Apache 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
- Summary: ***It is OK to delete data if no one saw it !!***



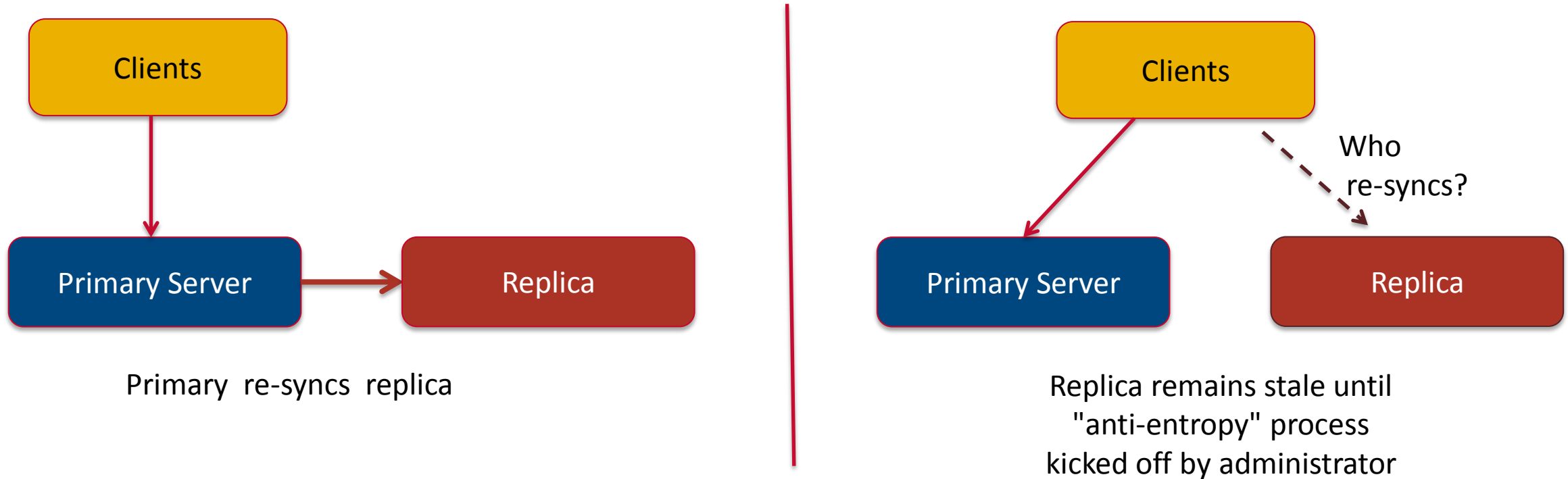
Apache HDFS meets its Design Goal

- 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
- Real-time 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 function 'normally', update ability needed

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



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?

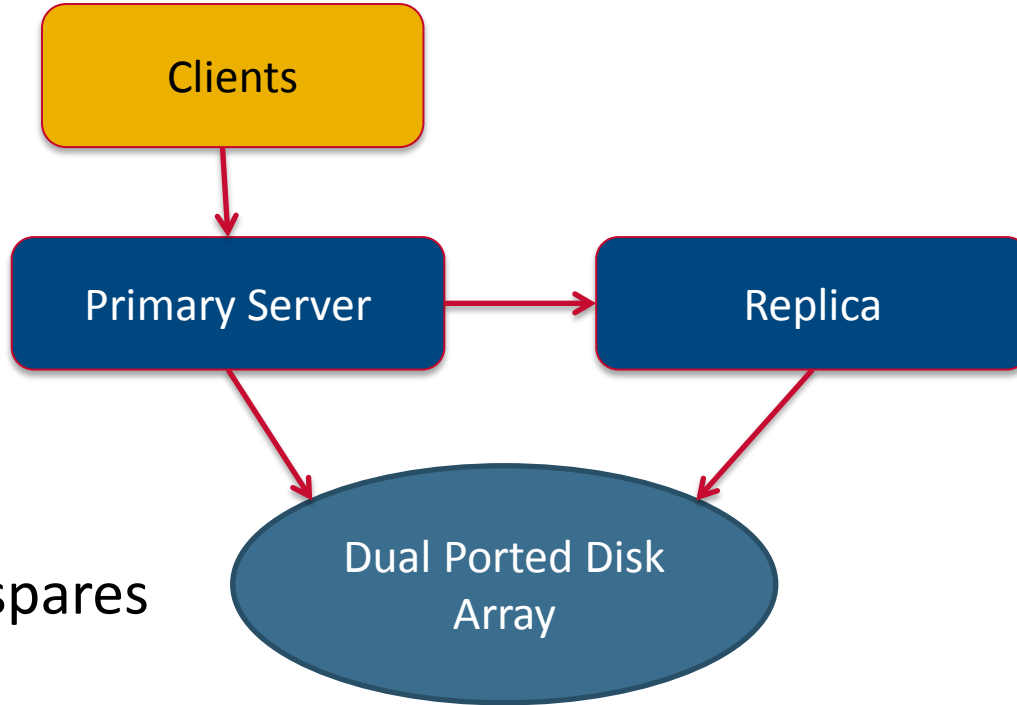


Traditional Solutions

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



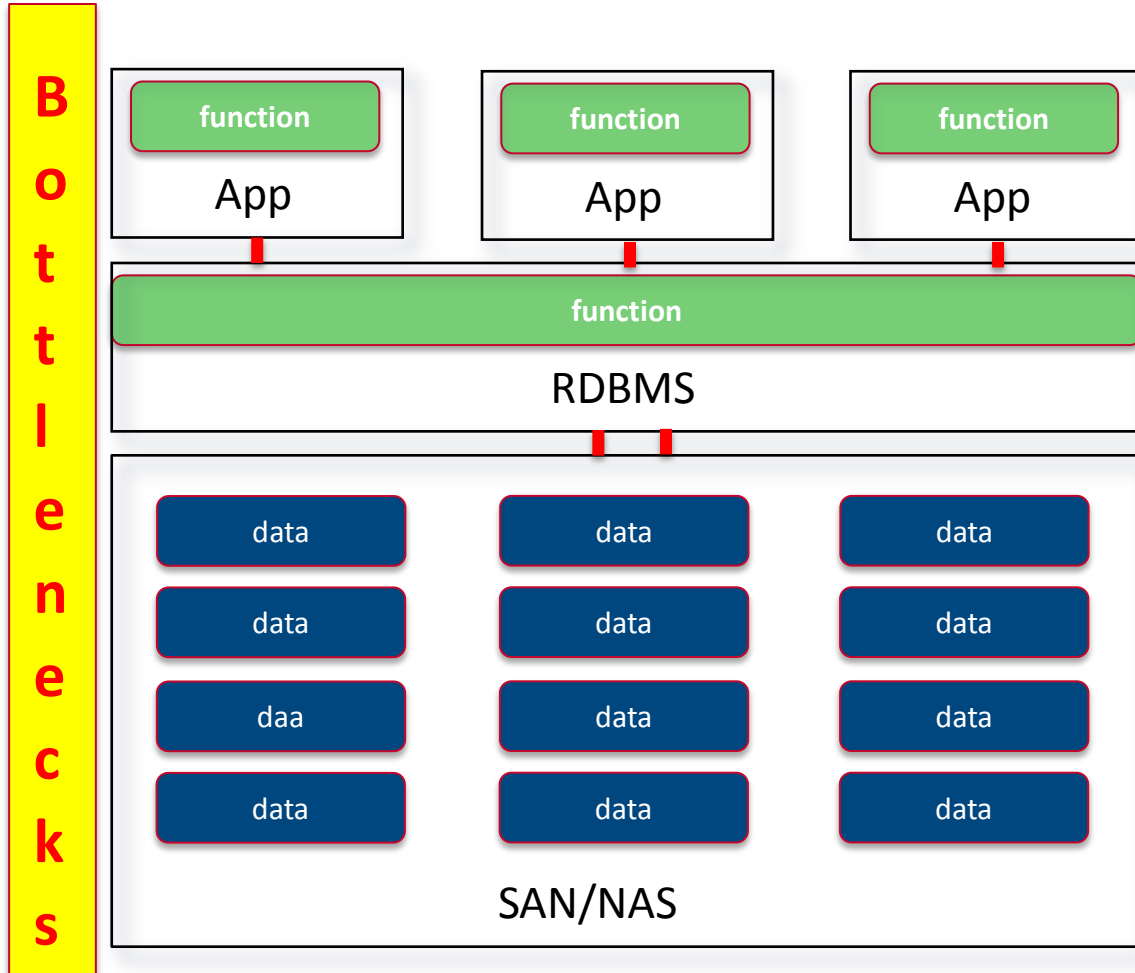


BIG DATA? YOU CAN'T HANDLE BIG DATA

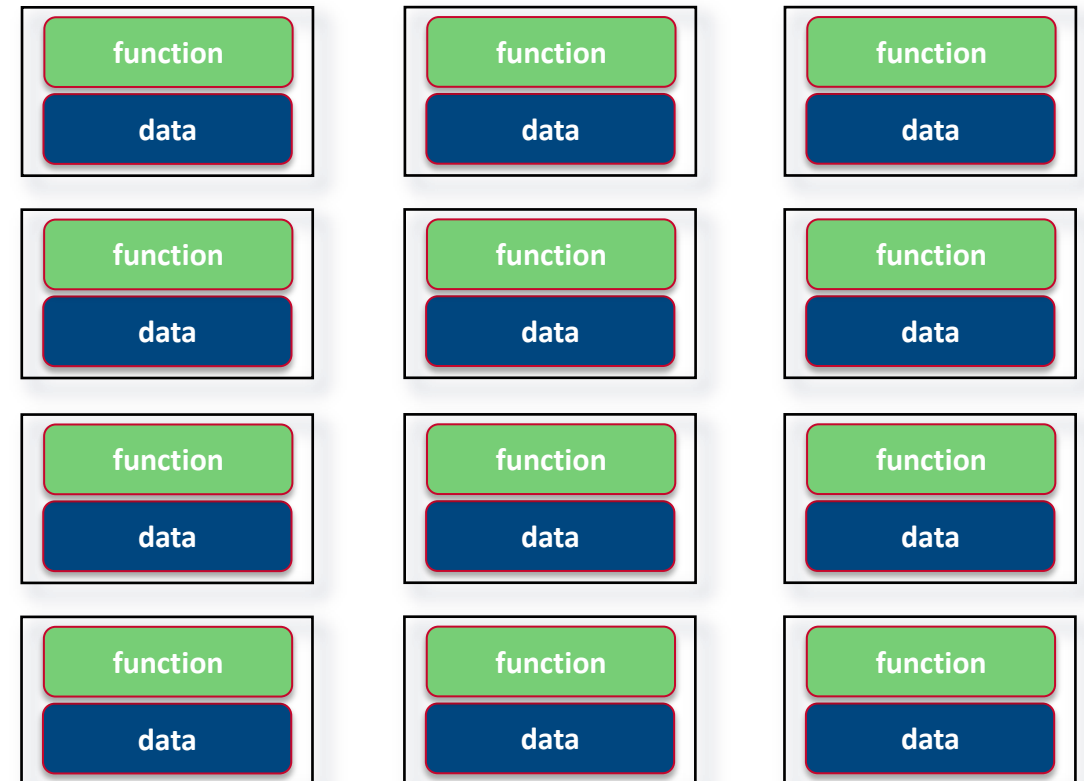


Forget Performance?

Traditional Architecture




Hadoop

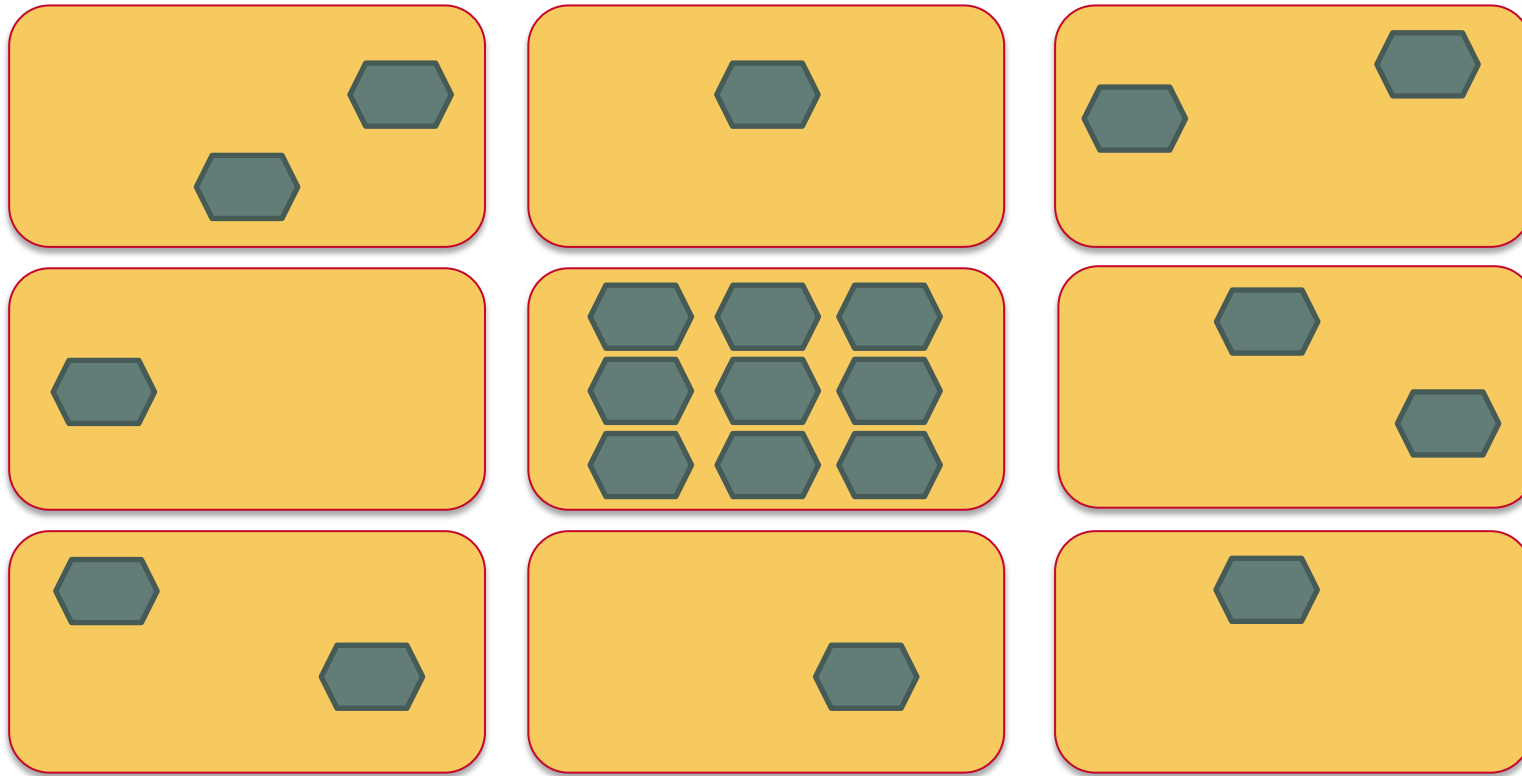


Geographically dispersed?



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

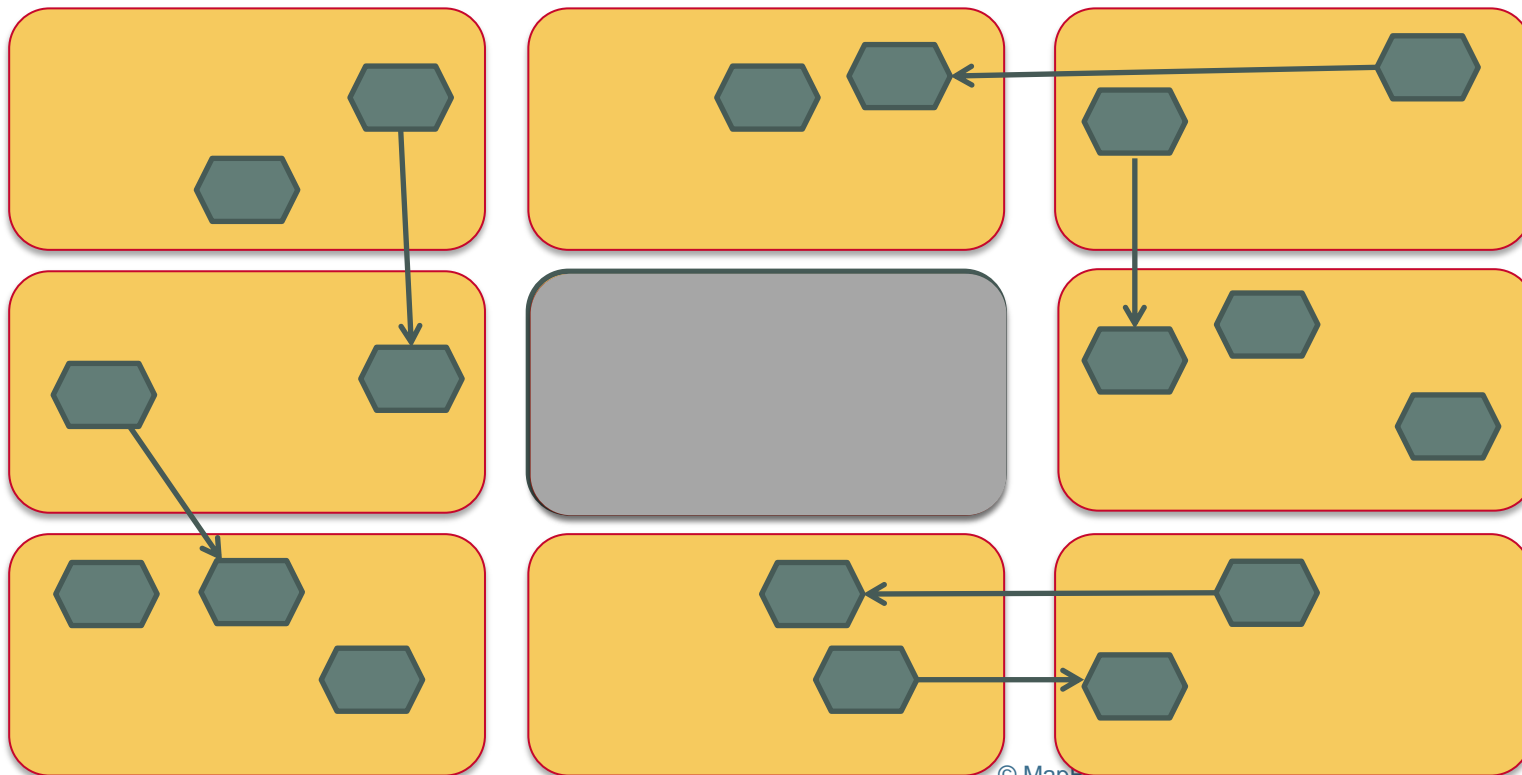


Why Does It Improve Things?



MapR Replication Example

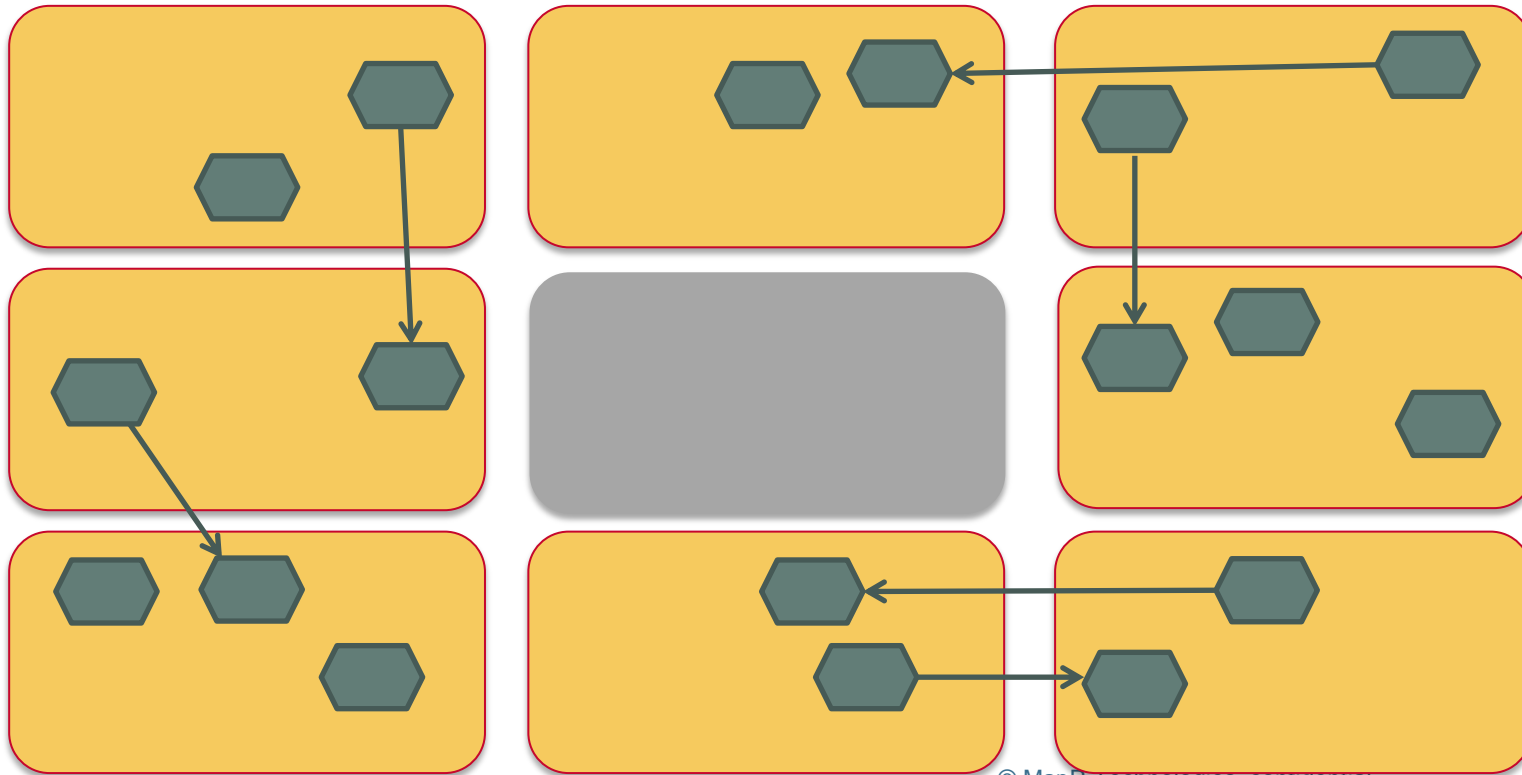
- 100-node cluster
- Each node holds 1/100th of every node's data
- When a server dies, entire cluster re-syncs the dead node's data



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 $1/100^{\text{th}}$ of the data

- Net speed up is 100x
 - 3.5 hours instead of 350
- **MTTDL is 100x better**



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!

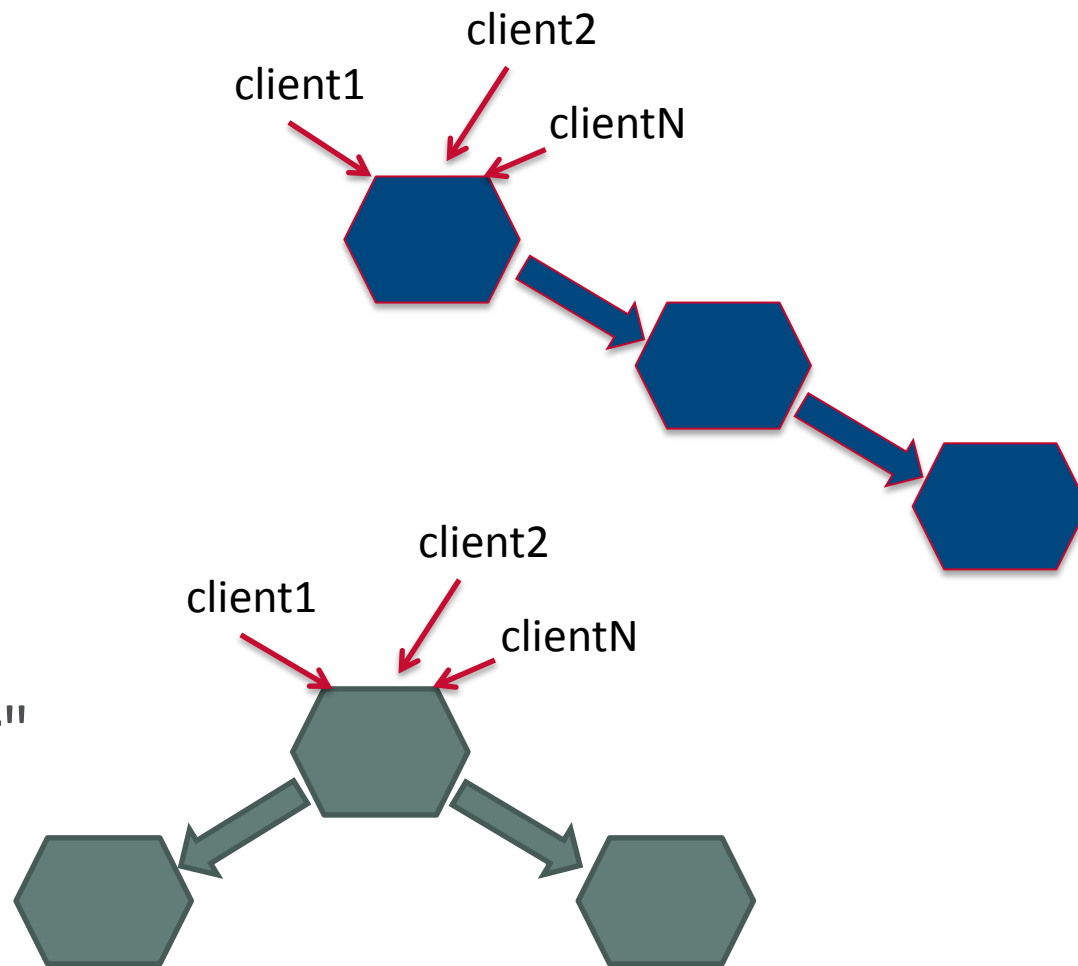


Why Is This So Difficult?



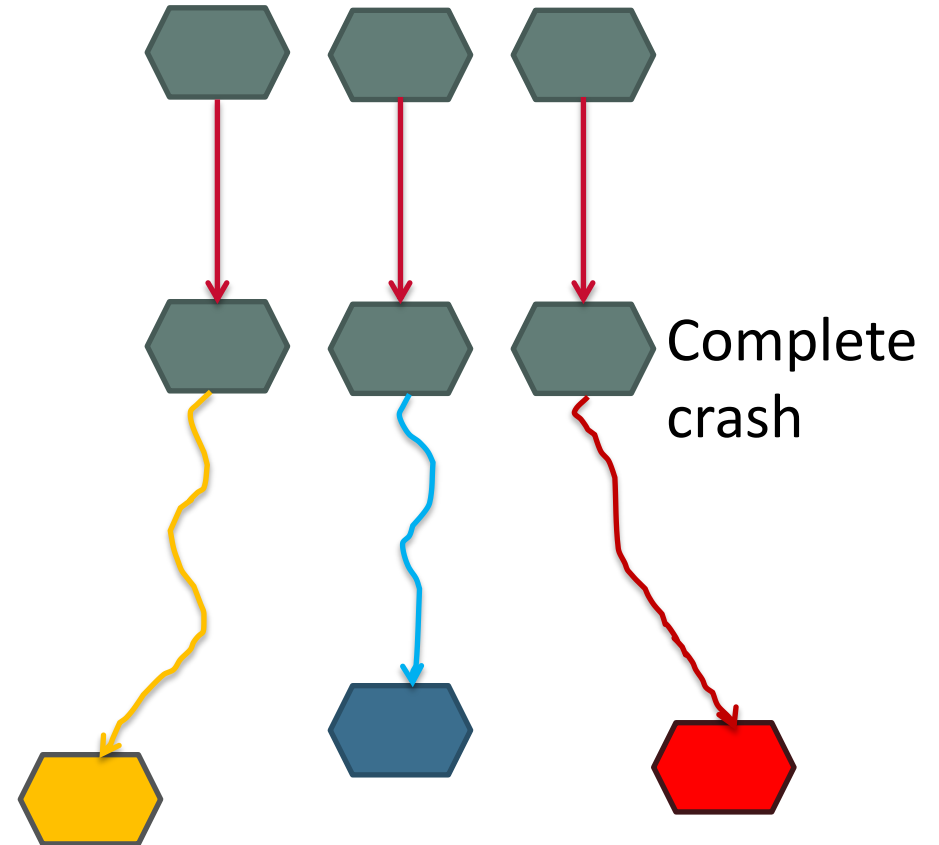
MapR's Strong Consistency

- 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



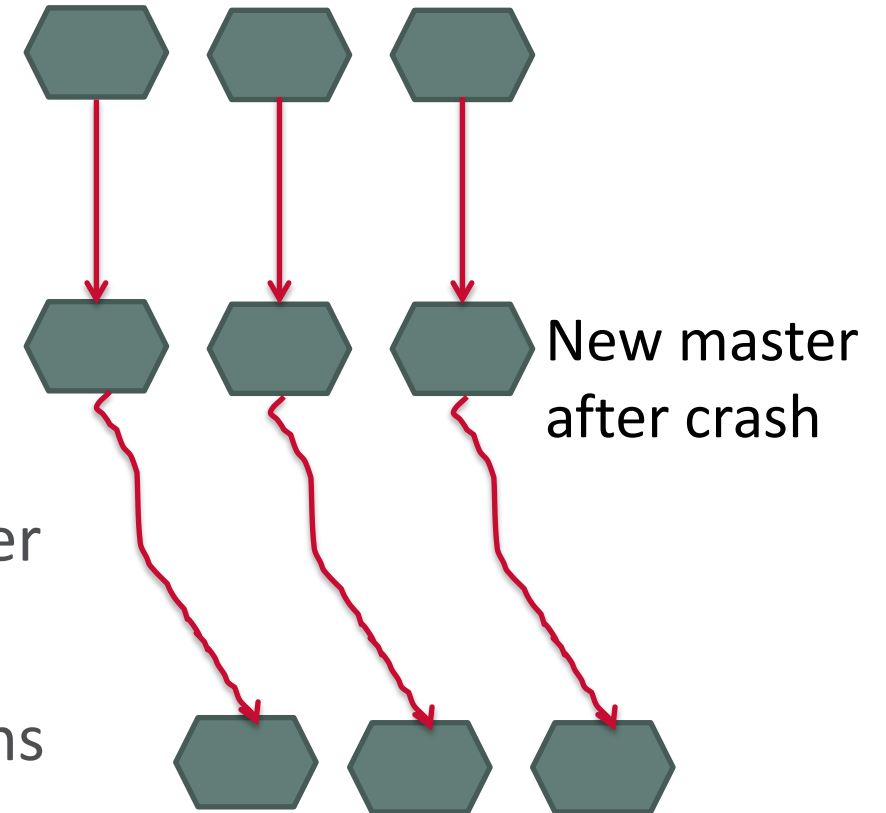
MapR Synchronous Replication

- 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



Contrast MapR with

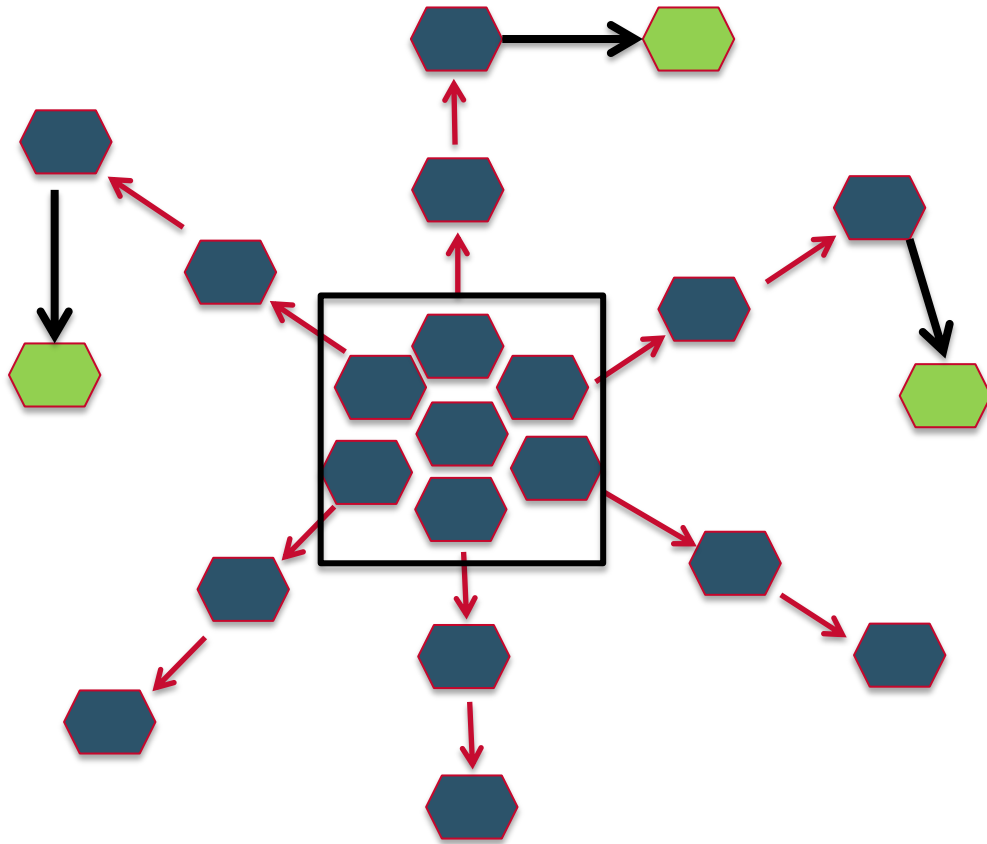
- Traditional NAS and SAN
 - relies on custom hardware
- Cassandra
 - declares it inconsistent
- Apache HDFS
 - deletes it

Only MapR provides

- strong consistency
- on shared-nothing commodity hardware



Container Balancing



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

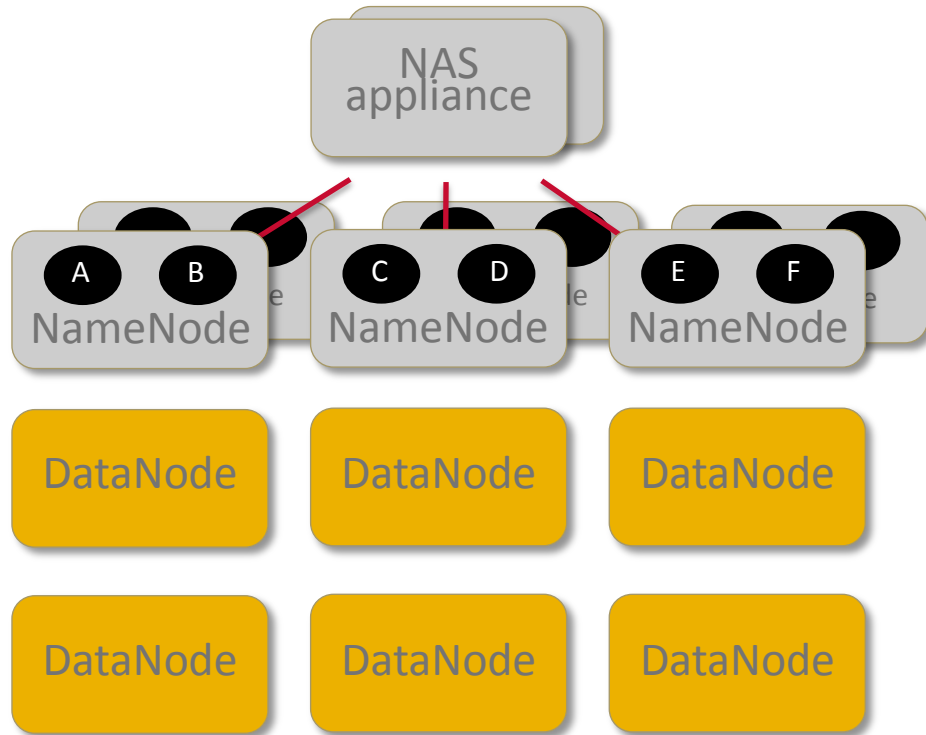


Where & How Does MapR Exploit This Unique Advantage?



MapR's No NameNode HA™ Architecture

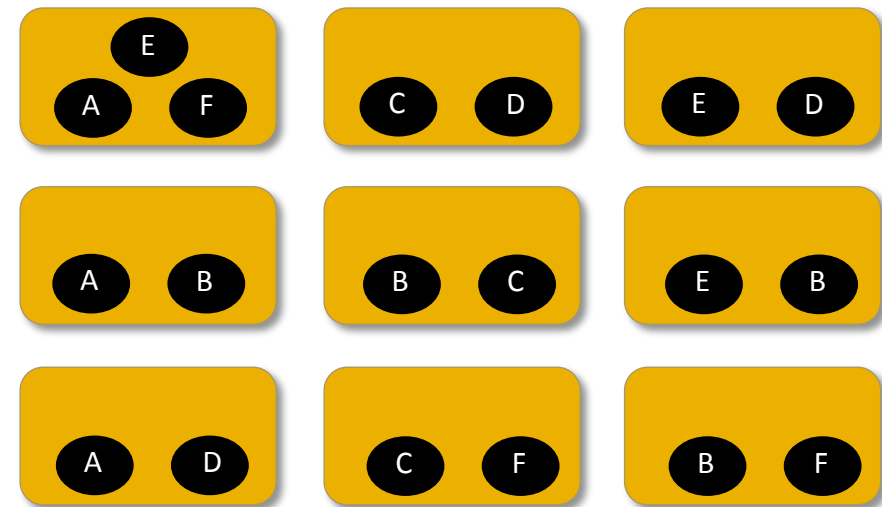
HDFS Federation



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

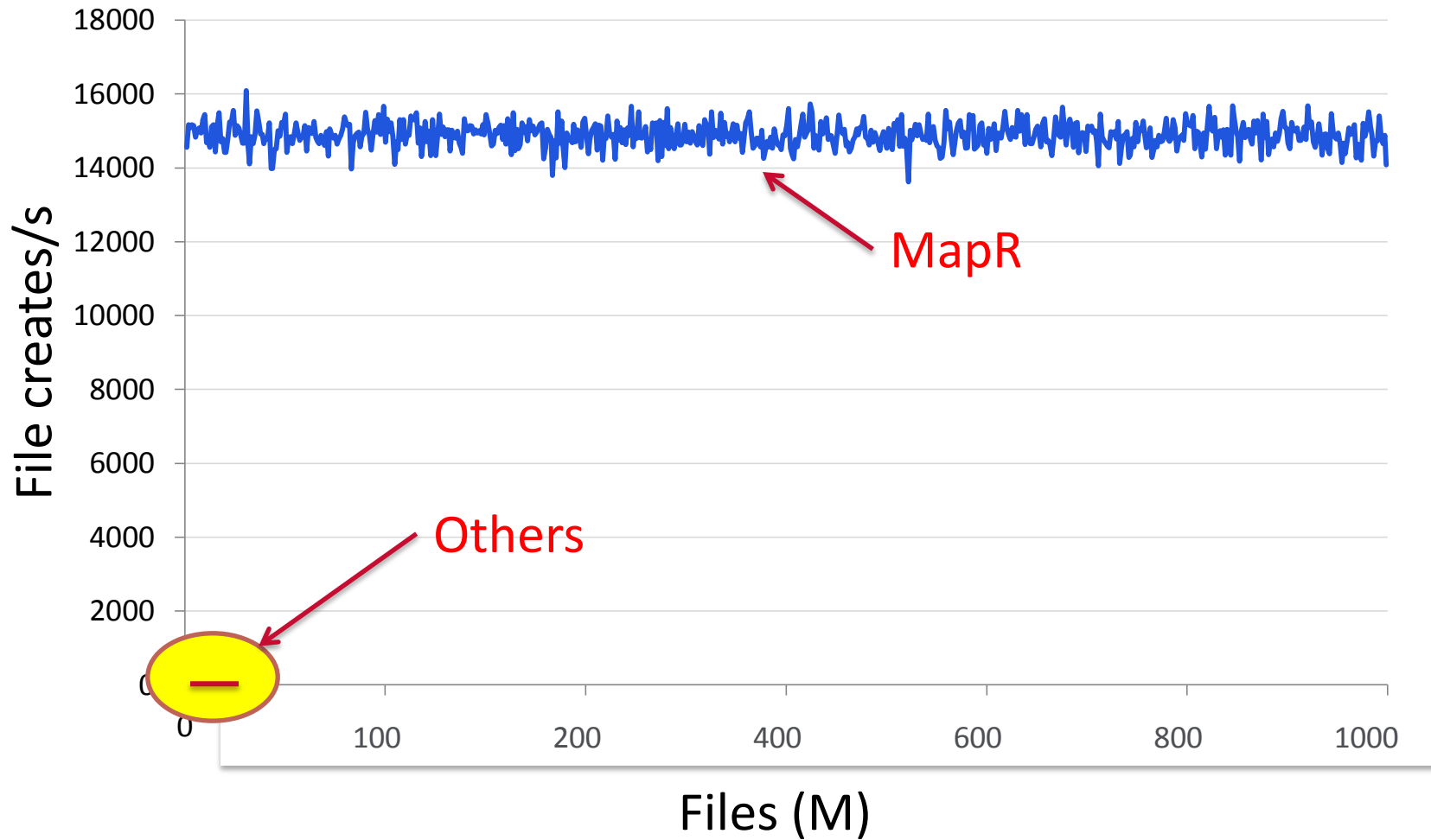


MapR (Distributed Metadata)



- Fully HA w/automatic failover
- Up to 1 trillion files
- 10-20x higher performance
- 100% commodity hardware

Meta Data Performance



Benchmark:

Write 100-byte files

Hardware: 10 nodes

2 x 4 cores

24 GB RAM

12 x 1 TB 7200 RPM

1 GigE NIC



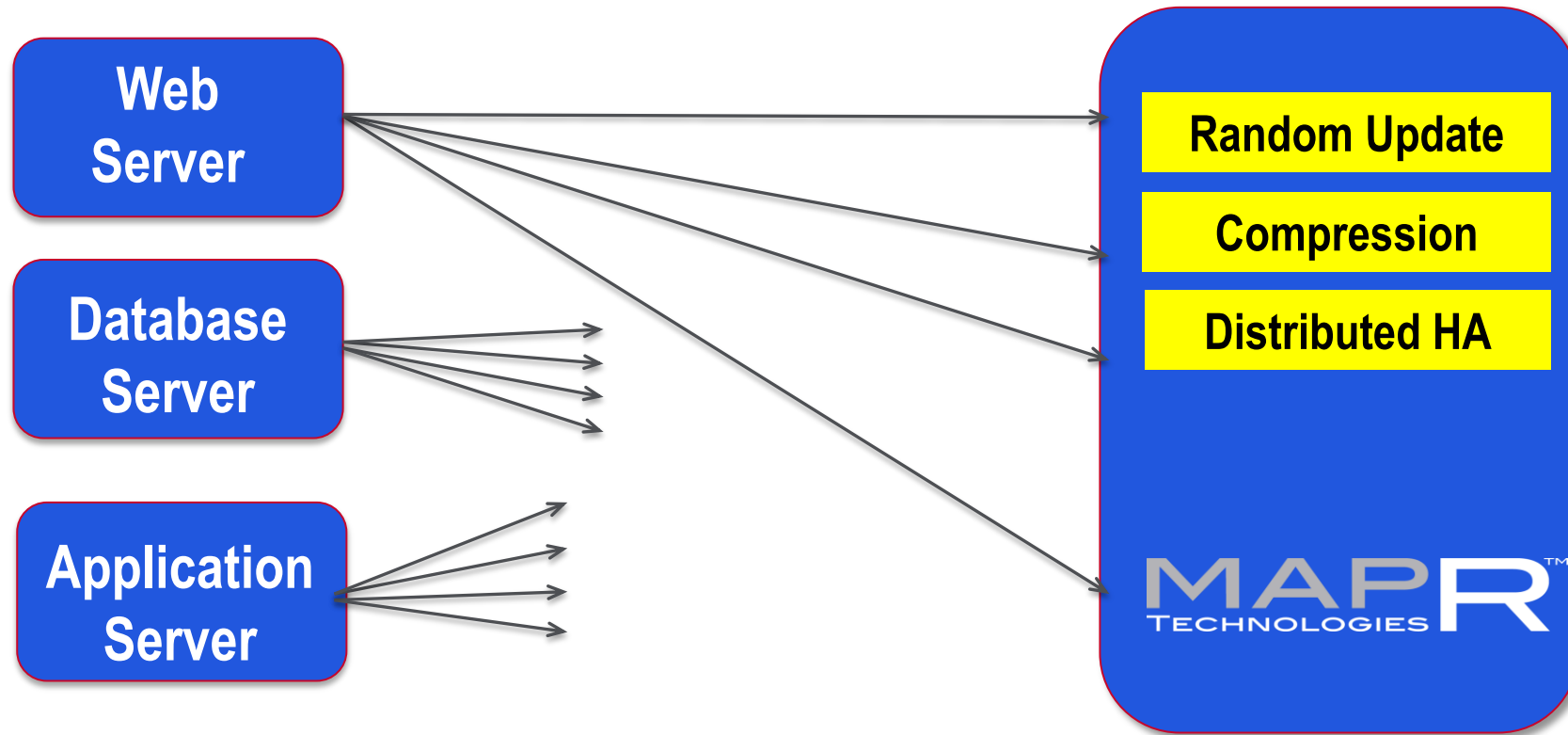
Where & How Does MapR Exploit This Unique Advantage?



MapR's NFS Allows Direct Deposit

Connectors not needed

Traditional applications integrate smoothly



MapR Scalability

MapR scales in every dimension

Cluster Size	10,000 nodes
Data in cluster	1-10 EB
Number of files	1 trillion
Number of volumes	100,000
Performance	3x - 20x
Footprint	50% smaller



MapR's Streaming Performance



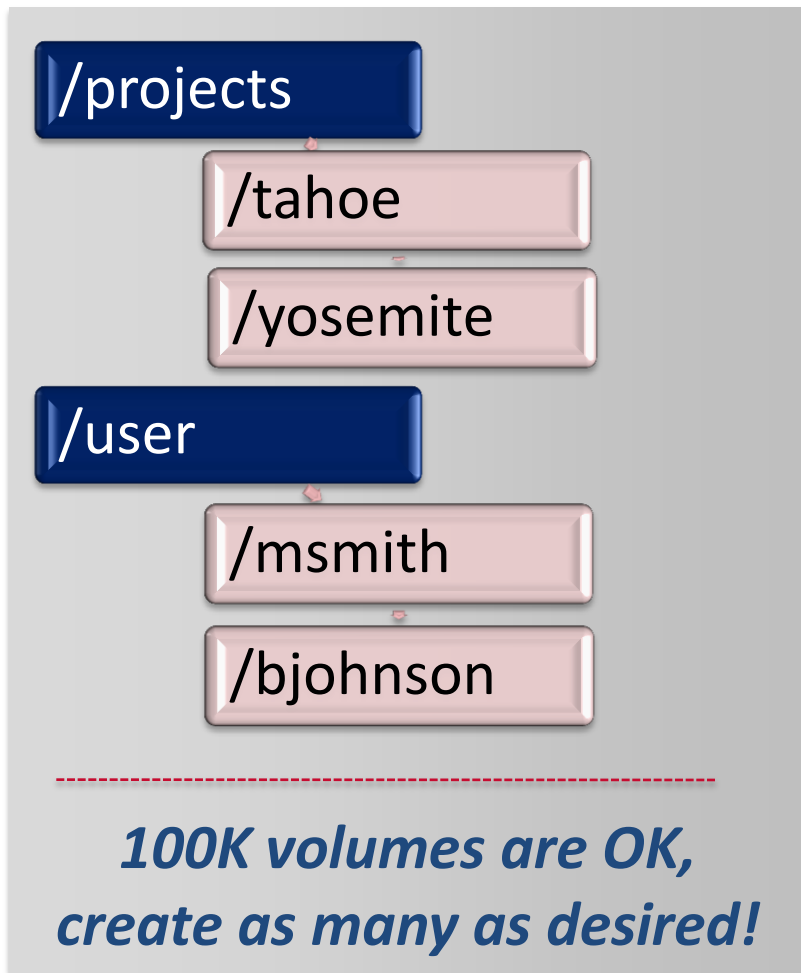
Tests: i. 16 streams x 120GB ii. 2000 streams x 1GB



Where & How Does MapR Exploit This Unique Advantage?



MapR Volumes & Snapshots

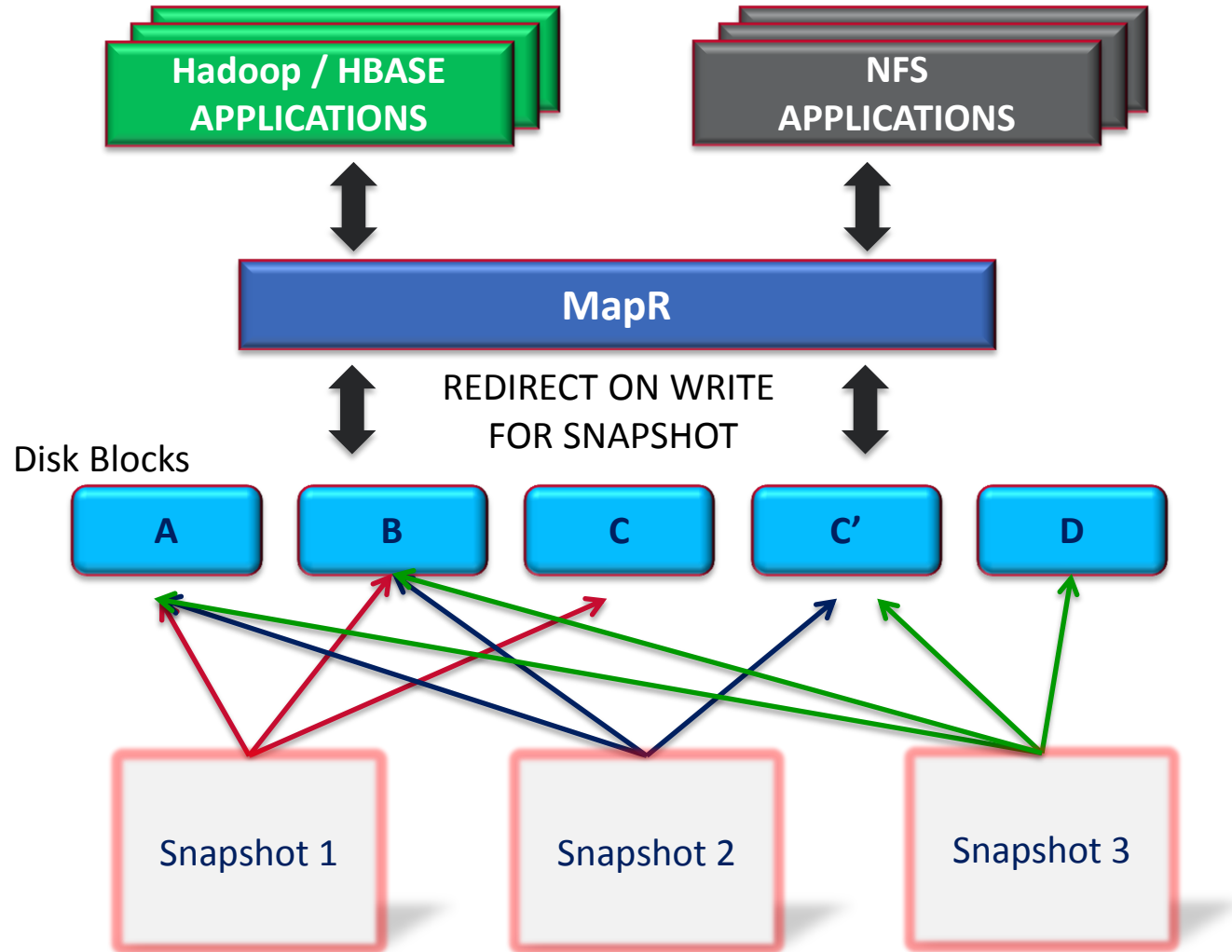


Volumes dramatically simplify data management

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



MapR Snapshots Technology



- Snapshots with automatic de-duplication
- Saves space by sharing blocks
- Lightning fast
- Zero performance loss on writing to original
- Scheduled, or on-demand
- Easy recovery with drag and drop



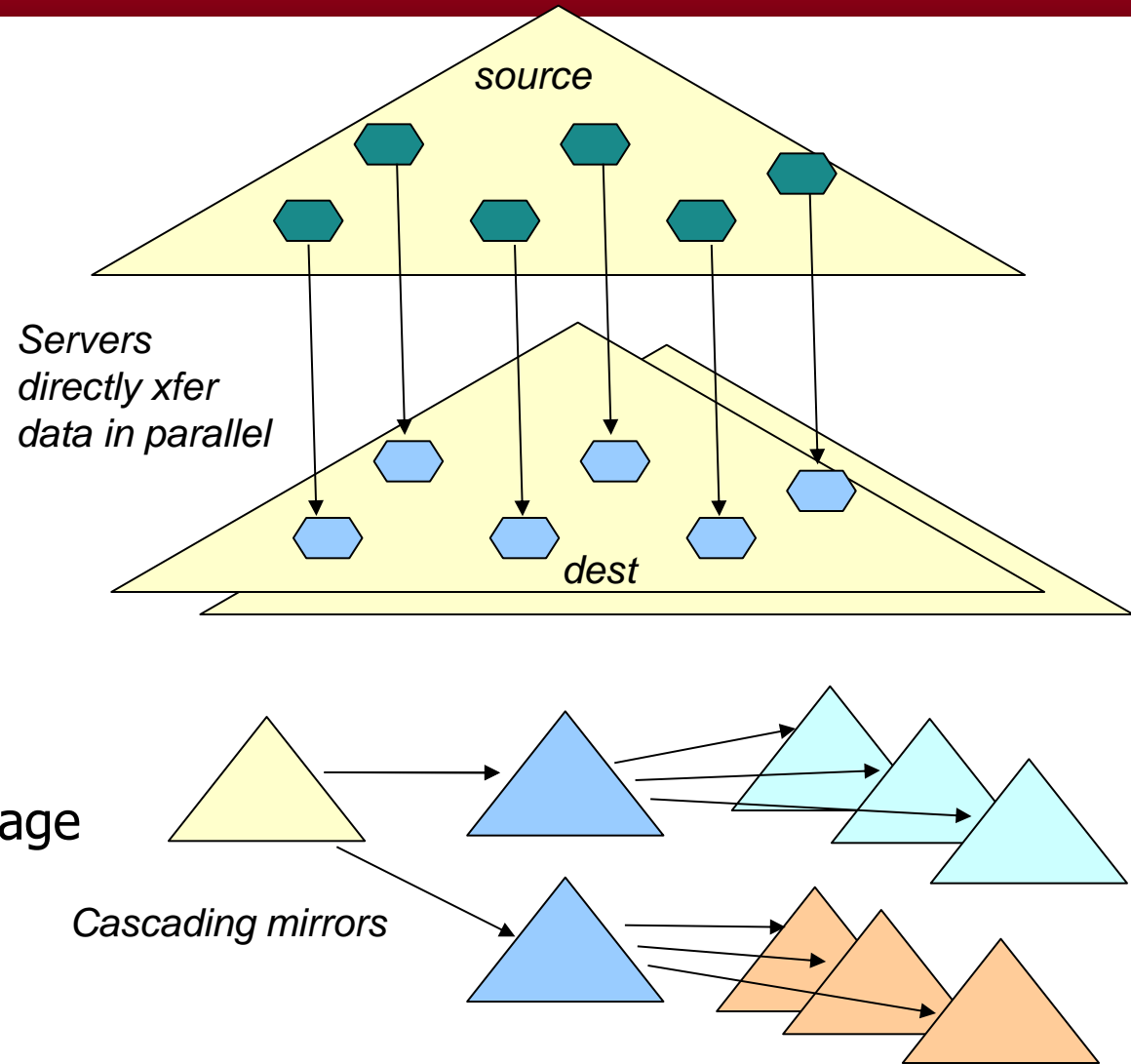
MapR Mirrors

Point-in-time physical copies of data

- Load balancing
- Cross-cluster backup
- Bulk data-transfer
- Scheduled/On-demand

Mirror is accessed automatically

- Path-names remain same
- Data transferred in parallel
- Atomic roll-forward of consistent image
- Old image left as snapshot



Multiple clusters

Clusters named using DNS convention

`ls /mapr/eng.mapr.com/home/srivas`

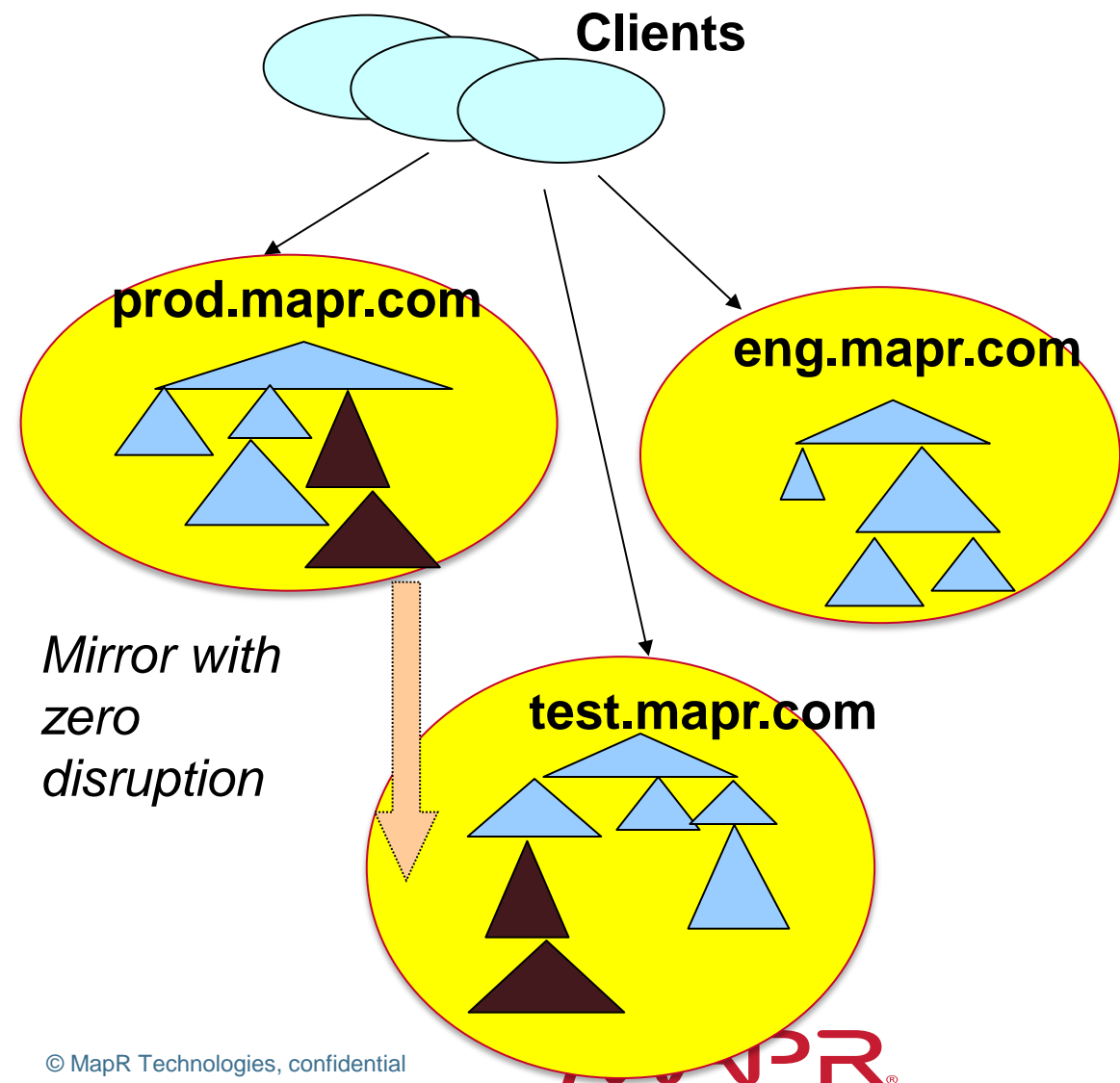
`vi /mapr/prod.mapr.com/tahoe/config.txt`

Add/remove cluster

- clients learn, no remounting needed

Mirror data across clusters

- relative path-names preserved even across volume boundaries



Where & How Does MapR Exploit This Unique Advantage?



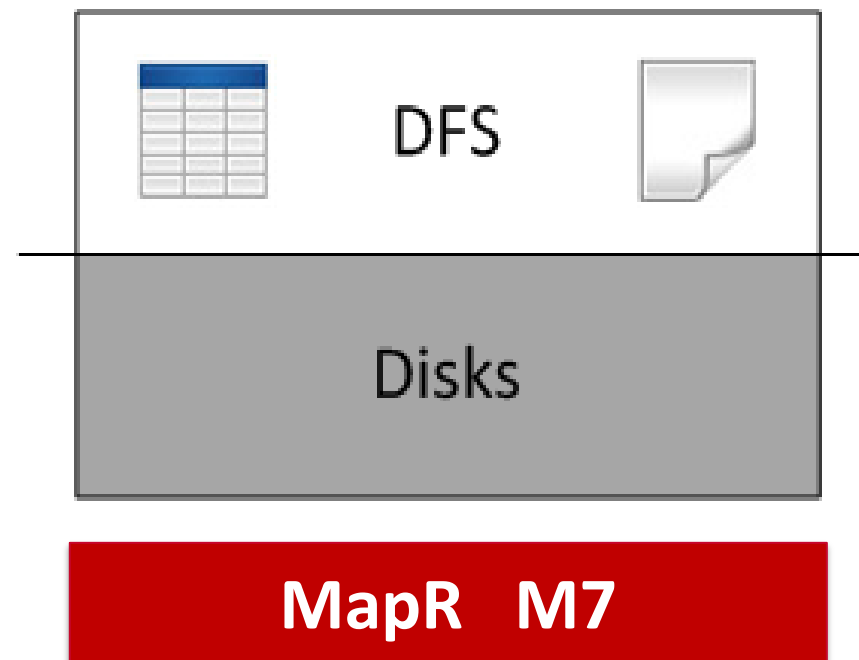
MapR M7 Tables

- 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



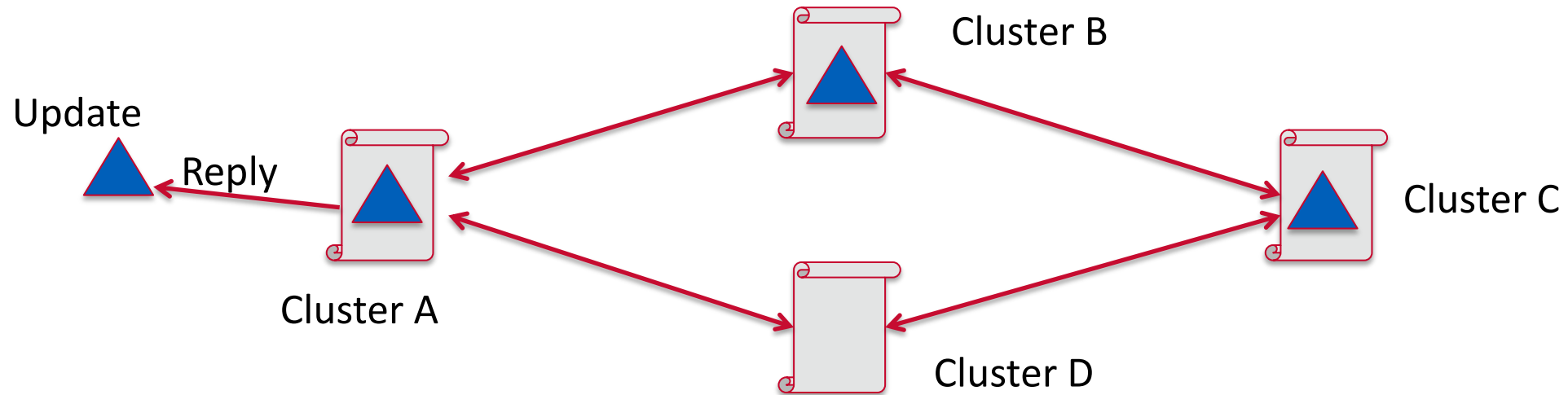
M7 Tables

- 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



M7 Tables Replication

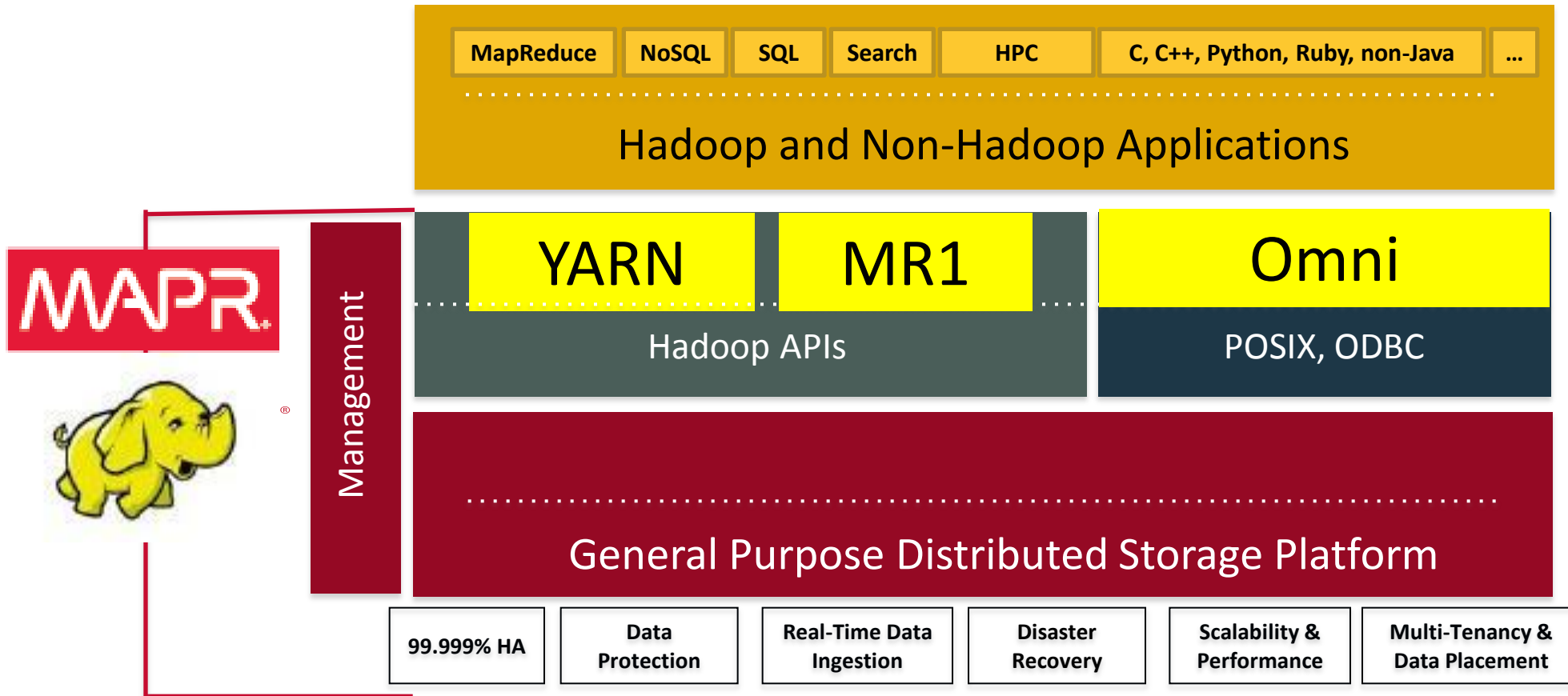
- Near instantaneous
 - Updates propagated to replica cluster(s) immediately, asynchronously
- Multi-site, WAN ready
- Multi-master
 - Can update at many clusters simultaneously



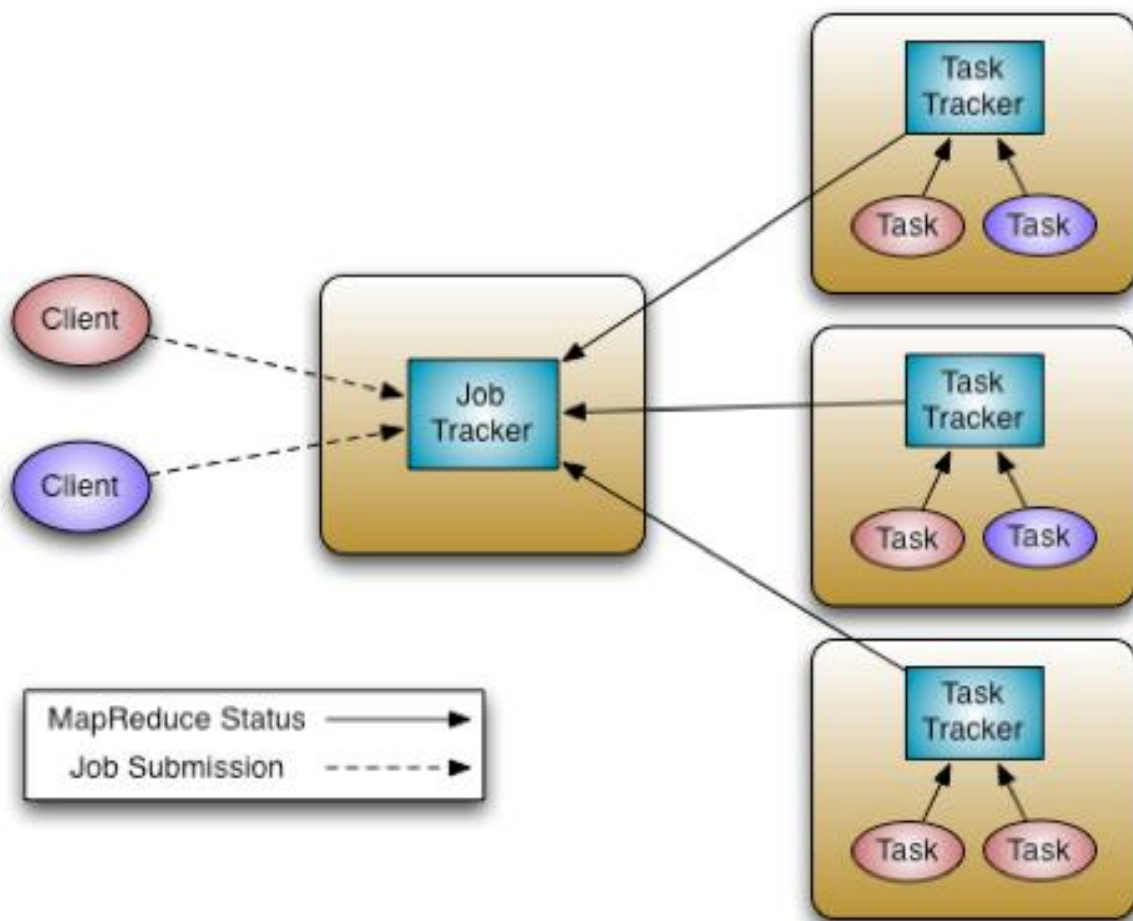
Where & How Does MapR Exploit This Unique Advantage?



MapR: The *Only* General Purpose Hadoop Distribution



MR1 – JobTracker and TaskTracker



TT sends periodic heart-beats to JT

- Slots available
- Task completions
- Keep alives

JT instructs TT on HB responses

- tasks to launch
- tasks to kill
- availability of inputs to reducers



MR1

Advantages

- Faster to schedule
- JVM re-use
- JT-TT traffic is optimized
- Failure management simple

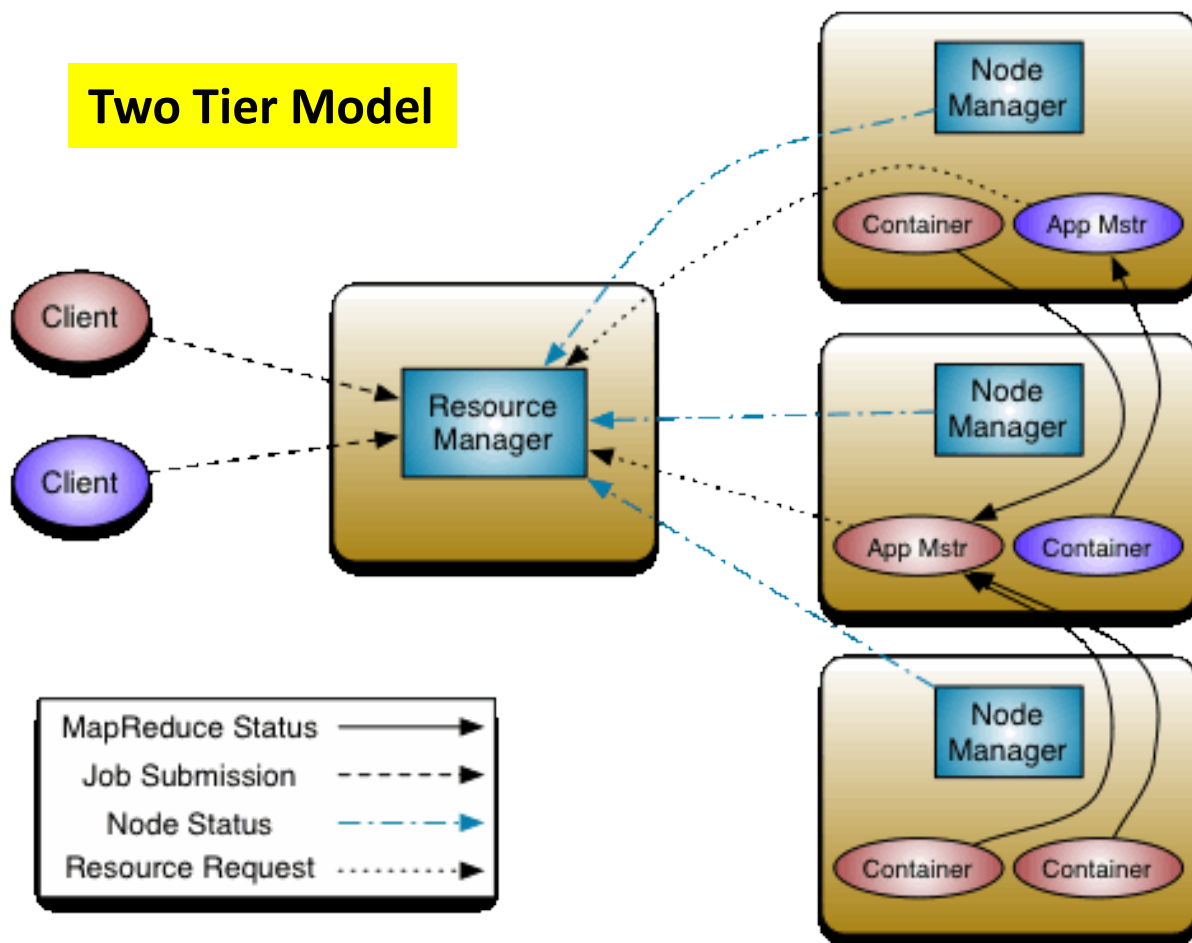
Disadvantages

- Static, fixed slots, difficult to adapt to different sized jobs
- Over-provisions resources
- Under-provisions resources
- Can only schedule map-reduce, and nothing else
- JT is bottleneck on large clusters



YARN and MR2

Two Tier Model



ResourceManager (RM) for whole cluster

- NodeManager (NM) is agent on each node
- Only manages CPU and RAM
- CPU/RAM allocated via *Yarn Containers*
 - By the NM on each node
 - Don't confuse with Linux containers

AppMaster for each “job”

- Launched inside a container
- Asks RM for more containers
 - MR2 for map and reduce slots
- Manages only 1 job
- RM/NM unaware of map-reduce



YARN

Advantages

- Can meet different jobs needs easily
- Does not over-provision
- More types of clustered services supported, instead of only map-reduce
- JT bottleneck solved “interestingly”



MapR Makes Hadoop Truly HA

- **ALL** Hadoop components have High Availability
 - e.g. YARN + MR1.0
- ResourceManager (RM) / JobTracker (JT) record their state in MapR
- On node-failure, RM or JT 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



YARN

Advantages

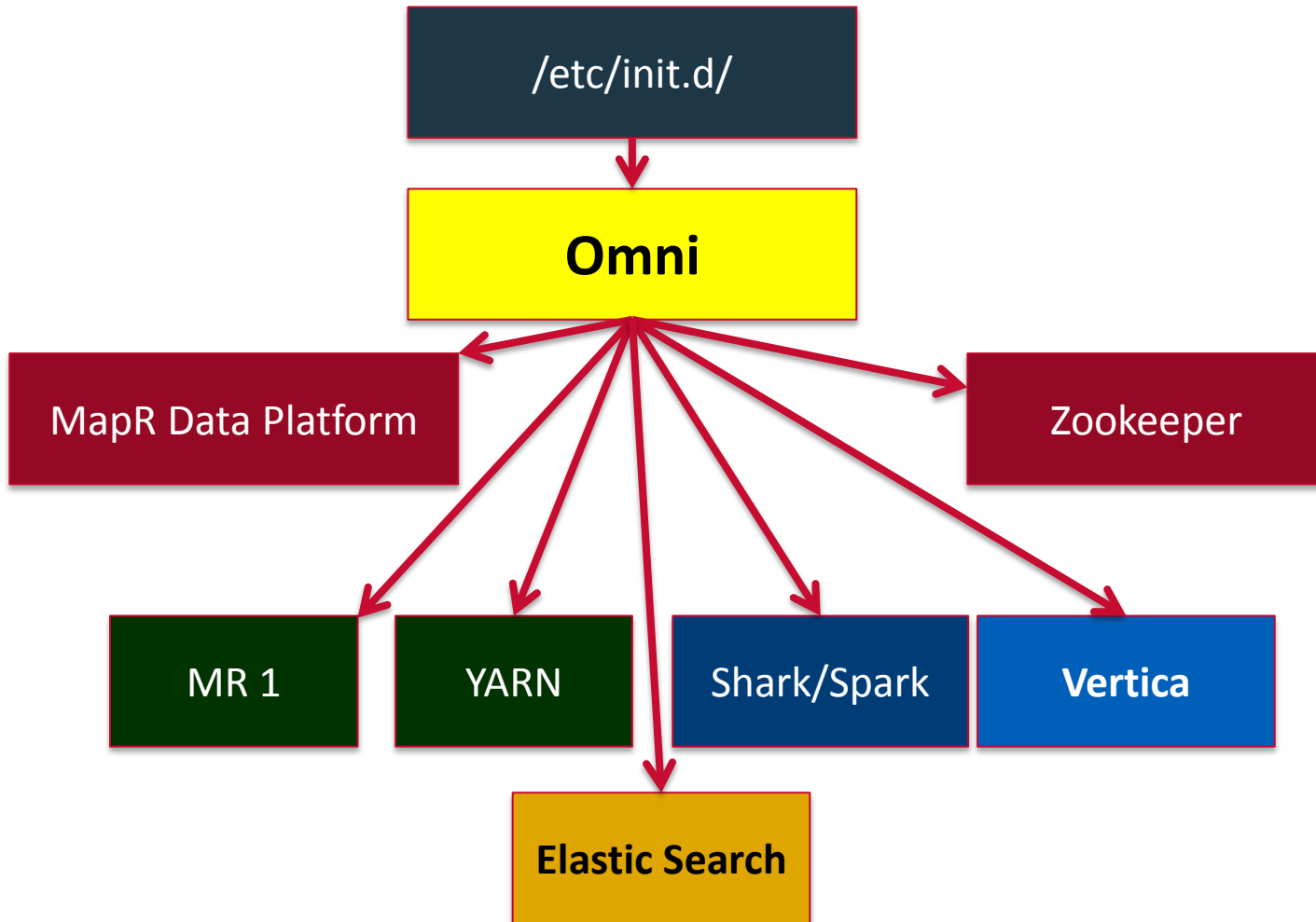
- Can meet different jobs needs easily
- Does not over-provision
- More types of clustered services supported, instead of only map-reduce
- JT bottleneck solved “interestingly”

Disadvantages

- Much slower to schedule
- No JVM re-use
- Apps have to code to YARN API to take advantage
- Failure recovery very complex, and not implemented in Apache
- Still no recourse for existing apps, cannot share cluster with Yarn



MapR's Omni



- Launched by **init**
- apportion CPU/RAM
- min, max, % system RAM
- Start, stop, restart, monitor
- Can do proxy failover
 - 1 of N
 - M of N



MapR Omni

- Configuration, not code
 - Declare service to Omni
 - Need HA or simple restart?
 - Sequencing
- Configure on needed nodes
 - use MapR's central config

```
/opt/mapr/conf/warden.vertica.conf
```

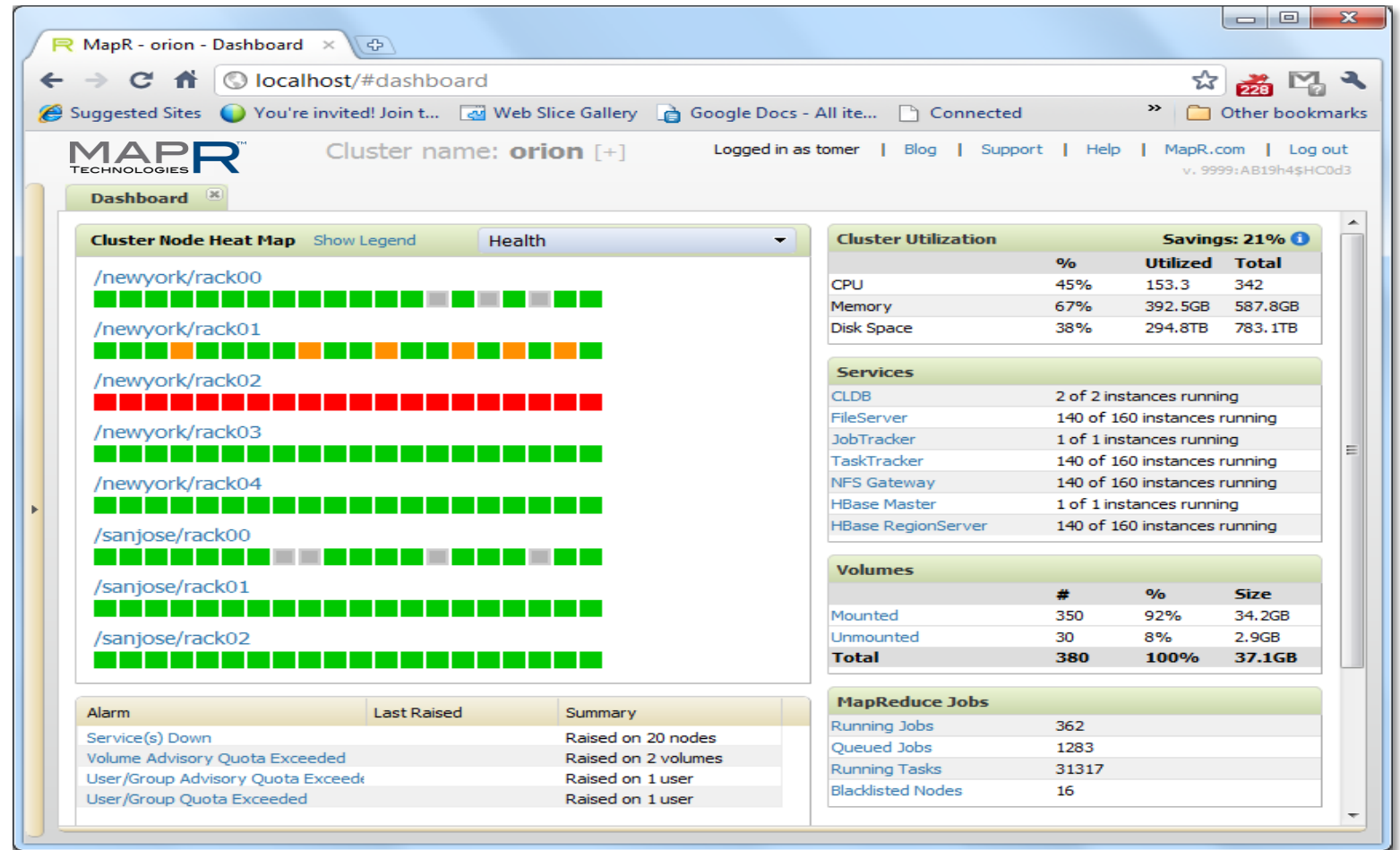
```
diplayname=vertica # on MapR's GUI
heap.min=1G
heap.max=10G
heap.percent=15
start=/opt/vertica/start-vertica
stop=/opt/vertica/stop-vertica
monitor=/opt/vertica/check-health
uri=...
HA=10 # of 20
```



Easy Management at Scale



- Complete visibility into your cluster
- Monitoring
- Provisioning
- Detecting Trends



MapR Does MapReduce (Fast!)

MAPR.



TeraSort Record

1 TB in 54 seconds

1003 nodes

MinuteSort Record

1.5 TB in 59 seconds

2103 nodes



MapR Does MapReduce (Faster!)

MAPR.



TeraSort Record

1 TB in 54 seconds

1003 nodes

MinuteSort Record

1.65 ~~1.5~~ TB in 59 seconds
300 ~~2103~~ nodes



Summary

MapR is the only general purpose framework for Hadoop

- MR1 + YARN + Omni: general purpose ***processing*** framework
- MapR Data Platform: general purpose ***storage*** system
- MapR Hadoop: the ***only*** distribution that combines both
- **Only from MapR: Hadoop and non-Hadoop Apps on same data**



What SQL-on-Hadoop do you like?

presto

SQL Query Engine

shark

SQL based analytics



Real-time
interactive queries

Impala

Real-time
interactive queries

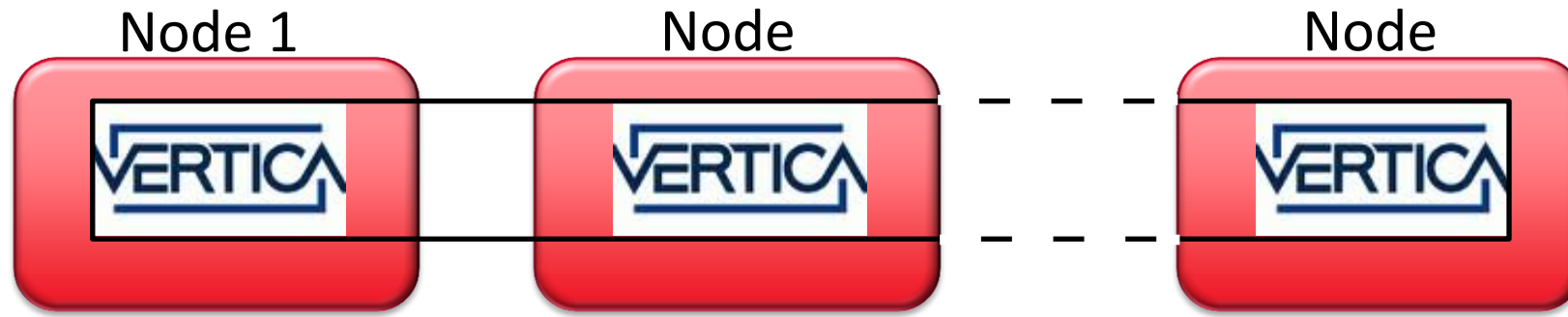


SQL conversion
to MapReduce





What is HP Vertica?



- MPP, columnar store, database
- Data sharded throughout cluster
- Provides ability to concurrently load and query data
- Very fast SQL query engine
- Famously used during Obama's 2012 re-election campaign



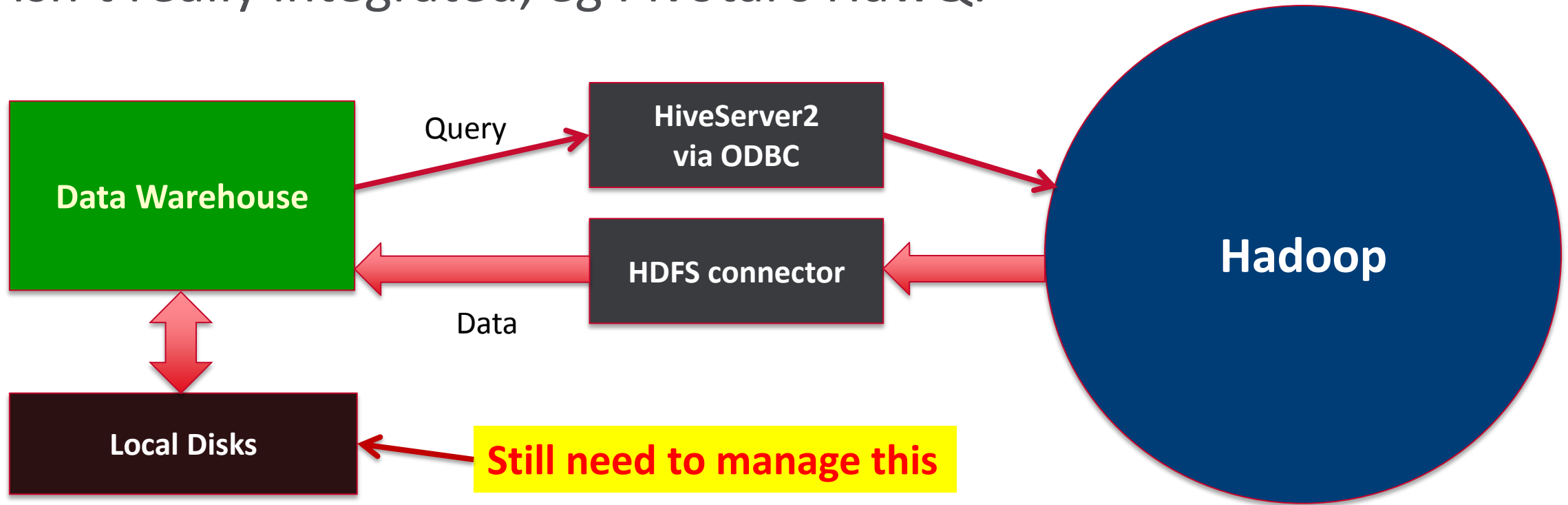
SQL on Hadoop: 2 approaches

	Retrofit Traditional DB Pivotal HAWQ, MS PolyBase, IBM BigSQL	Hadoop SQL Impala, Drill, Presto, Hive on Tez, Shark
<i>Technology approach</i>	<p>Traditional MPP engines using Hadoop as storage</p> <p>Must load data in proprietary file formats and use proprietary metadata stores</p>	<p>Purpose built query engines using Hadoop for both storage & processing</p> <p>Uses Hadoop open file formats and Hadoop metadata (Hive/Hcatalog)</p>
<i>Benefits</i>	<p>More complete SQL</p> <p>Maturity</p>	<p>Cost effectiveness</p> <p>Scalability</p> <p>Reuse</p>



Typical Database-on-Hadoop today

- Isn't really integrated, eg Pivotal's HawQ:



Query planning has no idea about data locality or distribution on Hadoop



Local Disk versus NAS/SAN



+ve: good performance
-ve: unmanageable



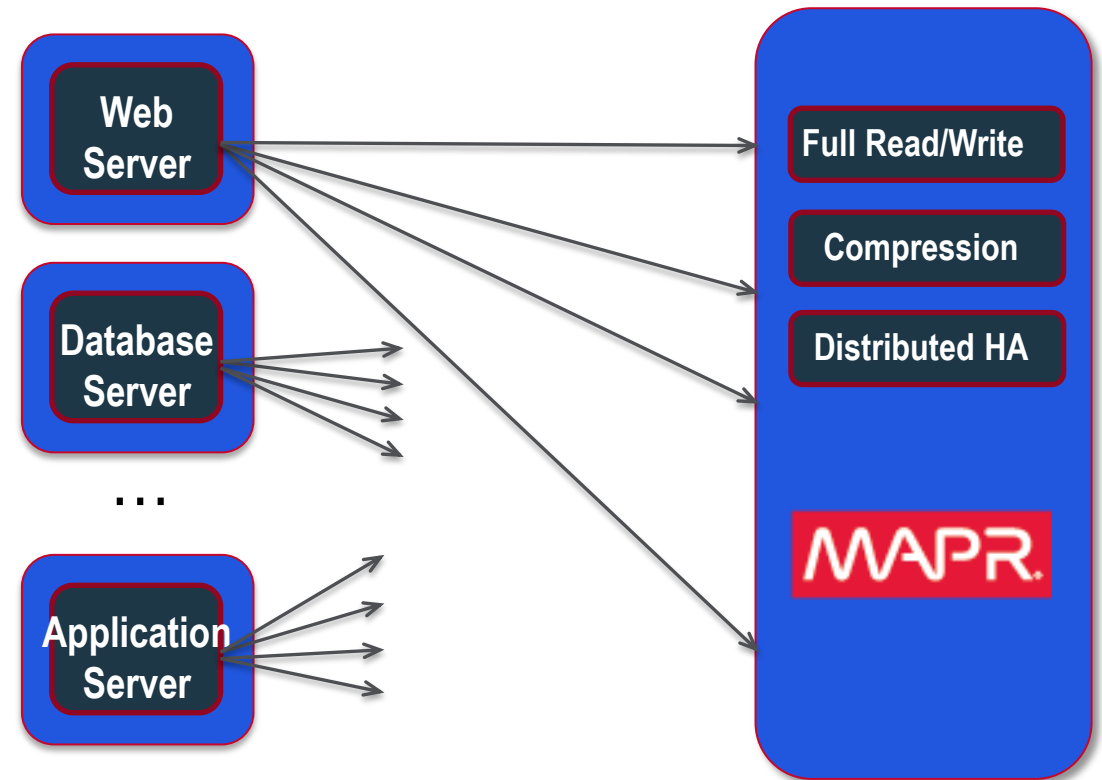
Very manageable:

- Expandable storage
- Less wasted disk space
- Repair/replace failed components
- Upgrade seamlessly without downtime
- Provision for performance
- Backup and Disaster-Recovery



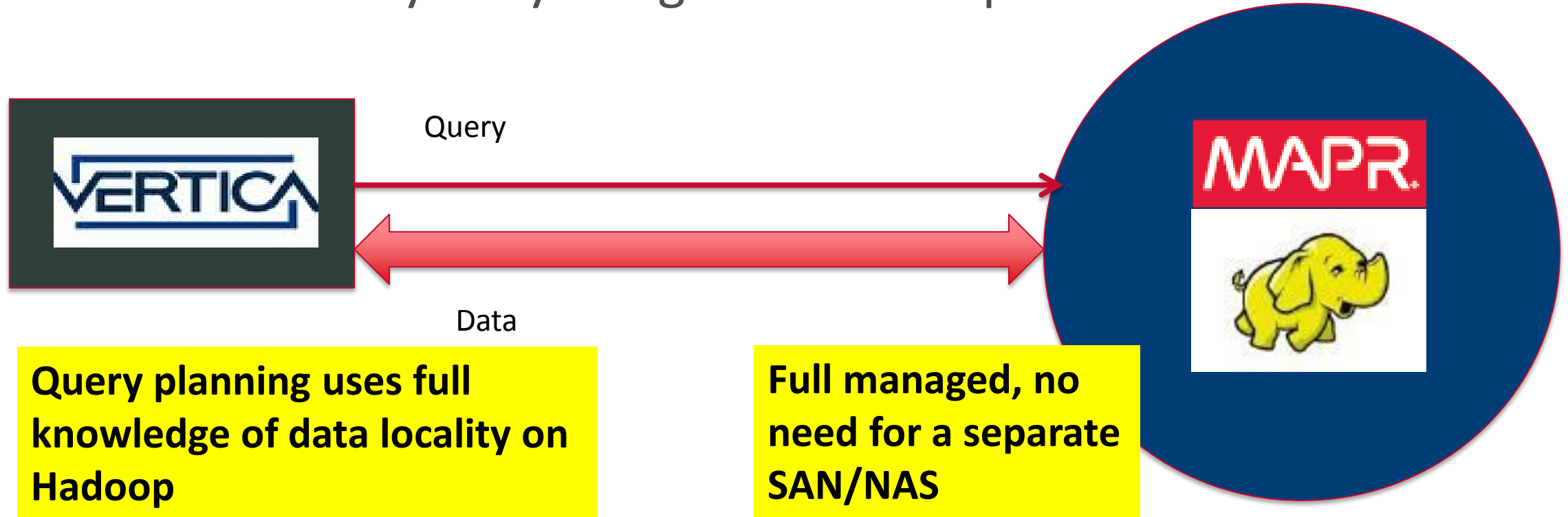
MapR: Local Disk with NAS capabilities

- Only MapR can provide **local-disk performance** with full NAS capabilities
 - Dynamic space management
 - Provision for performance
 - Tiering
 - Instantaneous, consistent backups
 - Automatic recovery from disk errors
 - Complete DR solution built-in



VERTICA on MapR

- Vertica is the only truly integrated Hadoop database



Broadest Analytics Capabilities

- Broadest set of analytics on one platform
 - Rich set of built-in analytic functions from HP Vertica
 - Add to it Hadoop
- HP Vertica on MapR is certified with a complete set of BI tools and designed for machine-generated SQL
- Fast schema-less data exploration with HP Vertica Flex Zone



Apache Open Source Community Projects

		Q1	Q2	Q3	Q4
Resource management	YARN 2.2	◆	◆	◆	◆
Batch	MapReduce	◆	◆	◆	◆
	Hive 0.12	◆	◆	◆	◆
	Pig 0.11		◆		◆
	Cascading 2.5	◆	◆	◆	◆
	Spark 0.8		◆	◆	◆
Interactive SQL	Drill 1.0		◆	◆	◆
	Shark 0.8		◆	◆	◆
	Impala 1.1	◆	◆	◆	◆
	Hive on Tez 0.13			◆	◆
Data integration	Flume 1.4.0	◆	◆	◆	◆
	Sqoop 1.4.4	◆		◆	
	HttpFS 1.0		◆		◆
Machine learning	Mahout 0.8	◆		◆	
	MLlib/MLBase 0.8.1		◆		◆
Coordination	Oozie 3.3.2	◆	◆	◆	◆
	ZooKeeper	◆	◆	◆	◆
Streaming	Storm 0.9.0		◆	◆	◆
	Spark Streaming 0.8		◆	◆	◆
Data management	Falcon 0.3			◆	◆
	Knox 0.3			◆	◆
	Sentry 1.2.0			◆	◆
GUI and provisioning	Hue 2.5		◆	◆	◆
	Savannah 0.4		◆	◆	◆
NoSQL and search	HBase 0.94	◆	◆	◆	◆
	Solr (LWS 2.6.1)	◆	◆	◆	◆

- Complete distribution with aggressive roadmap

➤ 14 projects in the distribution

➤ 10+ new projects expected in 2014

• Monthly update to projects

• Run multiple versions simultaneously

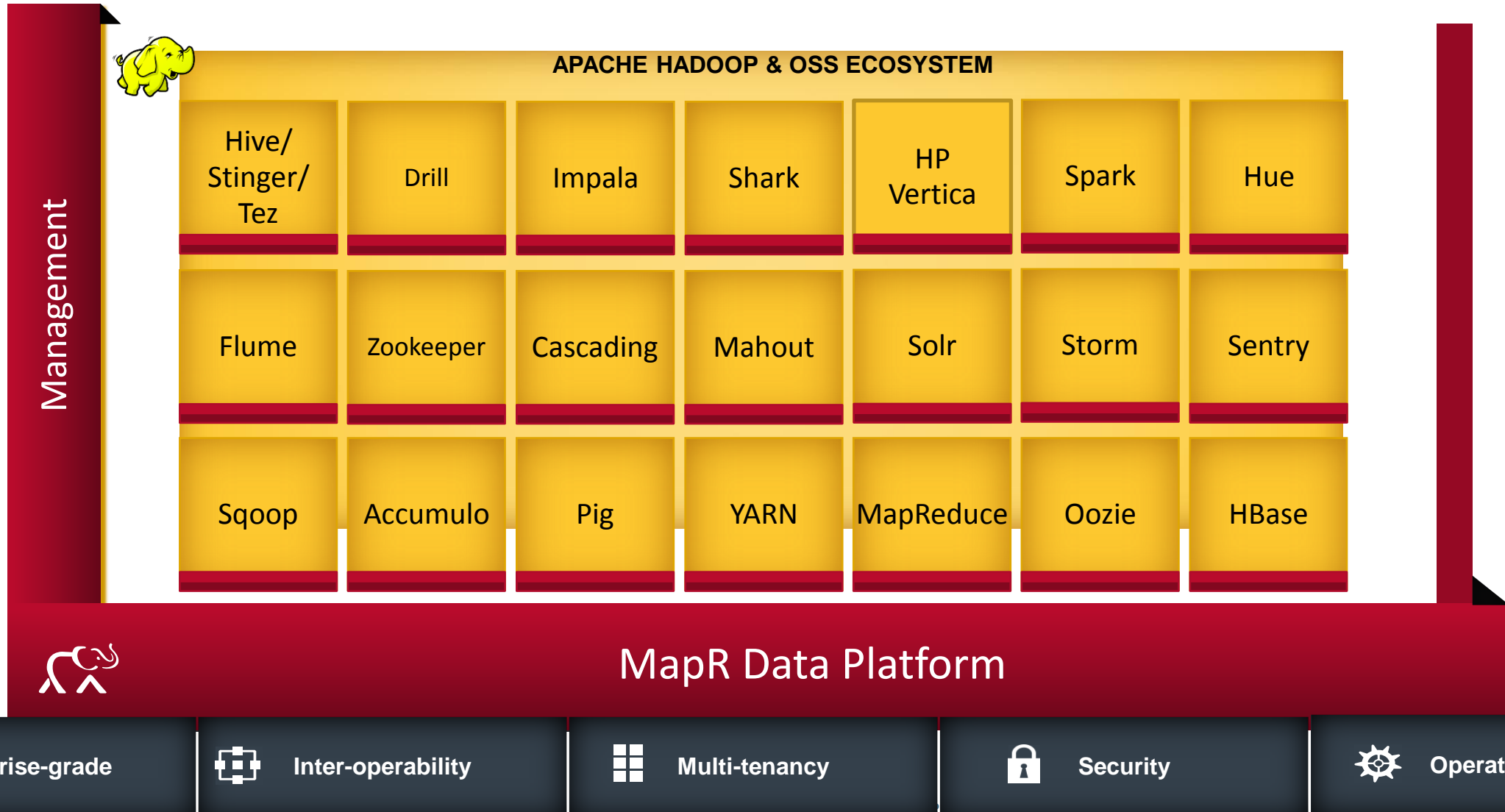
◆ Update

◆ New

MAPR®

© MapR Technologies, confidential

Enterprise Hadoop



Thank you!

(we are hiring ...)



M. C. Srivas
srivas@mapr.com

