TREND MICRO

Securing Your Web World



Cutting Edge Hadoop Technology and Trends

Andrew Purtell
Trend Hadoop Group

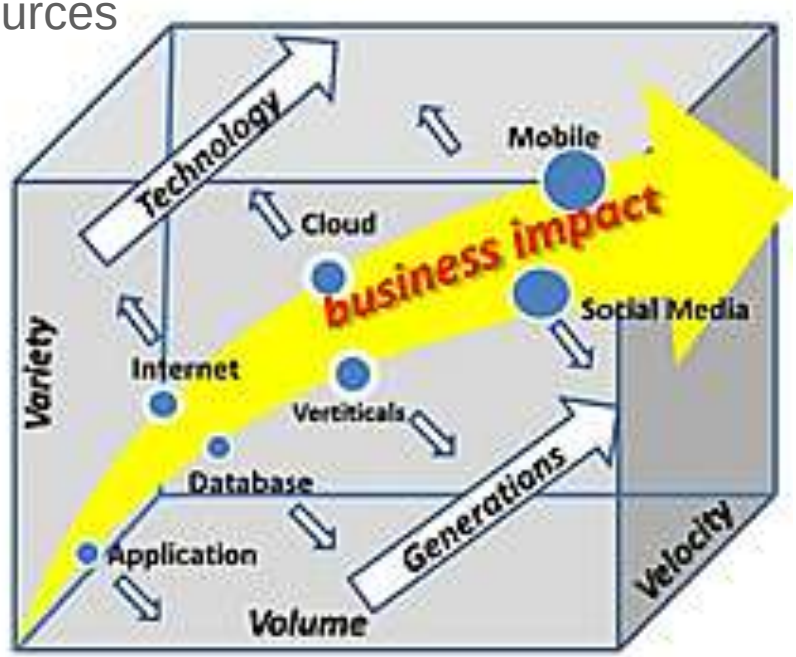
Who Am I?

- Engineer
- Researcher (Systems)
- Software Architect
- Hadoop and HBase user since 2008
  - (Hadoop 0.17, HBase 0.1)
- Committer and PMC member, Apache HBase, since 2009
- Manager, Trend Hadoop Group

Hadoop And “Big Data”

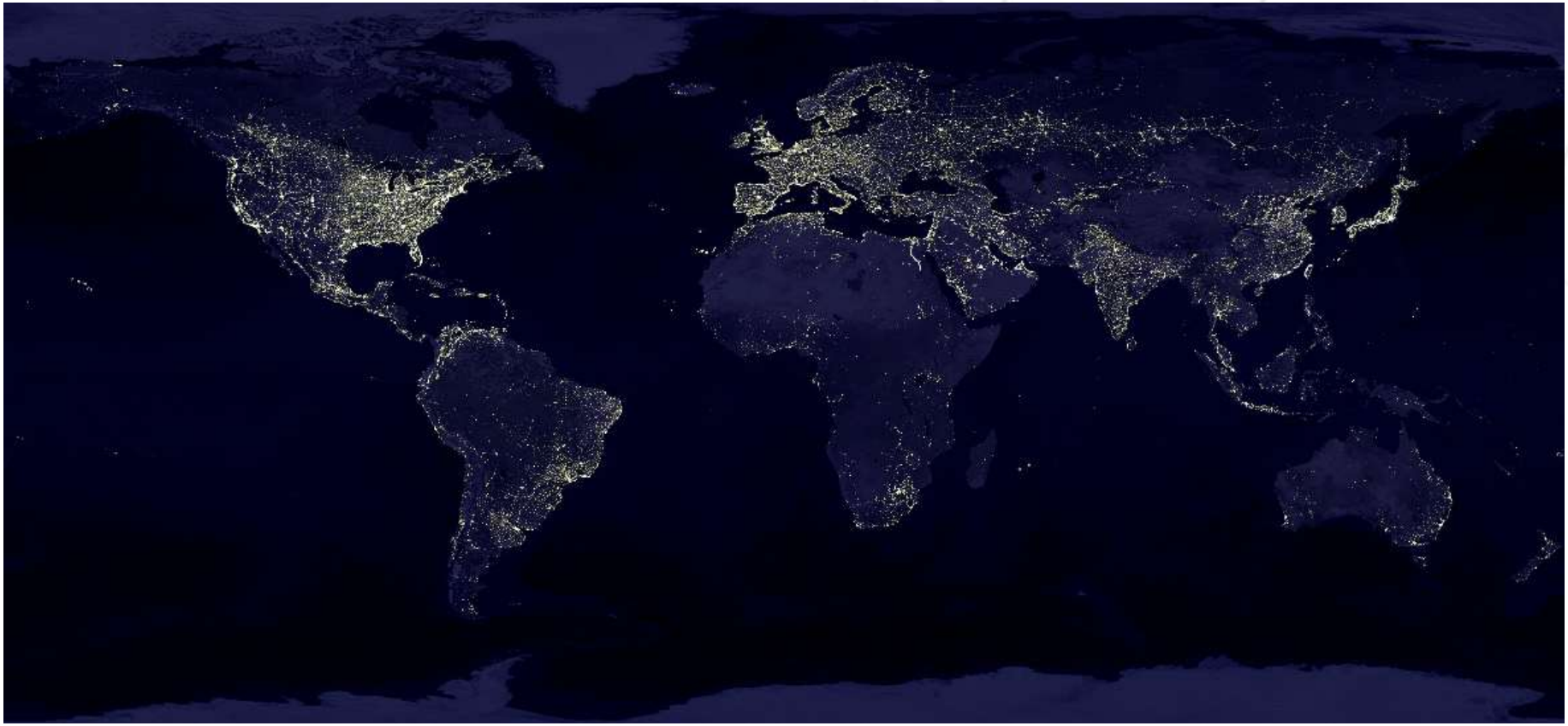
“Big Data”

- Organizations increasingly have a data problem
  - Datasets that are awkward to work with, beyond the ability of commonly used software tools to capture, manage, and process the data within a tolerable elapsed time
  - VVV model: Volume, Variety, Velocity
    - Volume: Amount of data (scale)
    - Variety: Range of data types and sources
    - Velocity: Speed of data in/out



- Driven by
  - Smartphone adoption (mobile)
  - Social media
  - Cloud and the Internet of Things
  - Business impact

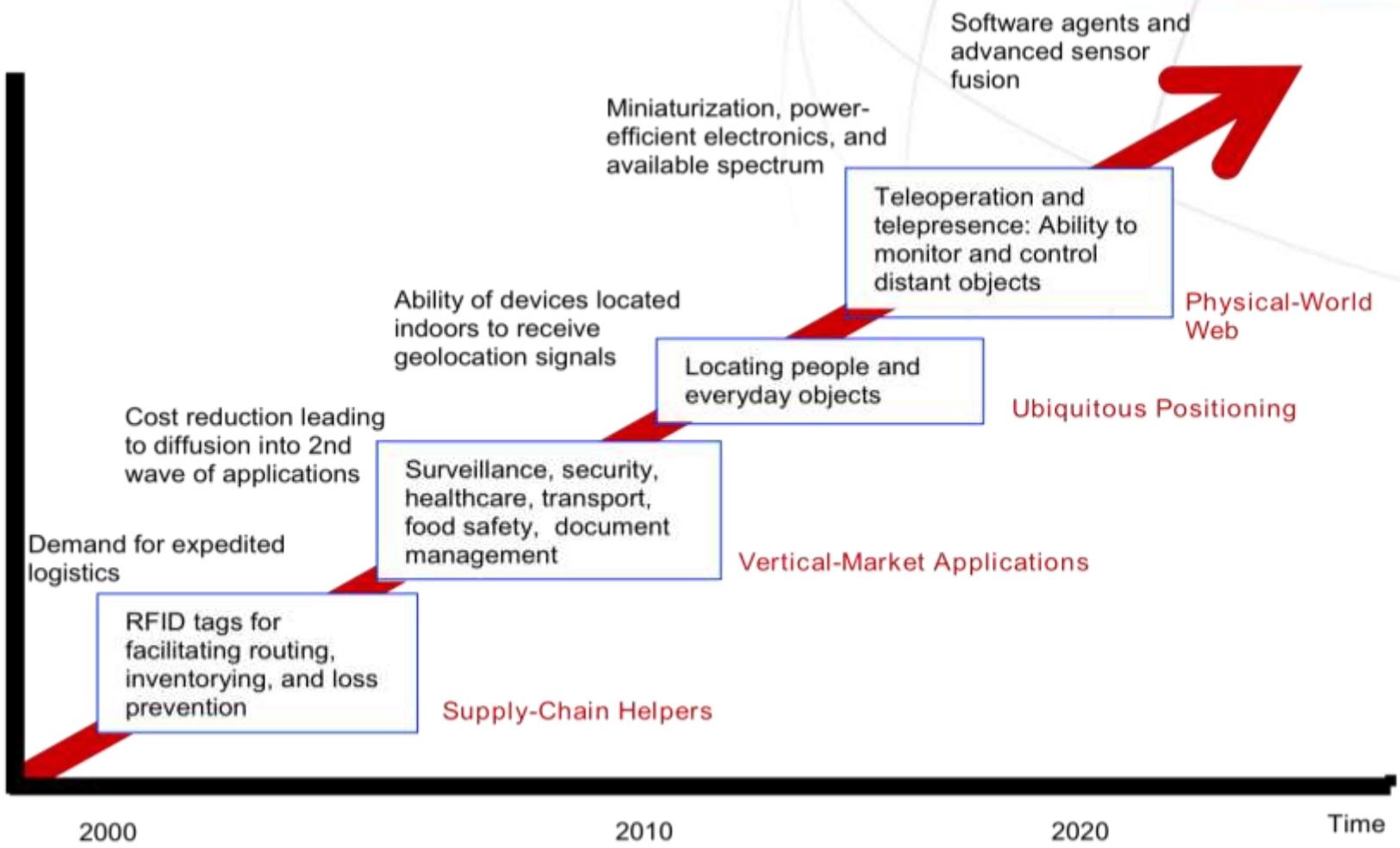
The Scale of “Big Data”



Total number of humans (potential information sources): **7.039 billion**
Percentage of humans who own a mobile phone: **80%**
Out of the **5 billion mobile phones** in the world, **1.08 billion** are smart phones

From geographic.org/maps/new2/city_lights_space_world_map.html

The Scale of "Big Data"



The Internet of Things may come to encode as many as **50 to 100 trillion** objects, and collectively we will want to know something about all of them

From http://en.wikipedia.org/wiki/Internet_of_Things/

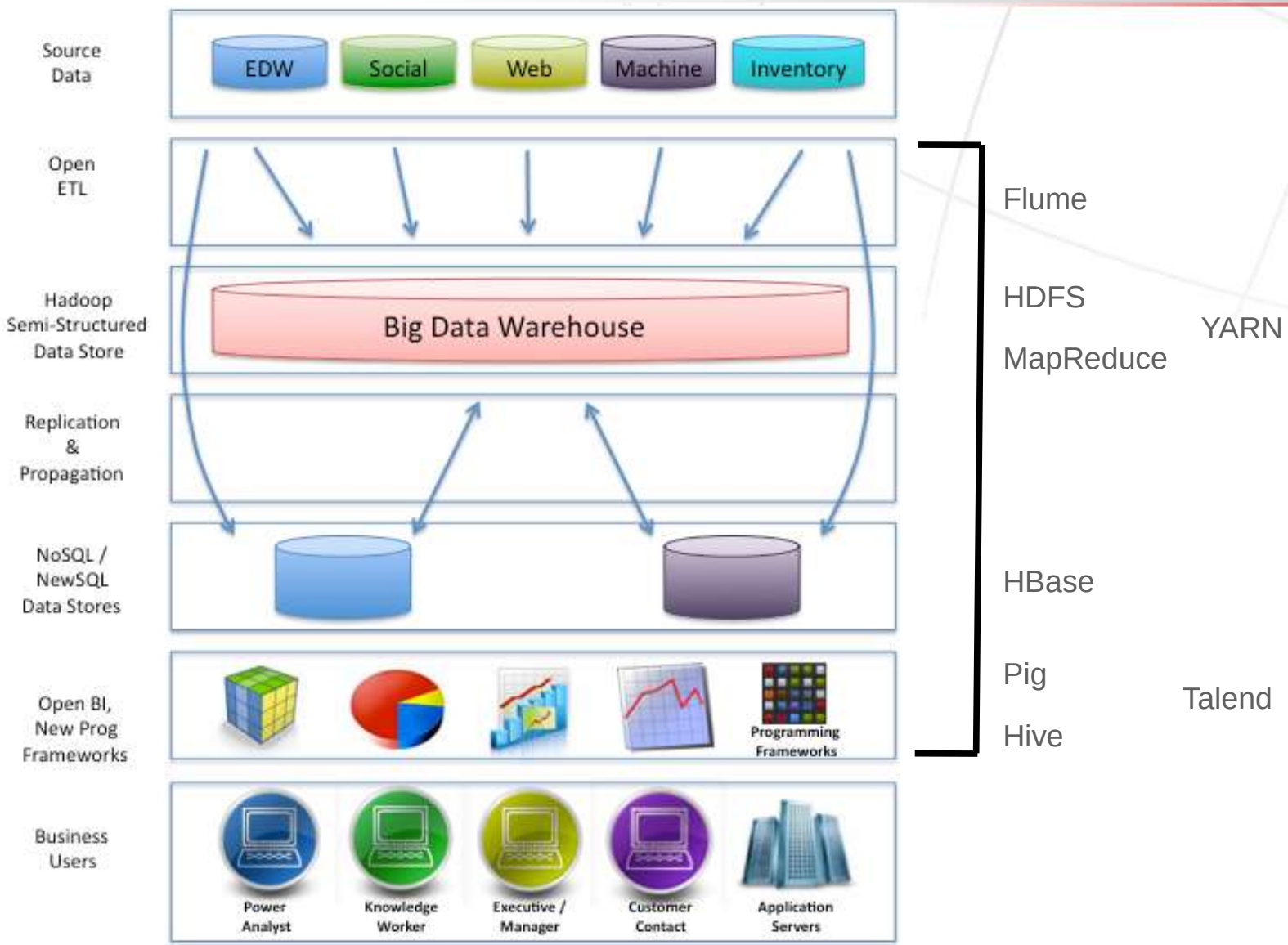
Hadoop, A Commodity Platform for Big Data

- A commodity software framework that combines with commodity hardware for affordable and manageable raw storage and compute horsepower
- Why is Hadoop widely adopted for handling Big Data?
  - Commodity is the only route to affordability for scale problems
  - Designed to solve the fast (parallel), reliable analysis of both structured data and complex data at the same time
  - Self-healing: Hadoop delivers data — and can run large-scale, high-performance processing jobs — in spite of system changes or failures
  - Large and growing ecosystem of tools
    - An integrated ecosystem for Big Data

Volume →
Velocity →
Variety →

  - HDFS: Distributed reliable scalable storage
  - HBase: Massively scalable database
  - MapReduce: Distributed programming framework
  - Pig: Powerful data flow programming
  - Hive: SQL-like analysis

Hadoop, A Data Warehousing Ecosystem



Adapted from http://blog.infochimps.com/2012/08/28/the-data-era-moving/

Finding The Hadoop Cutting Edge

Where Is The Hadoop Cutting Edge?

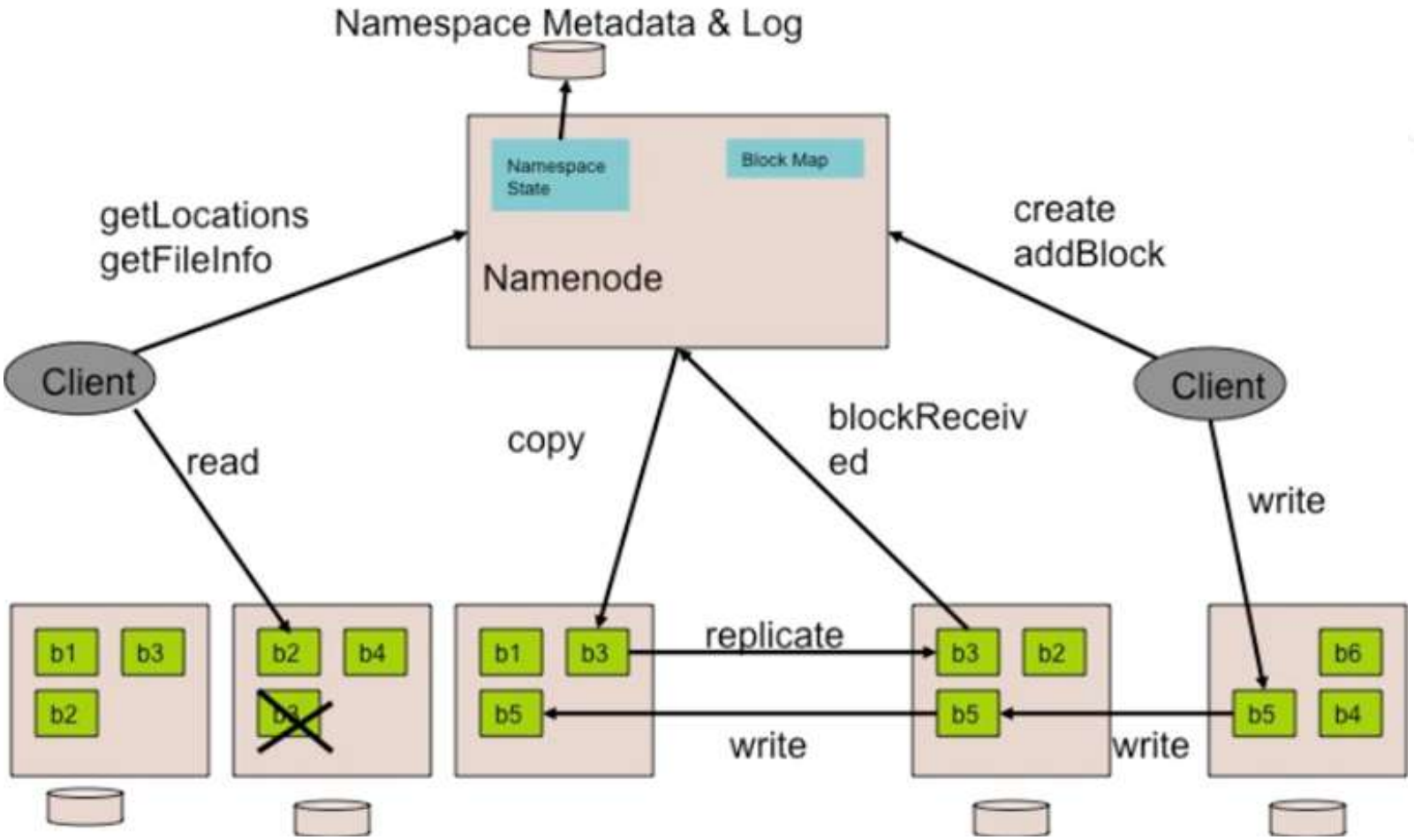
- As the Hadoop ecosystem has seen widespread adoption, and new users test its limits, some shortcomings are apparent
  - Where high availability is crucial
  - When the MapReduce computational model is a poor fit
  - When decisions must be made quickly
  - When a SQL-like interface (HQL) sets the wrong user expectations
  - Issues presented by the JVM
  - Issues presented by the increasing density and affordability of solid state storage
  - Where GPU computation offload is desired

- We find the cutting edge and trends where solutions for these problems are needed

Where High Availability Is Crucial

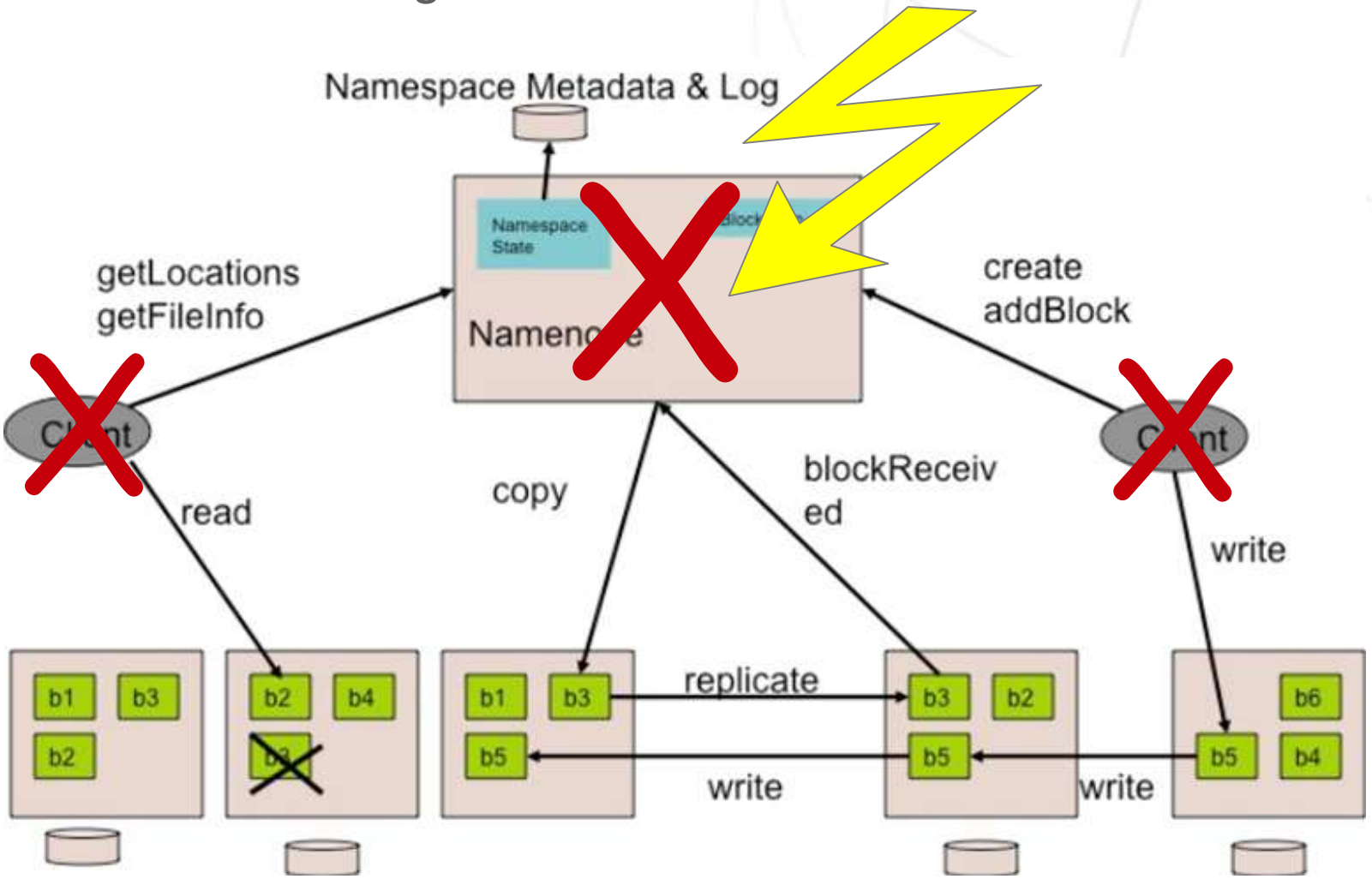
Where High Availability Is Crucial

- Problem: The singleton NameNode



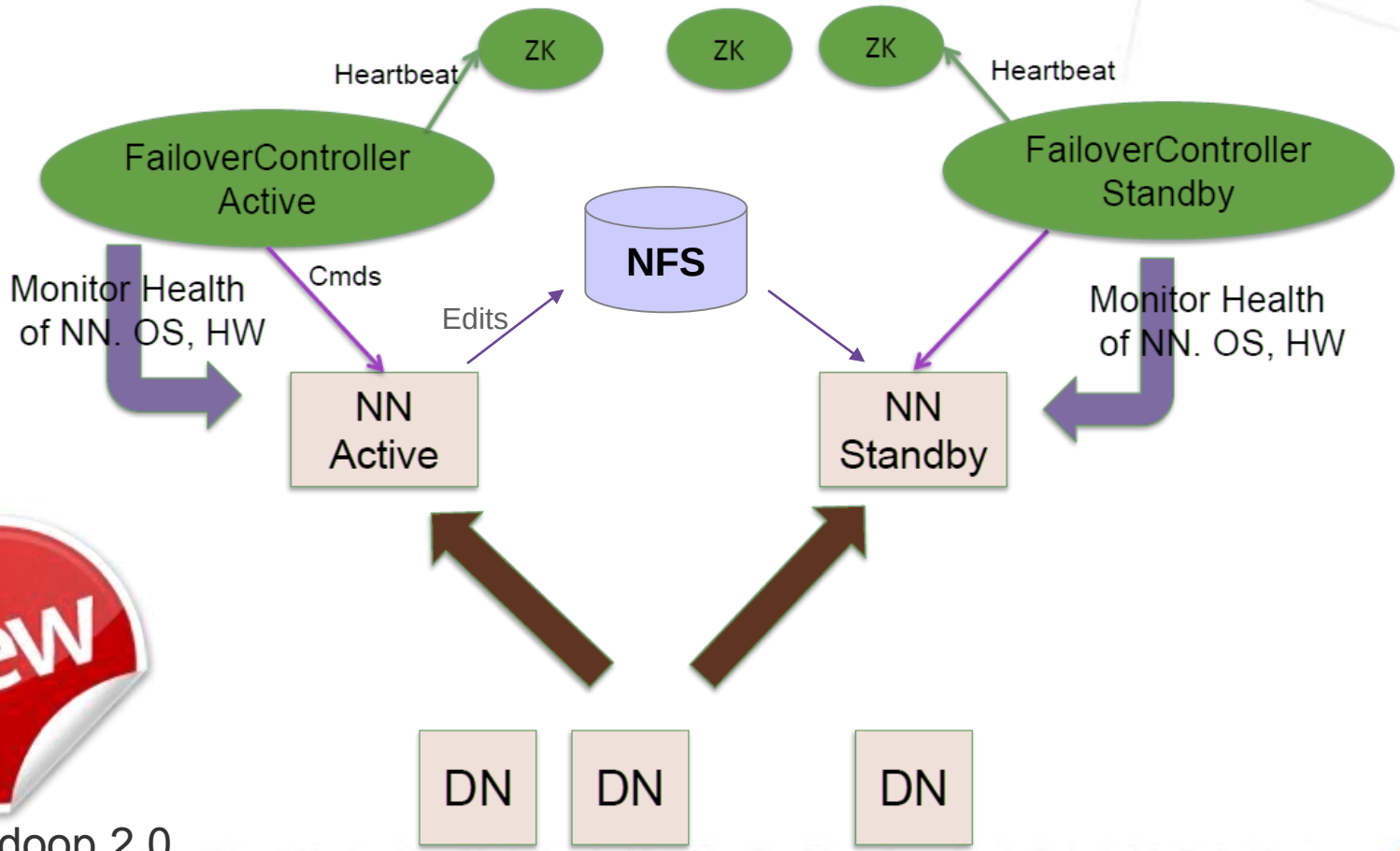
Where High Availability Is Crucial

- Problem: The singleton NameNode



Highly Available HDFS

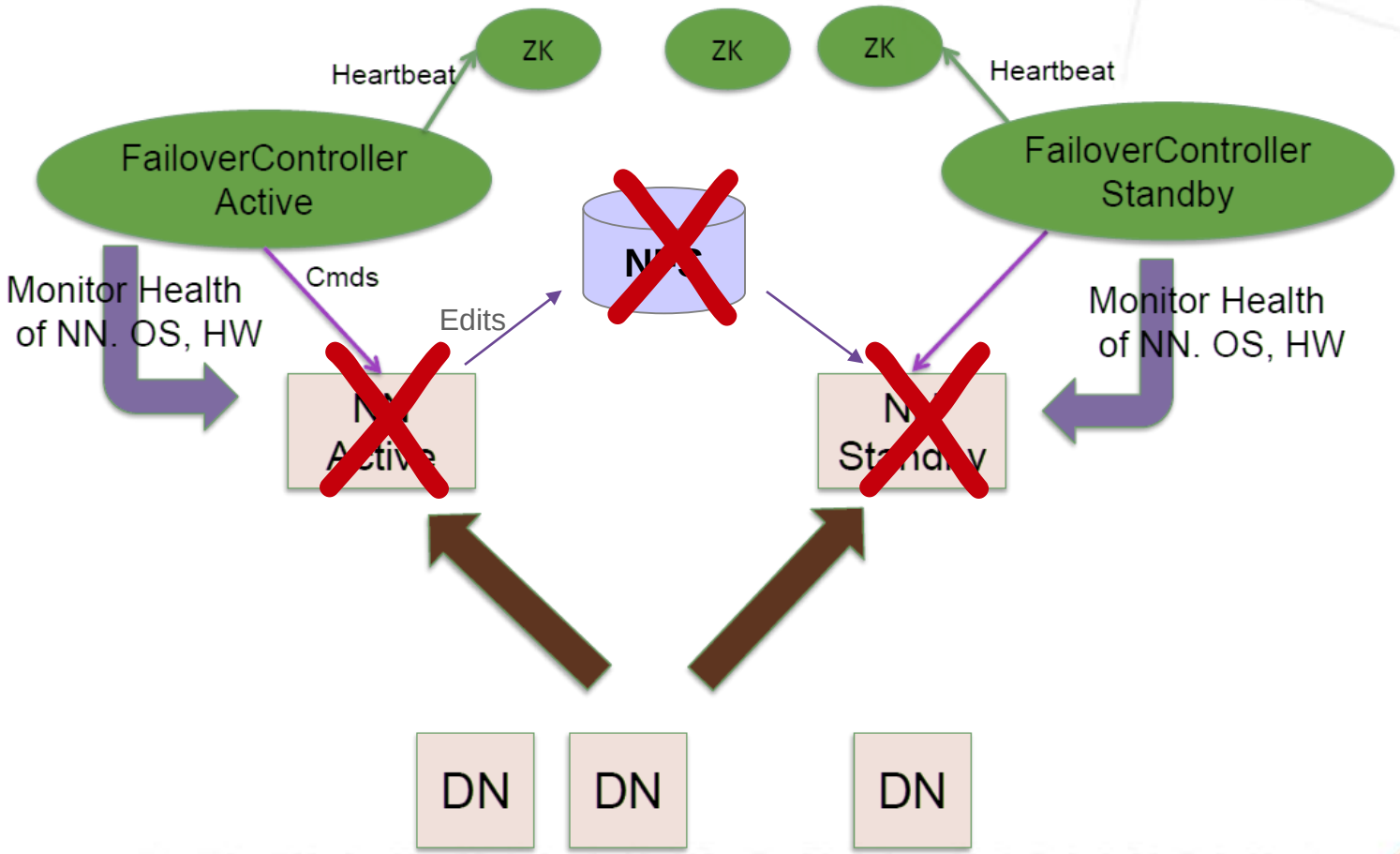
- Deploy an active and standby NameNode with manual or automatic failover
  - Failover control is a new component: Fencing, promotion



… in Hadoop 2.0

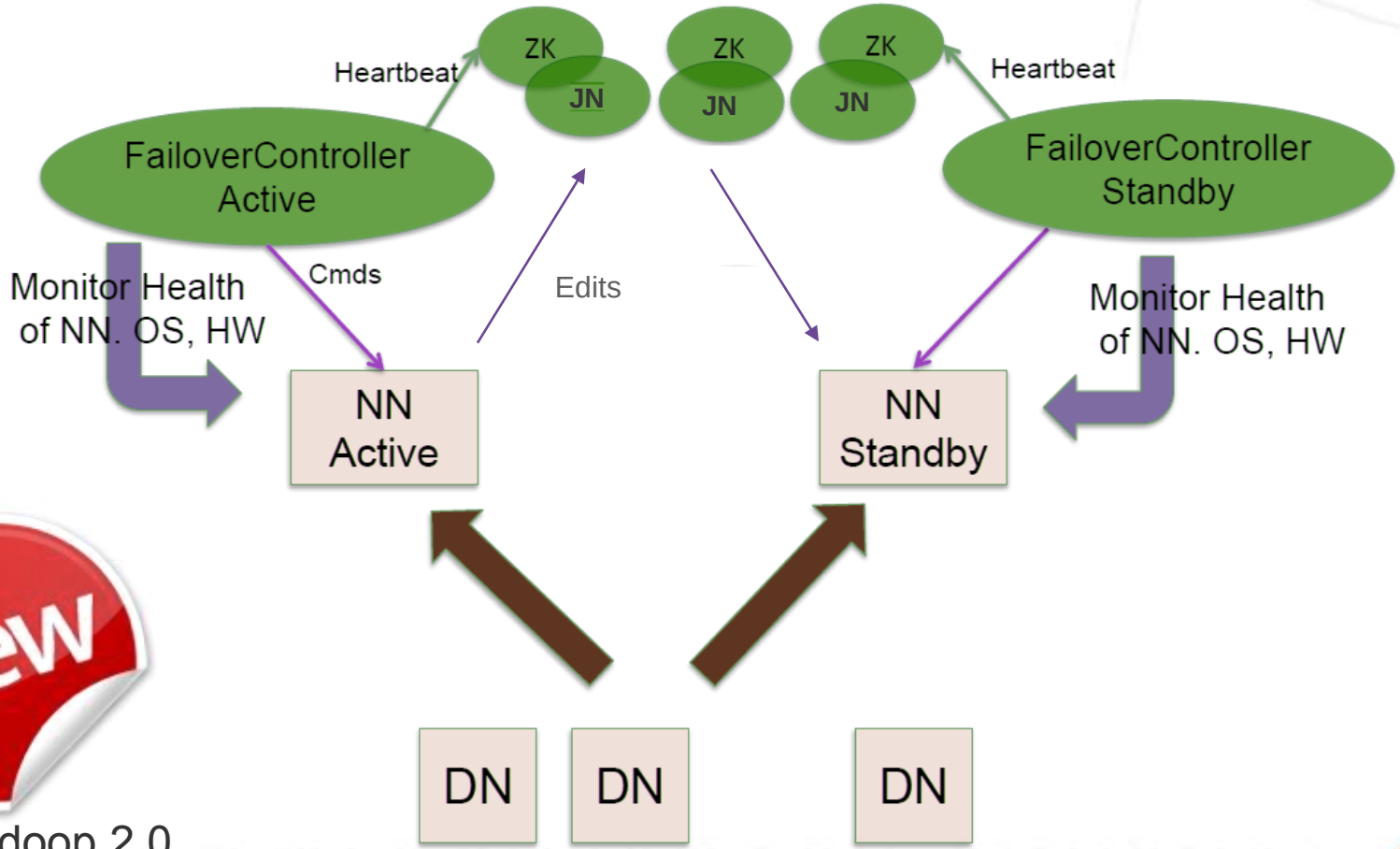
Highly Available HDFS

- But... the current solution requires NFS for sharing the edit log
  - Doesn't this move the SPOF from the NameNode to a NFS filer?



Highly Available HDFS

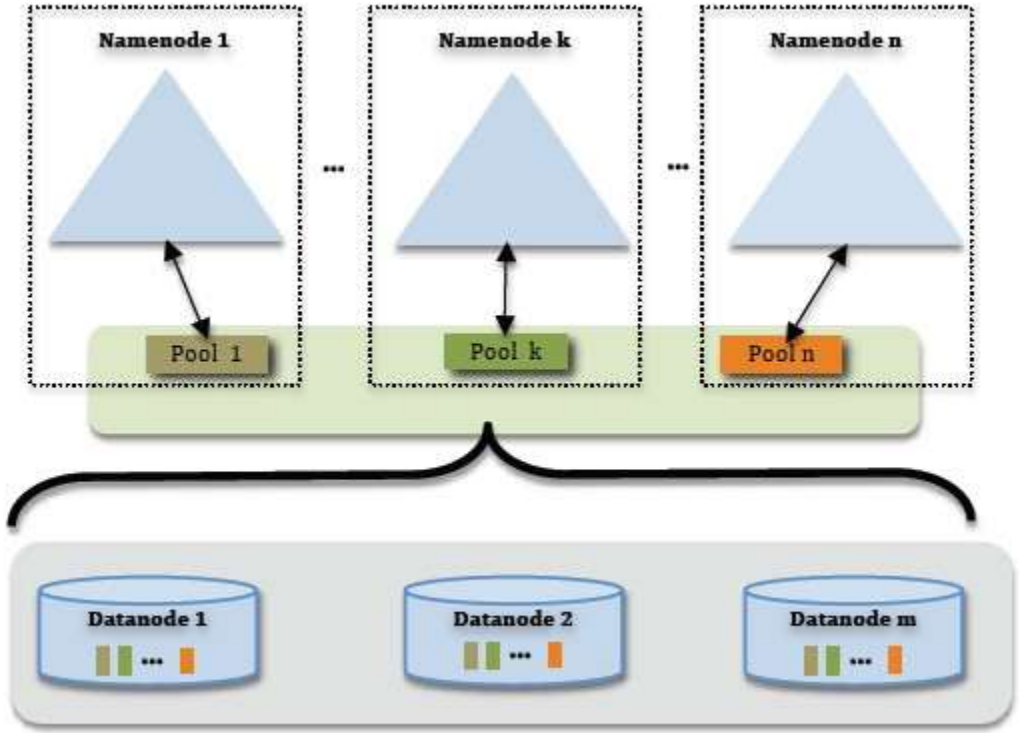
- Further improvements in development
  - JournalNodes
    - Quorum Journal Node: https://issues.apache.org/jira/browse/HDFS-3077
    - BookKeeper Journal Node: https://issues.apache.org/jira/browse/HDFS-3399



… in Hadoop 2.0

HDFS Federation

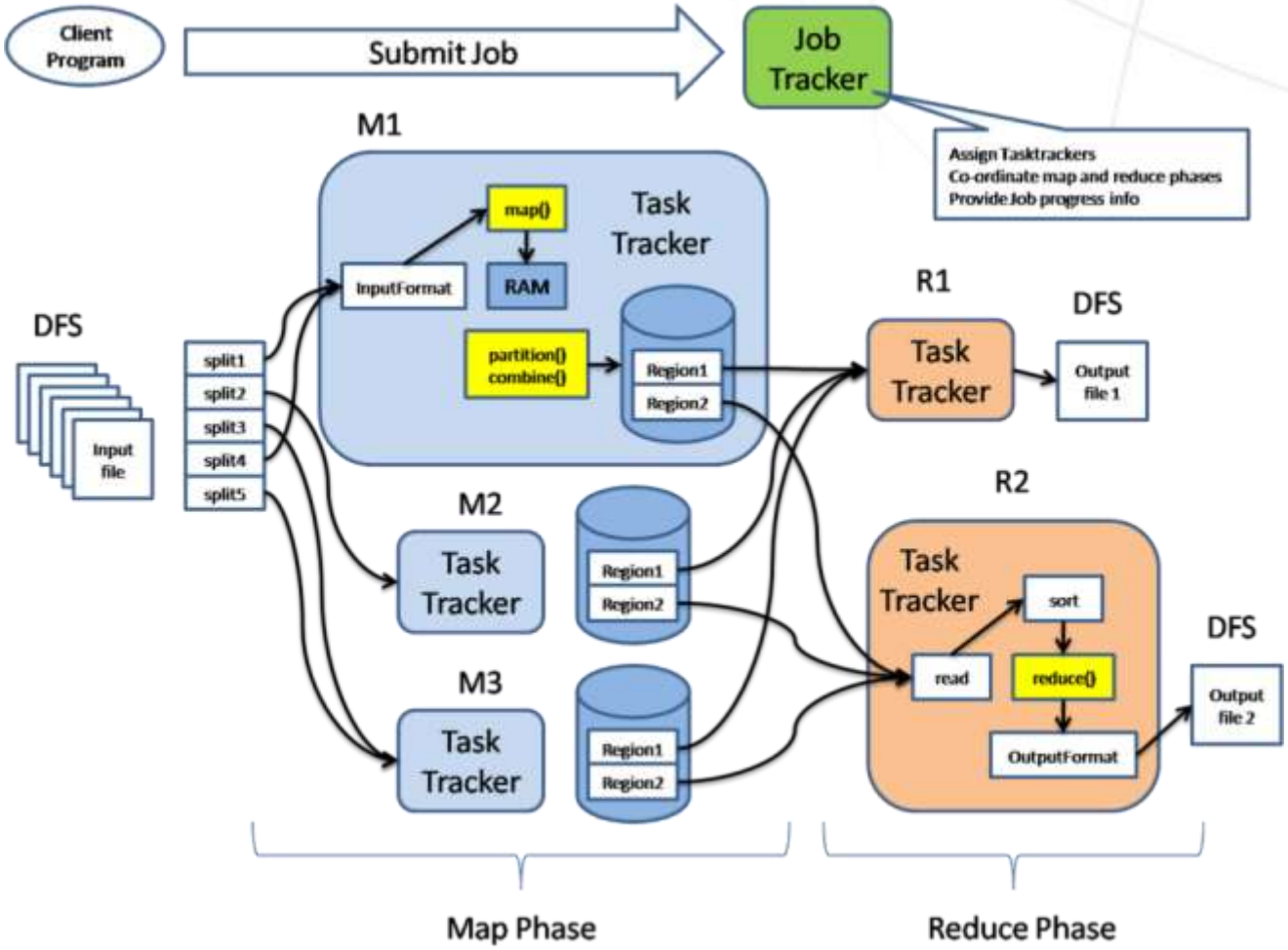
- Split the namespace for further reduction of failure domain scope
  - One big shared pool of DataNodes
  - Each NameNode manages only part of global namespace
  - Client uses mount table
  - Use HA for each NN
  - *Optional: Dedicated NN per app*



… in Hadoop 2.0

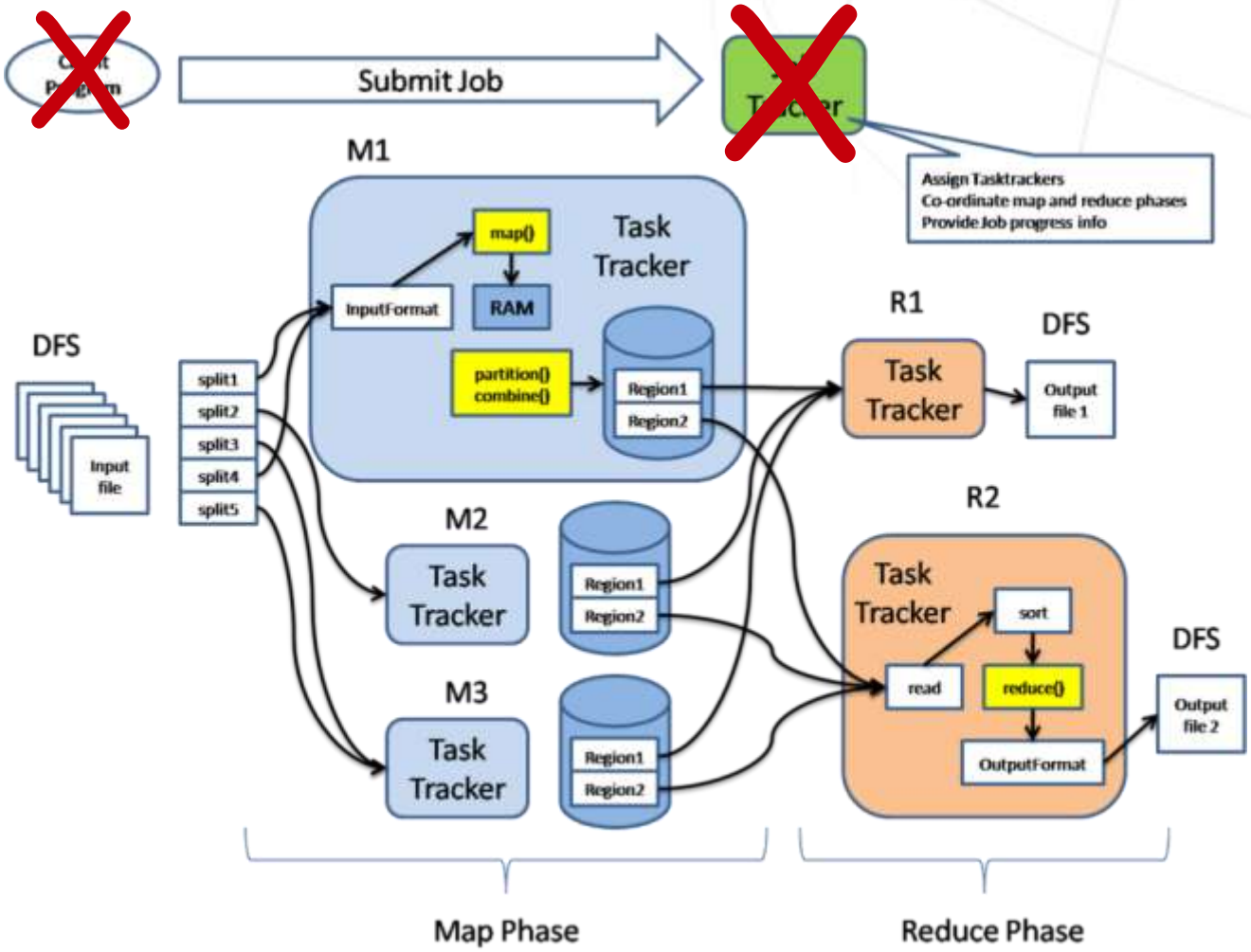
Where High Availability Is Crucial

- Problem: The singleton JobTracker



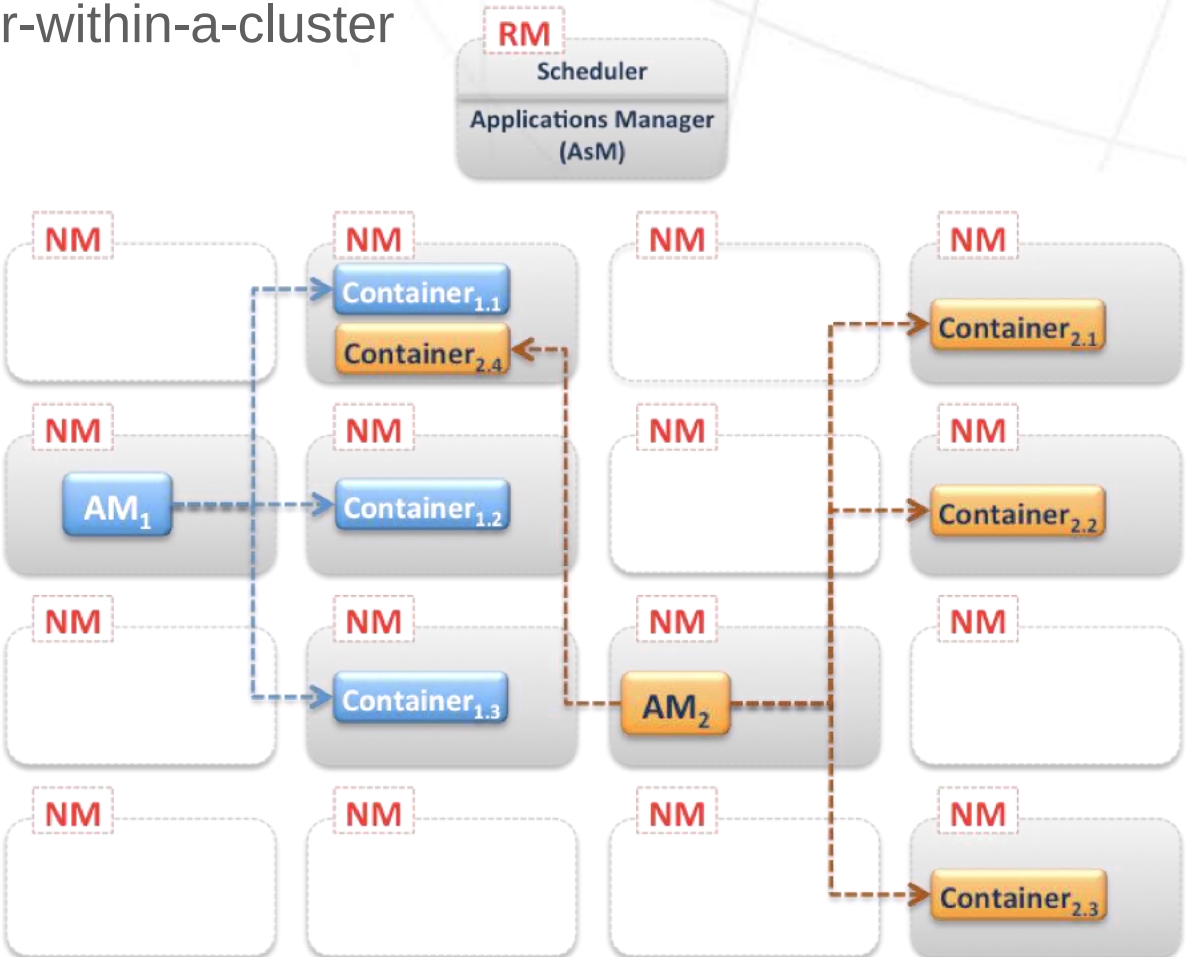
Where High Availability Is Crucial

- Problem: The singleton JobTracker



Where High Availability Is Crucial

- YARN (MapReduce v2) HA architecture
  - Highly available resource broker
  - Each job is a cluster-within-a-cluster

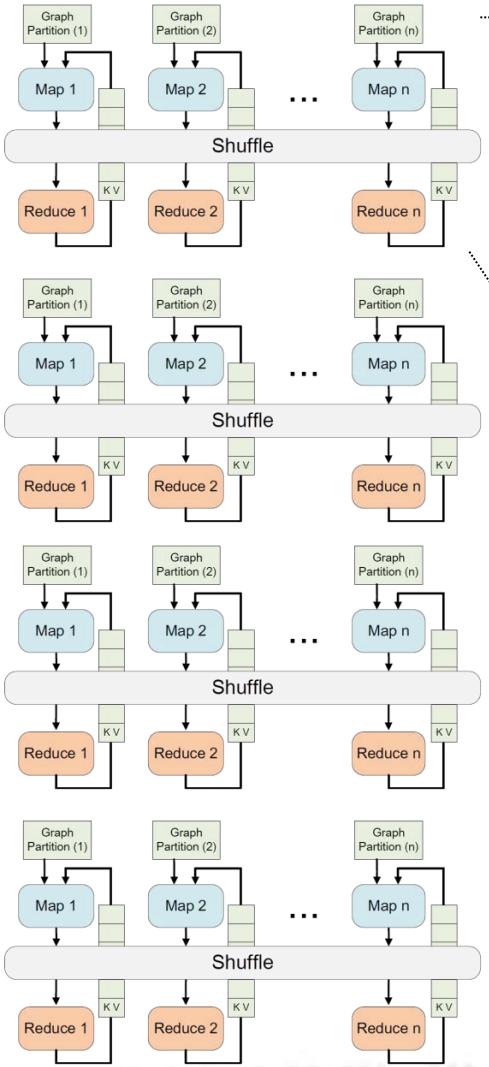


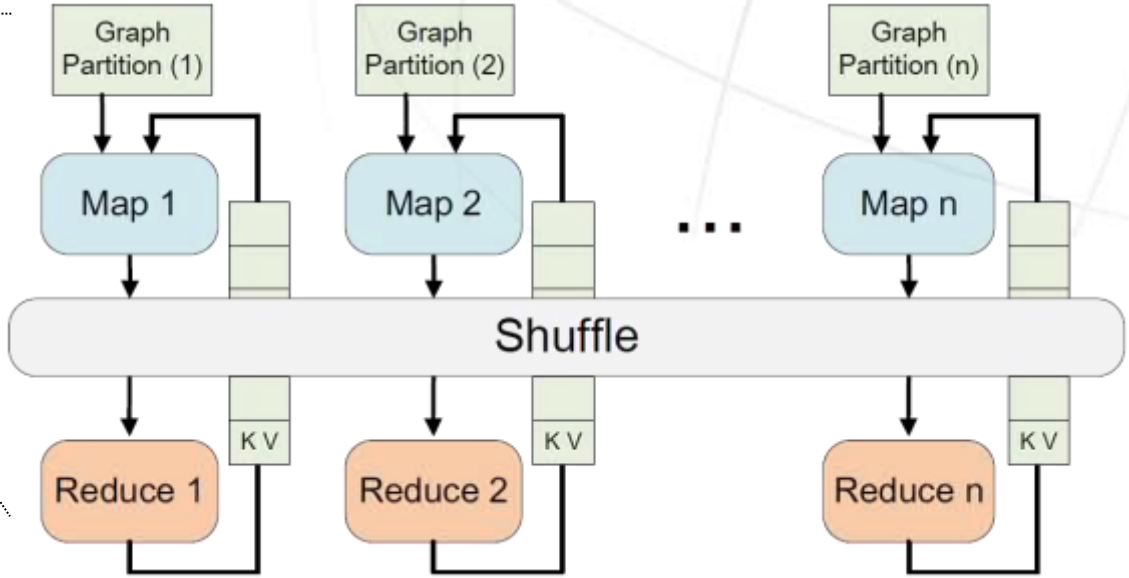
… in Hadoop 2.0

When MapReduce Is A Poor Fit

When MapReduce Is A Poor Fit

- Problem: MR can be inefficient for iterative workloads



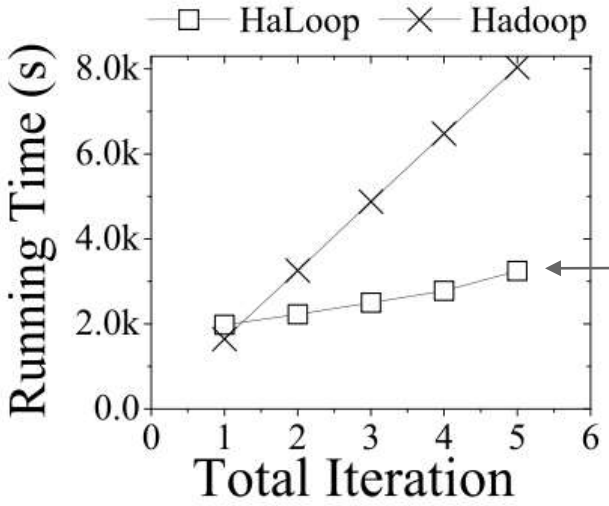


- Save and restore ***all*** state between iterations
- Large shuffle phase overhead on every job

  (For fault tolerance, all mapper output is written to then read in from disk.)

When MapReduce Is A Poor Fit

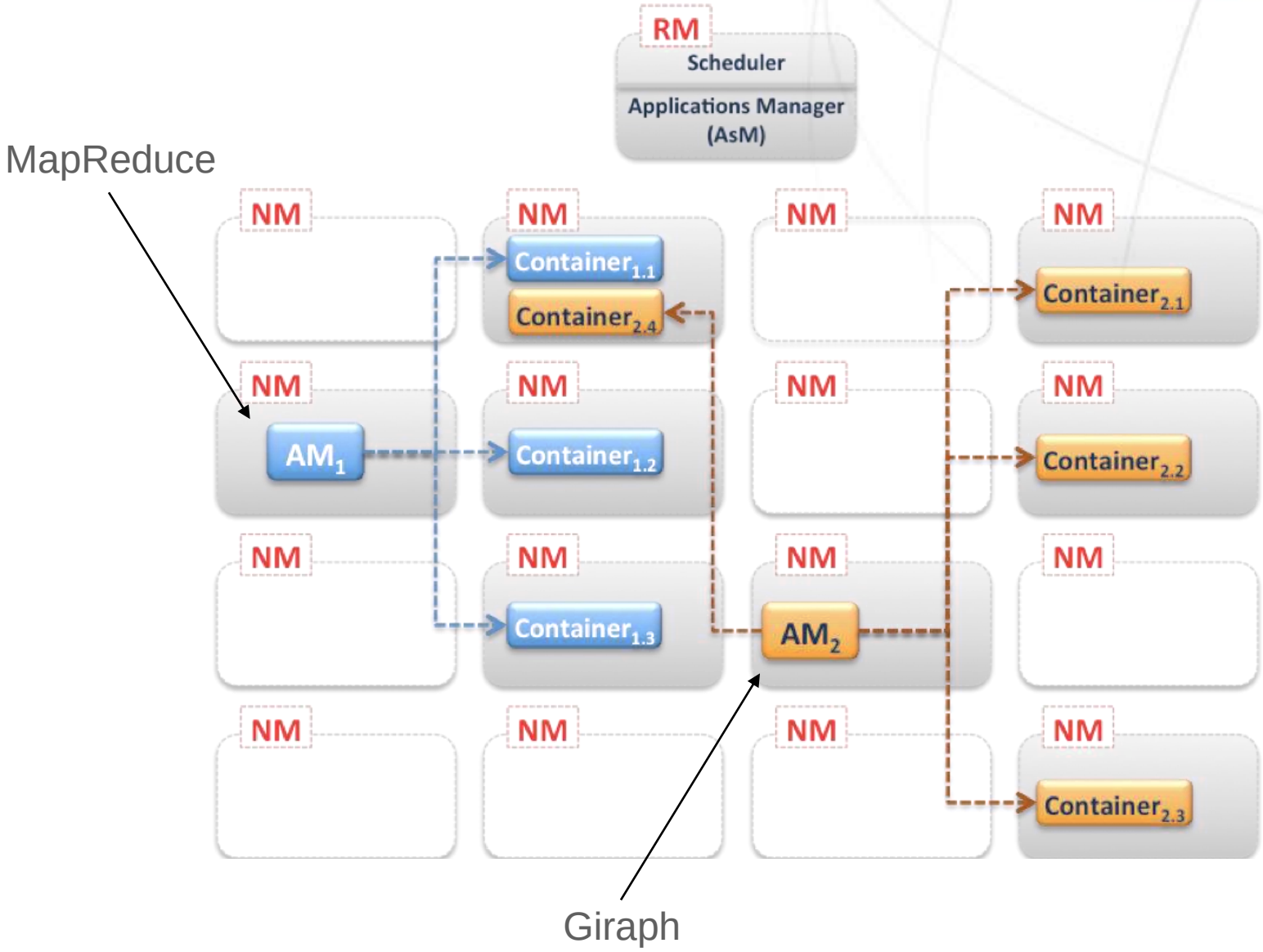
- YARN (MapReduce v2) generalizes Hadoop to admit new computational models *alongside* existing MR jobs
  - Apache Giraph: https://issues.apache.org/jira/browse/GIRAPH-13
  - Open MPI: https://issues.apache.org/jira/browse/MAPREDUCE-2911
  - Spark: https://github.com/mesos/spark-yarn/
  - HaLoop (http://clue.cs.washington.edu/node/14) and any other effort that produces an experimental MapReduce fork



Iteration costs become sub-linear when appropriate* computational models are used

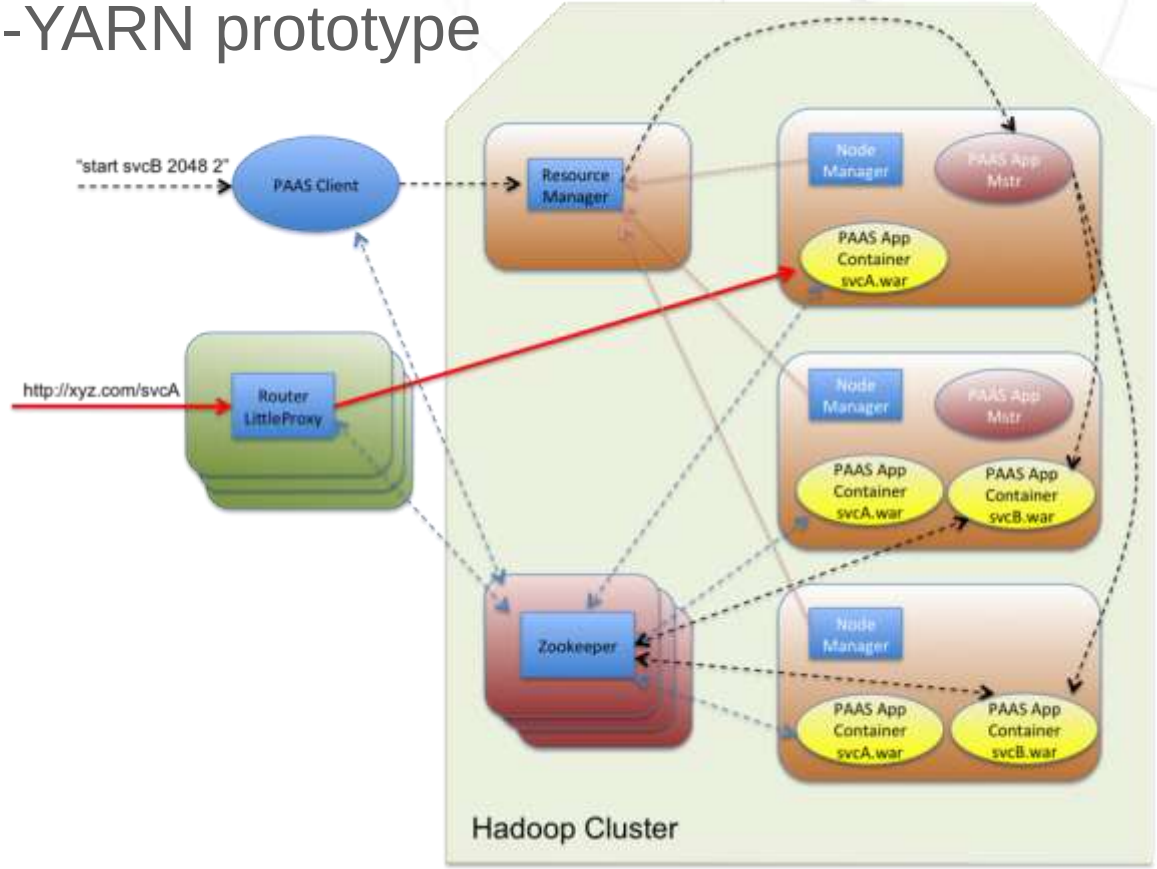
* - “Appropriate” is application specific

Yet Another Resource Negotiator



(Not just) Yet Another Resource Negotiator

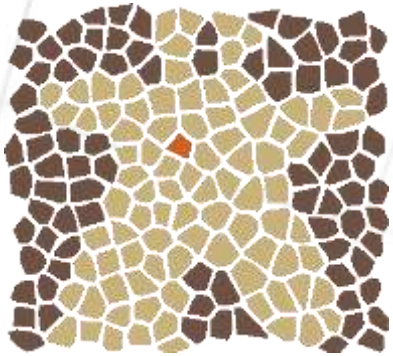
- A general purpose resource management framework
  - Containers can reserve and allocate CPU, memory, and disk resources independently; containers can execute any kind of program
- SAP Lab's PaaS-on-YARN prototype



http://jaigak.blogspot.com/2012/07/paas-on-hadoop-yarn-idea-and-prototype.html

Apache Giraph for Graph Algorithms

- Apache Giraph brings iterative in-memory graph computation to the Hadoop platform
  - http://giraph.apache.org/
  - Message passing paradigm and API
  - Checkpointing for fault tolerance
  - ZooKeeper for coordination
  - Inspired by Google's Pregel graph computation framework
  - Uses “Hadoop-like” concepts in the API for familiarity: writables, combiners, input and output formats
  - Hosts the iterative graph computation as a set of Hadoop mappers
  - Number of vertices possible scales linearly with number of worker nodes



When MapReduce Is A Poor Fit

- Generic pluggable shuffle under development
  - For example, for some jobs sorting can be skipped
  - MAPREDUCE-4049 (Plugin for generic shuffle service): https://issues.apache.org/jira/browse/MAPREDUCE-4049

- Hadoop Reduce is currently fixed into three phases
  - Shuffle: The framework fetches the relevant partition of the output of all the mappers (via HTTP)
  - Sort: The framework groups reducer inputs by keys
  - Reduce: Reducer tasks are executed on sorted output

  - Better transports than HTTP? (e.g. RDMA)
  - Clever partitioning?
  - Turn off sorting when it's not needed (e.g. HBase output formats)?

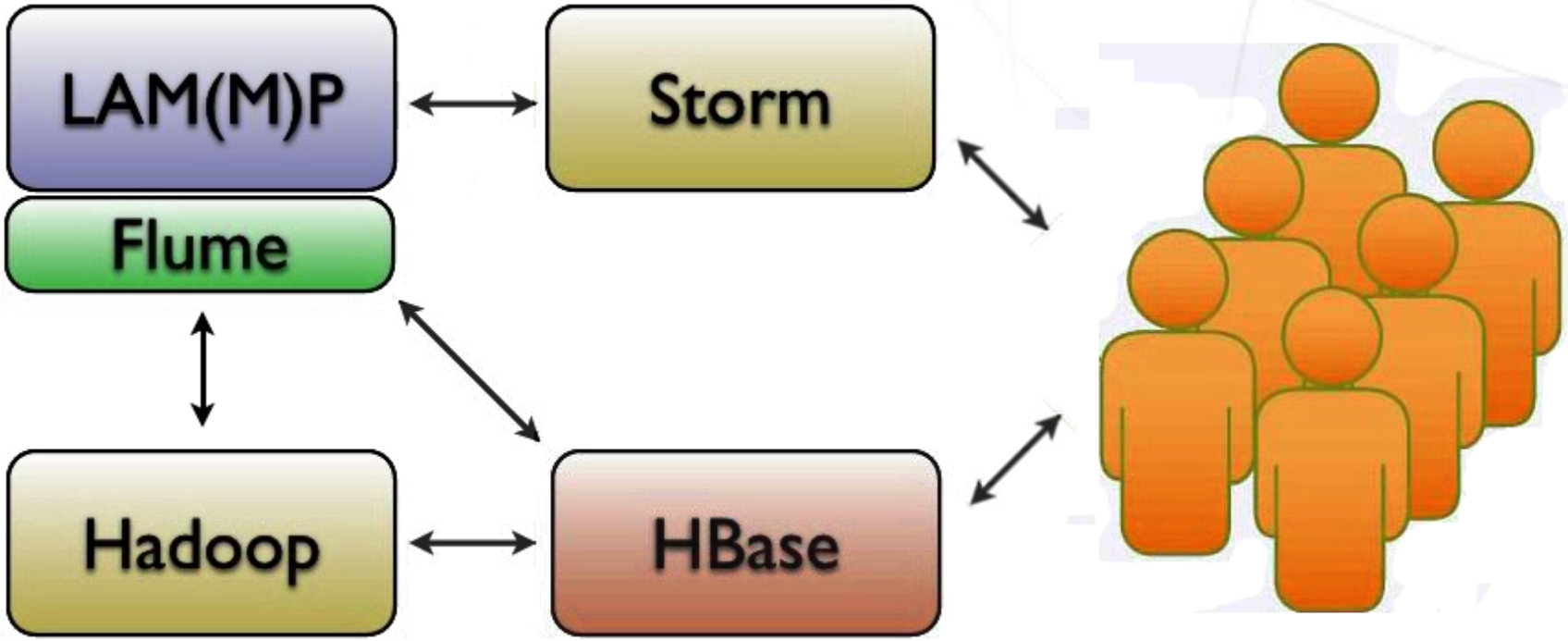
When Decisions Must Be Made Quickly

MapReduce Scheduling Latency

- Problem: MapReduce is a high throughput but also high latency batch processing model

- Hybrid architectures including Storm
  - Storm: http://storm-project.net/
  - "A distributed computation system for reliably processing unbounded streams of data, doing for (near-)realtime processing what Hadoop does for batch processing"
  - Metapattern
    - Hadoop does batch materialization of results from historical data
    - Shards of short term updates are produced by Storm topologies and overlaid on Hadoop results
      - Estimates are maintained for instantaneous query
      - Reads need to merge batch and stream results
    - Shards are then merged into history at the next batch iteration
      - "Eventual accuracy"

A Typical Hybrid Architecture Including Storm



From http://www.slideshare.net/larsgeorge/from-batch-to-realtime-with-hadoop-berlin-buzzwords-june-2012

HDFS Impedance Mismatch w/ Random Access

- Problem: HDFS is optimized for streaming large files, and limited to write-once-read-many use cases

- HBase supports other use cases that require random (indexed) access short reads and writes, which is why it is very commonly used with Hadoop today

The OLAP on HBase Trend

- Urban Airship's Datacube
  - https://github.com/urbanairship/datacube
  - Java implementation of a data cube* backed by a pluggable database backend, currently only supporting HBase, with a real-time query API
  - “We handle about ~10K events per second per node”

- Adobe's OLAP on HBase as a service
  - http://slidesha.re/KgokcC
  - Another data cube implementation built on HBase, offered as an internal SaaS service inside Adobe
  - Low latency ingestion with high throughput and a real-time query API

* - A data cube is a multidimensional generalization of the spreadsheet: http://en.wikipedia.org/wiki/OLAP_cube

When HQL Sets The Wrong User Expectations

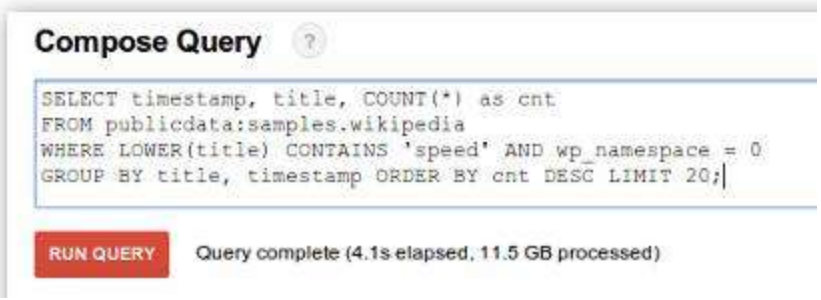
When HQL Sets The Wrong User Expectations

- SQL-like queries against data stored in Hadoop should return answers as quickly as competing solutions (for example, MPP databases)

- By presenting a SQL-like language (HQL), Apache Hive can set inappropriate user expectations
  - The relative simplicity of SQL is a user accessibility advantage
  - The easy integration with ODBC/JDBC is an integration advantage
  - However, *every* Hive query uses MapReduce as execution engine, and suffers job setup latency

When HQL Sets The Wrong User Expectations

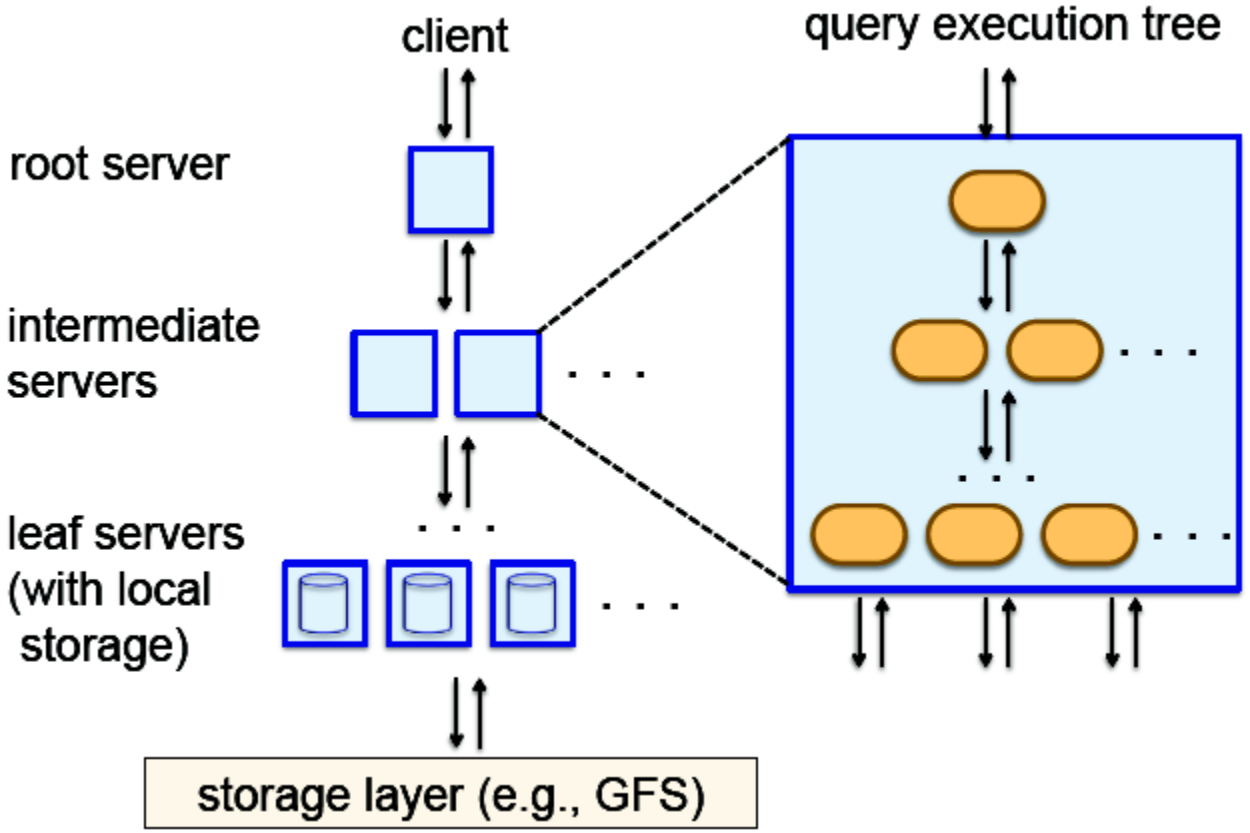
- Apache Drill (incubating)
  - http://incubator.apache.org/projects/drill.html
  - A distributed system for interactive analysis of large-scale datasets, inspired by Google's Dremel
  - Core rationale: “Hadoop is designed to achieve very high throughput, but is not designed to achieve the sub-second latency needed for interactive data analysis and exploration.”
- If Drill is realized, fast SQL-like interaction with datasets of billions or trillions of data items will be possible
  - Dremel is the engine behind BigQuery, and makes Big Data seem small



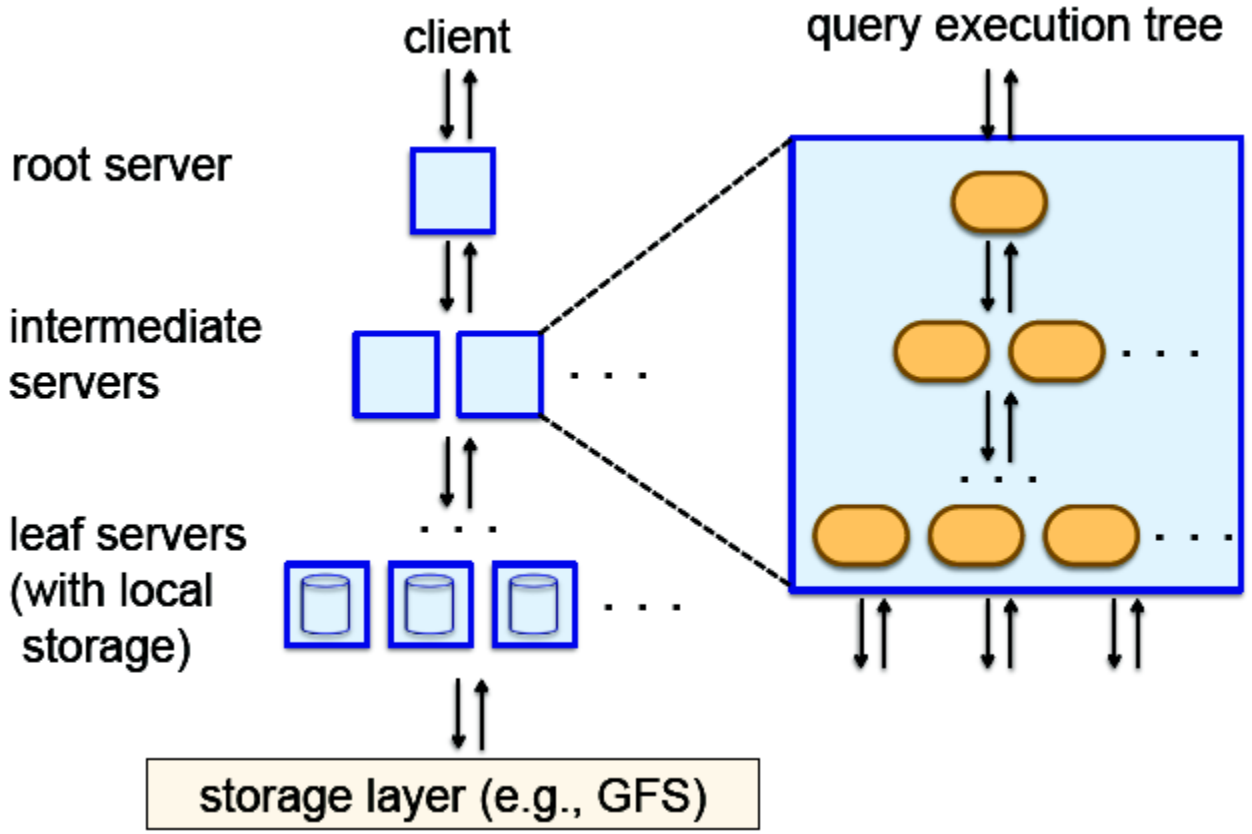
Drill Vs. Hadoop

- Drill as an open clone of Dremel
  - Dremel paper: http://research.google.com/pubs/pub36632.html
- To be built on top of the Hadoop platform
  - Dremel requires an ETL process for importing raw data into its column store format – MapReduce serves this purpose
  - Data files hosted in HDFS
  - Drill will also be able to query alternate stores, e.g. HBase
- Why is Drill faster than MapReduce? (Or Hive, or Pig...)
  - Column striped storage substantially reduces I/O
  - Executes on data in-place
  - Uses aggregator trees instead of map-reduce
    - Queries are pushed down to a tree of execution engines and rewritten at each step, lower level results merged back up

Dremel Execution Model



Dremel Execution Model



Strongly resembles search engine architecture

Issues Presented By the Java Virtual Machine

JVM Sandbox Limits Performance Optimizations

- Problem: The Java Virtual Machine abstractions prevents many OS and hardware level* performance optimizations

- Recent changes to Hadoop call out to Linux-specific JNI extensions for good performance wins
  - Hints to the OS blockcache via fadvise()
  - Hardware CRC32 checksum calculations

… in Hadoop 2.0

- The JVM is open source – OpenJDK – so modifying the sandbox instead of breaking out of it is an option

* - Yes, it is still possible to design cache-aware data structures

Garbage Collection Limits Large Memory Use

- Problem: Practical JVM garbage collector limitations prevent effective use of large memory systems

- Various efforts to build in an efficient second level cache outside the GC domain
  - Teracotta BigMemory (commercial product): http://terracotta.org/products/bigmemory
  - Taobao's open source direct OpenJDK modifications: http://www.slideshare.net/RednaxelaFX/jvm-taobao
- New G1 GC collector is workable since JDK 1.7_u4 (?)
  - G1 means “garbage first”
  - Targeted for multi-processor machines with large memories.
  - The promise: Low (~10s of ms) GC pause time goals are met with high probability even with very large heaps
  - http://www.oracle.com/technetwork/java/javase/tech/g1-intro-jsp-135488.html

Issues Presented By Solid State Storage

Lack Of Storage Device Awareness

- Problem: No HDFS support for tiered storage
  - HDFS cannot differentiate between block device types
  - No optimizations available for SSD storage
  - No ability for higher layers in the stack (e.g. HBase) to detect and leverage SSDs

- If we had this support, we could consider ideas such as adaptive differentiated block placement for HBase
  - An early idea in this space
  - HBase can detect SSD storage and relocate blocks for CFs with strong random access read pattern there
  - https://issues.apache.org/jira/browse/HDFS-2832
  - https://issues.apache.org/jira/browse/HBASE-6572

Low SSD Latencies Expose New Challenges

- Problem: Orders of magnitude change in I/O latency means the bottleneck shifts to locking, data structures, internal JVM contention points

- Dhruba Borthakur at Facebook experimented with HBase and SSDs and identified several issues
  - http://hadoopblog.blogspot.com/2012/05/hadoop-and-solid-state-drives.html
  - "*A database that is layered above HDFS would not be able to utilize all the IOPS offered by a single SSD*" because the DFSClient code paths are long – Need to investigate optimized DFS clients, DataNode code paths, better read and caching strategies
  - "*Heavy lock contention and heavy context switching causes the regionserver to not be able to use all the available CPU on the machine*" – Need to investigate reducing the number of threads in a RegionServer, nonblocking IO and RPC

Where GPU Offload Is Desired

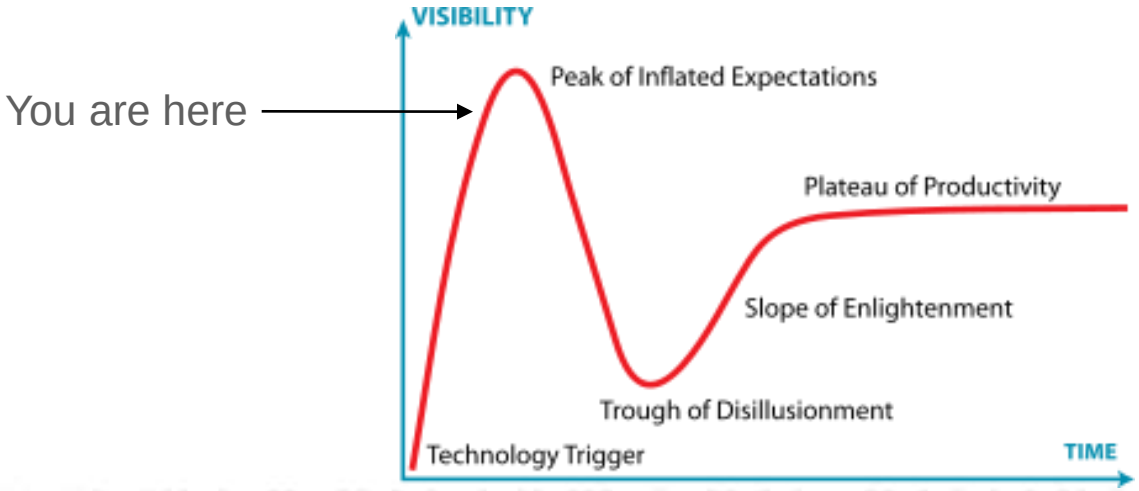
GPU Computation Offload

- Problem: Offload of parallel / vector math to GPU is a promising application performance optimization and increasingly popular in scientific computing
- Not really considered yet in the Hadoop context
- The Rootbeer project can convert Java bytecode into a native GPU application
  - https://github.com/pcpratts/rootbeer1
- Perhaps framework integration for tiered architectures
  - MapReduce partitions the problem domain in parallel in the large
  - Computation over each partition in processed locally on the GPU's vector engine
  - MapReduce combines the results in the reduce phase

Summary And Predictions

Hadoop And “Big Data”

- “Big Data” as a marketing term may be a fad, but the underlying phenomenon is not and is only getting bigger
- Hadoop is and will be a widely used commodity operating system for Big Data for the next 3-5 years
- The Hadoop ecosystem will continue a rapid pace of evolution, but the innovation will not outpace a “trough of disillusionment”



Hadoop Ecosystem Evolution

- Vendors will have to differentiate, which will maintain diversity in the ecosystem, but with some fragmentation
- Short term developments (1-2 years) I expect to see
  - HDFS and HBase will see closer co-development, especially regarding solid state storage
  - Apache YARN as a new TLP and, eventually, an ecosystem that goes beyond Hadoop roots into the larger grid computing space
  - Apache Giraph continues to grow and do well
  - Apache Drill is realized

End

Q&A