Petabyte-scale Data with Apache HDFS

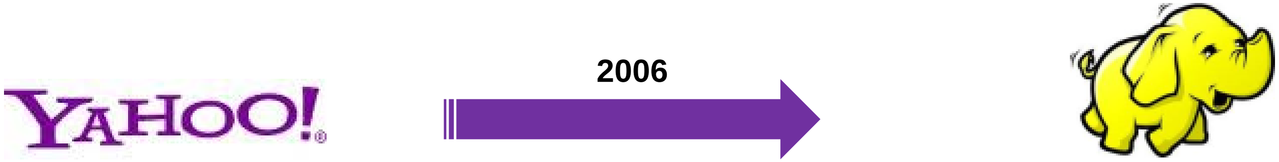
Matt Foley
Hortonworks, Inc.
mfoley@hortonworks.com

Matt Foley - Background

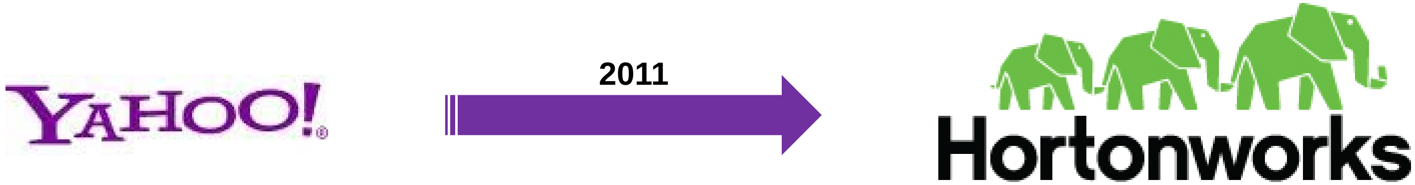
- **MTS at Hortonworks Inc.**
  - HDFS contributor, part of original ~25 in Yahoo! spin-out of Hortonworks
  - Currently managing engineering infrastructure for Hortonworks
  - My team also provides Build Engineering infrastructure services to ASF, for Hadoop core and several related projects within Apache
  - Formerly, led software development for back end of Yahoo Mail for three years – 20,000 servers with 30 PB of data under management, 400M active users
  - Did startups in Storage Management and Log Management

- **Apache Hadoop, ASF**
  - Committer and PMC member, Hadoop core
  - Release Manager – Hadoop-1.0

Company Background



- In 2006, Yahoo! was a very early adopter of Hadoop, and became the principle contributor to it.
- Over time, invested *40K+ servers* and *170PB storage* in Hadoop
- Over 1000 active users run 5M+ Map/Reduce jobs per month
- In 2011, Yahoo! spun off ~25 engineers into Hortonworks, a company focused on advancing open source Apache Hadoop for the broader market (http://www.wired.com/wiredenterprise/2011/10/how-yahoo-spawned-hadoop)



Tutorial

- **Agenda**
  - –Overview of HDFS architecture – ½ hour
  - –Hardware choices
  - –Rack topology awareness
  - –Federated metadata servers (Hadoop 2.0)
  - –Other Hadoop Improvements
  - –Calculating the probability of data loss

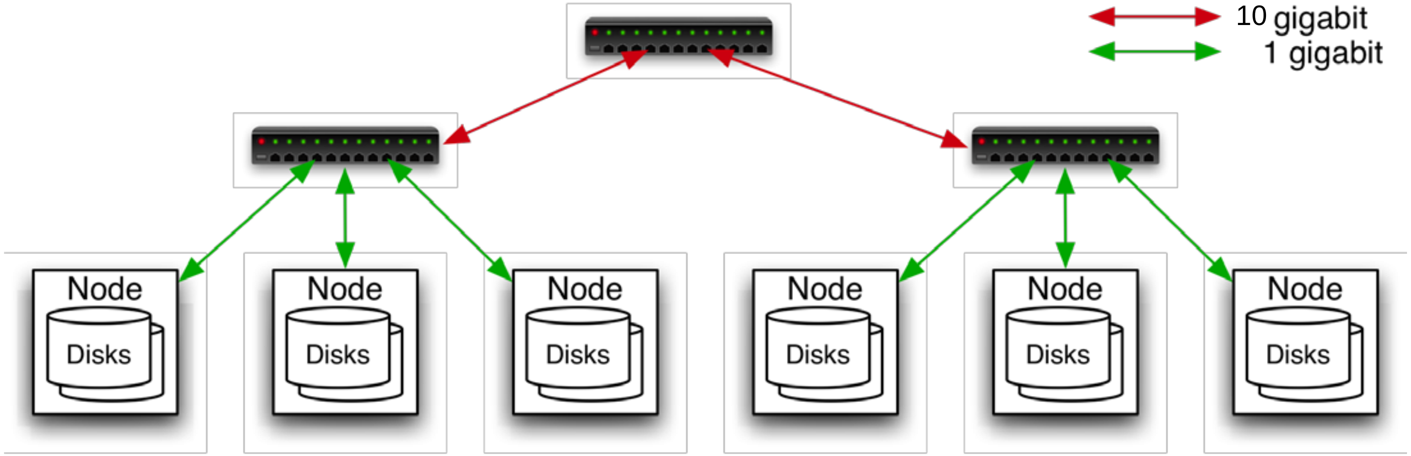
Overview of HDFS and Hadoop

- **What is Hadoop?**
- **HDFS Architecture**
- **Using Hadoop: MapReduce example**
- **Hadoop Ecosystem**

What is Hadoop?

- **Hadoop - Open Source Apache Project**
  - Framework for reliably storing & processing petabytes of data using *commodity* hardware and storage
- **Scalable solution**
  - Computation capacity
  - Storage capacity
  - I/O bandwidth
- **Core components**
  - HDFS: Hadoop Distributed File System - distributes data
  - Map/Reduce - distributes application processing and control
- **Move computation to data and not the other way**
- **Written in Java**
- **Runs on**
  - Linux, Windows, Solaris, and Mac OS/X

Commodity Hardware Cluster

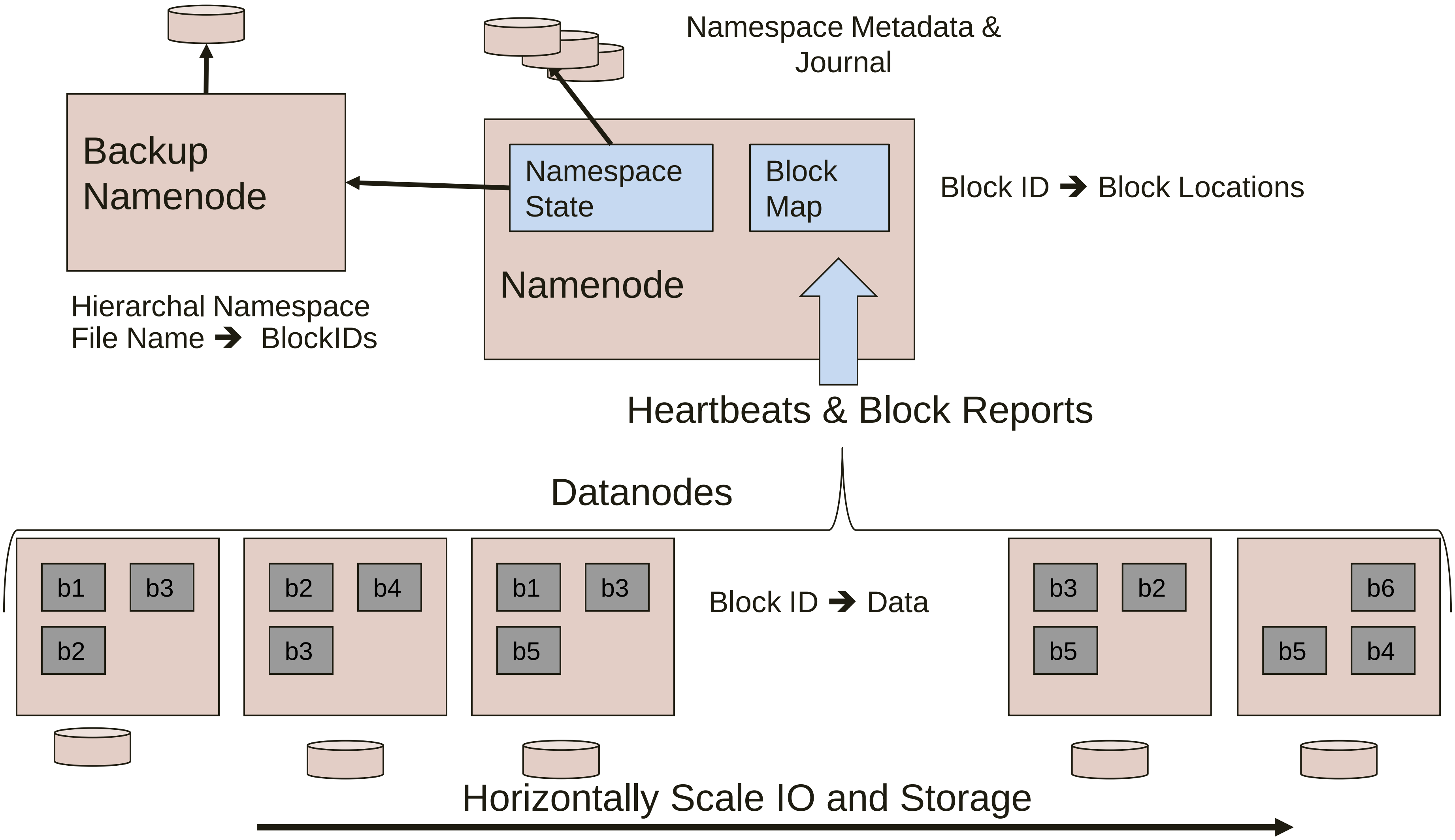


- **Typically in 2- or 3-level architecture**
  - – Nodes are commodity Linux servers
  - – 20 - 40 nodes/rack
  - – Uplink from rack is 10 or 2x10 gigabit
  - – Rack-internal is 1 or 2x1 gigabit all-to-all
- **“Flat fabric” 10Gbit network architectures being planned at growing number of sites**

Hadoop Distributed File System (HDFS)

- **One PB-scale file system for the entire cluster**
  - Managed by a single *Namenode*
  - Files are written, read, renamed, deleted, but append-only
  - Optimized for streaming reads of large files
- **Files are broken into uniform sized blocks**
  - Blocks are typically 128 MB (nominal)
  - Replicated to several *Datanodes*, for reliability
  - Exposes block placement so that computation can be migrated to data
- **Client library directly reads data from Data Nodes**
  - Bandwidth scales linearly with the number of nodes
  - System is topology-aware
  - Array of block locations is available to clients

HDFS Diagram



Namespace Metadata & Journal

Backup Namenode

Namespace State

Block Map

Namenode

Block ID ➔ Block Locations

Hierarchal Namespace
File Name ➔ BlockIDs

Heartbeats & Block Reports

Datanodes

Block ID ➔ Data

Horizontally Scale IO and Storage

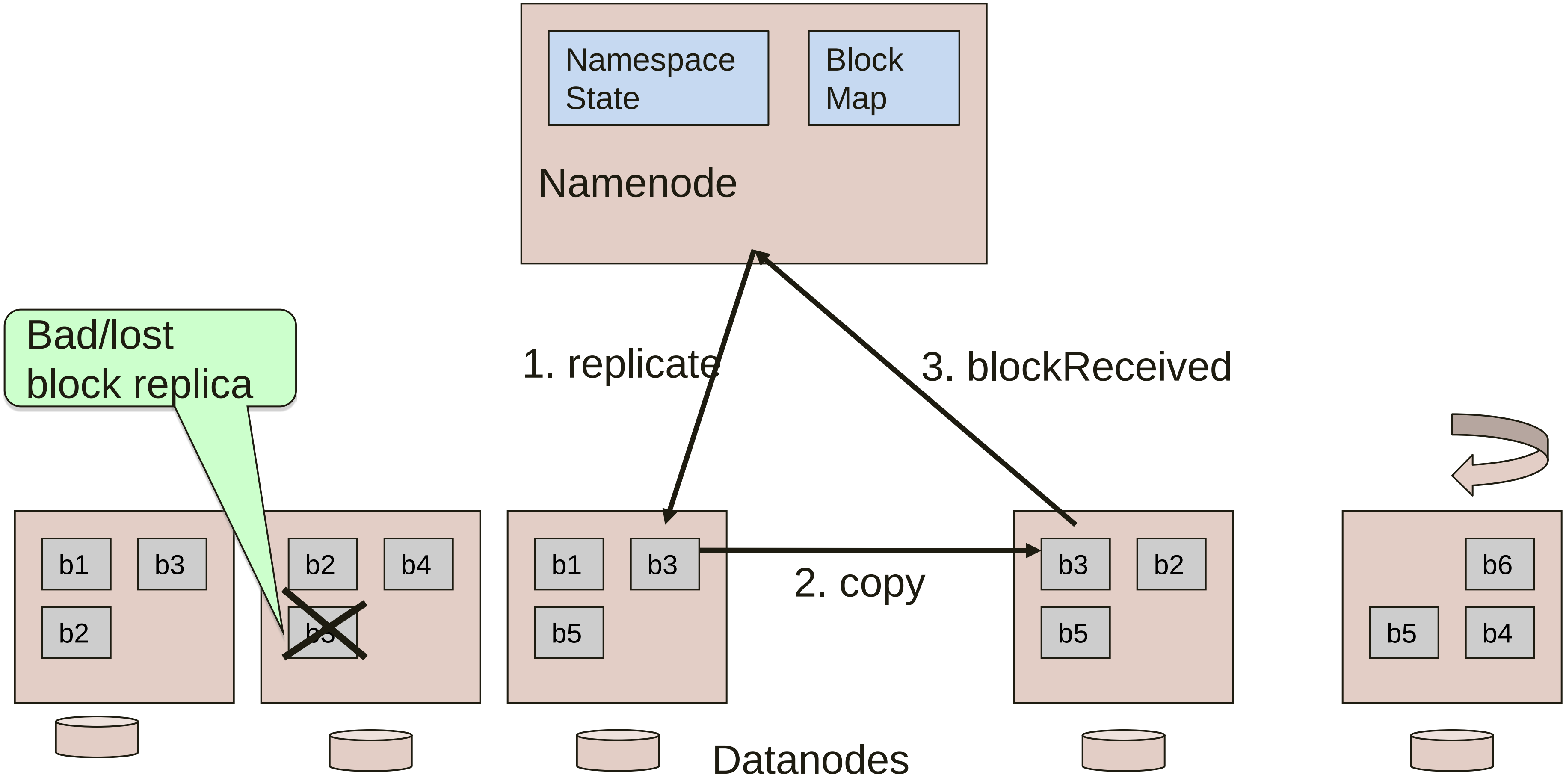
Block Placement

- **Default is 3 replicas, but settable**
- **Blocks are placed (writes are pipelined):**
  - –First replica on the local node or a random node on local rack
  - –Second replica on a remote rack
  - –Third replica on a node on same remote rack
  - –Other replicas randomly placed
- **Clients read from closest replica**
  - –System is topology-aware
- **Block placement policy is pluggable**

Block Correctness

- **Data is checked with CRC32**
- **File Creation**
  - –Client computes block checksums
  - –DataNode stores the checksums
- **File access**
  - –Client retrieves the data and checksum from DataNode
  - –If Validation fails, Client tries other replicas
- **Periodic validation by DataNode**
  - –Background DataBlockScanner task

HDFS Data Reliability



Active Data Management

- **Continuous replica maintenance**
- **End-to-end checksums**
- **Periodic checksum verification**
- **Decommissioning nodes for service**
- **Balancing storage utilization**

Map/Reduce

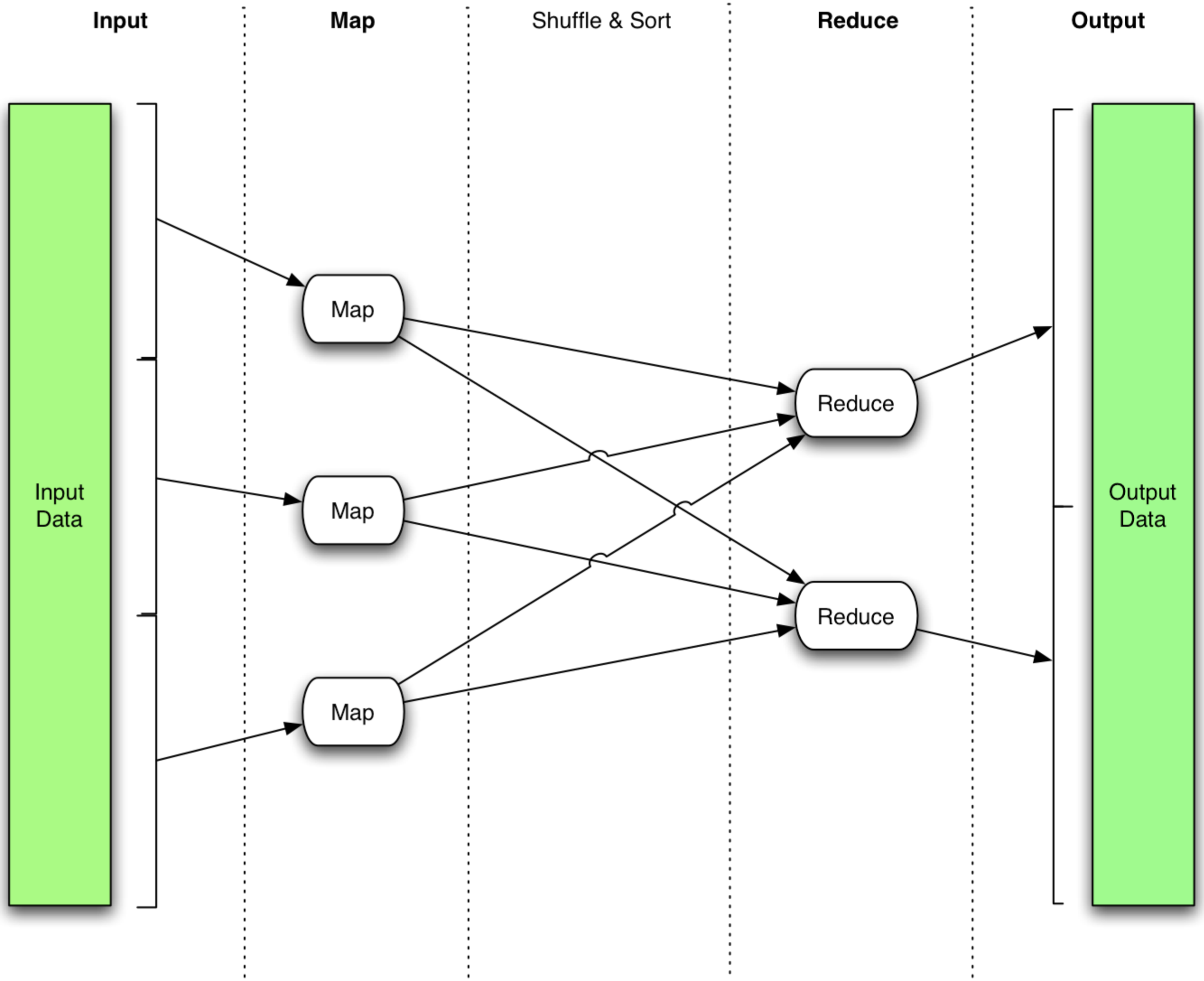
- **Programming model for efficient distributed computing**

- **Works sort of like a Unix pipeline:**
  - cat input | grep | sort | uniq -c > output
  - Input | **Map** | Shuffle & Sort | **Reduce** | Output

- **Another analogy: Inhale/Exhale/Repeat**
  - See next slide

- **Strengths:**
  - Usable by majority of software developers
  - Streaming through data, reducing seeks
  - Pipelining of processing
  - Automatic reliability and re-execution on failure

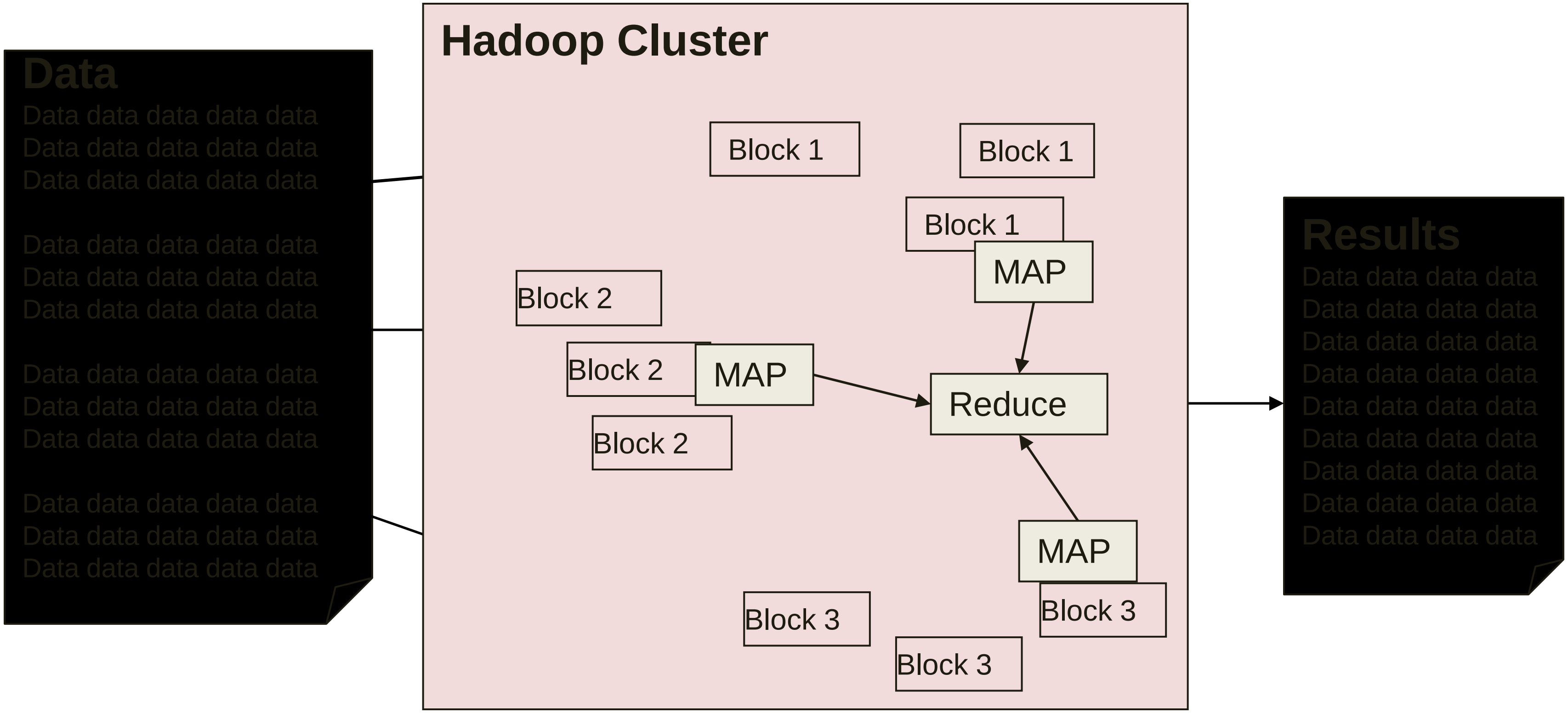
Map/Reduce Data Flow



Map/Reduce features

- **Java, C++, and text-based APIs**
  - In Java use Objects, in C++ use bytes
  - Text-based (streaming) great for scripting or legacy apps
  - Higher level interfaces: Pig, Hive, others
- **Automatic re-execution on failure**
  - Every Map and every Reduce is a task per node
  - In a large cluster, some nodes are always slow or flaky
  - Framework re-executes failed tasks
- **Locality optimizations**
  - With large datasets, bandwidth to data must be managed
  - Map-Reduce queries HDFS for locations of input data
  - Map tasks are scheduled close to the inputs when possible

Computation close to the data



Data

Data data data data data
Data data data data data
Data data data data data

Data data data data data
Data data data data data
Data data data data data

Data data data data data
Data data data data data
Data data data data data

Data data data data data
Data data data data data
Data data data data data

Hadoop Cluster

Block 1

Block 1

Block 1

MAP

Block 2

Block 2

MAP

Reduce

Block 2

MAP

Block 3

Block 3

Block 3

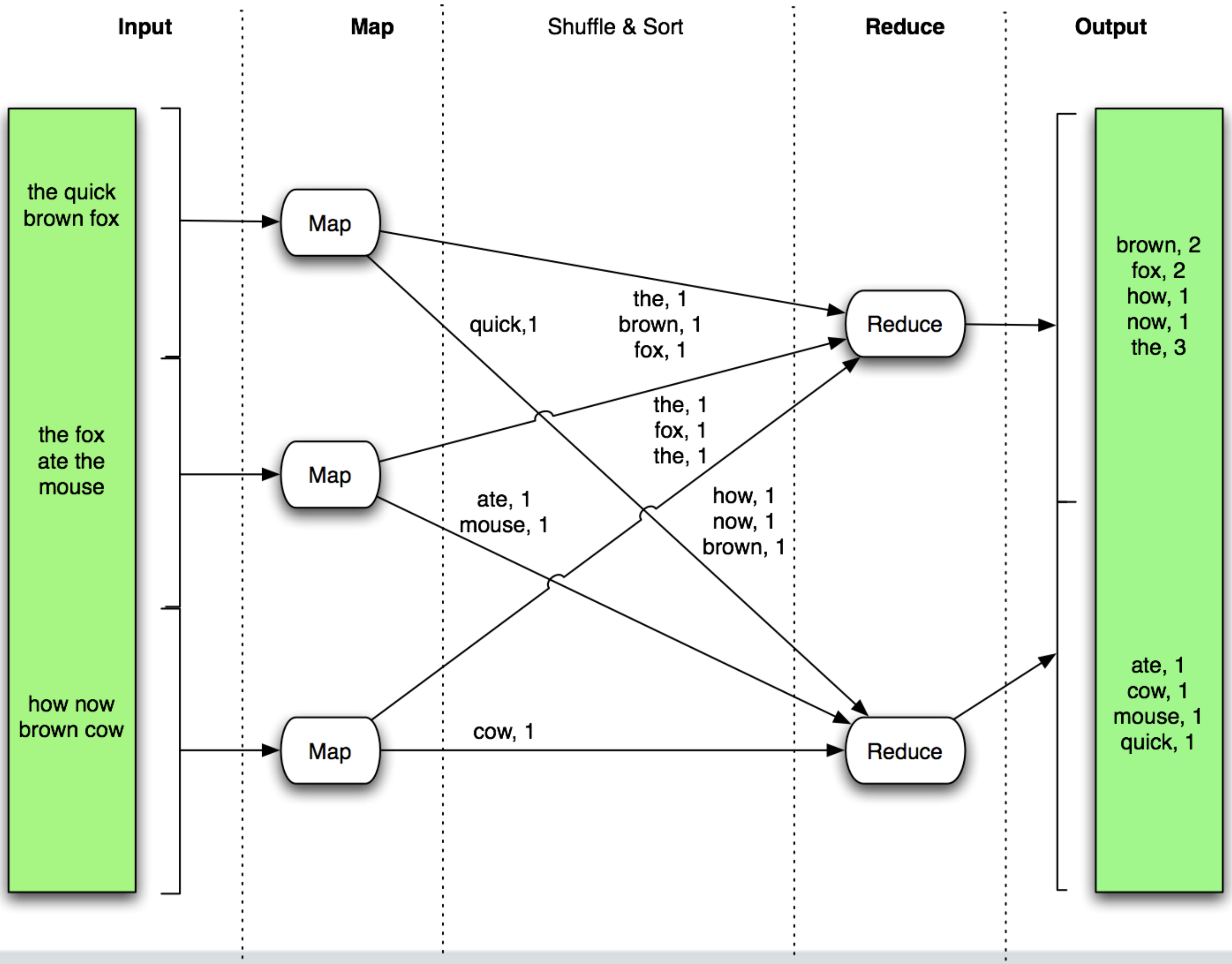
Results

Data data data data
Data data data data
Data data data data
Data data data data
Data data data data
Data data data data
Data data data data
Data data data data
Data data data data

Word Count Example

- **Mapper**
  - Input: value: lines of text of input
  - Output: key: word, value: 1
- **Reducer**
  - Input: key: word, value: set of counts
  - Output: key: word, value: sum
- **Launching program**
  - Defines the job
  - Submits job to cluster

Word Count Dataflow



Example: Word Count Mapper

```
public static class MapClass extends MapReduceBase
    implements Mapper<LongWritable, Text, Text, IntWritable> {

    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

    public void map(LongWritable key, Text value,
                    OutputCollector<Text, IntWritable> output,
                    Reporter reporter) throws IOException {
      String line = value.toString();
      StringTokenizer itr = new StringTokenizer(line);
      while (itr.hasMoreTokens()) {
        word.set(itr.nextToken());
        output.collect(word, one);
      }
    }
  }
```

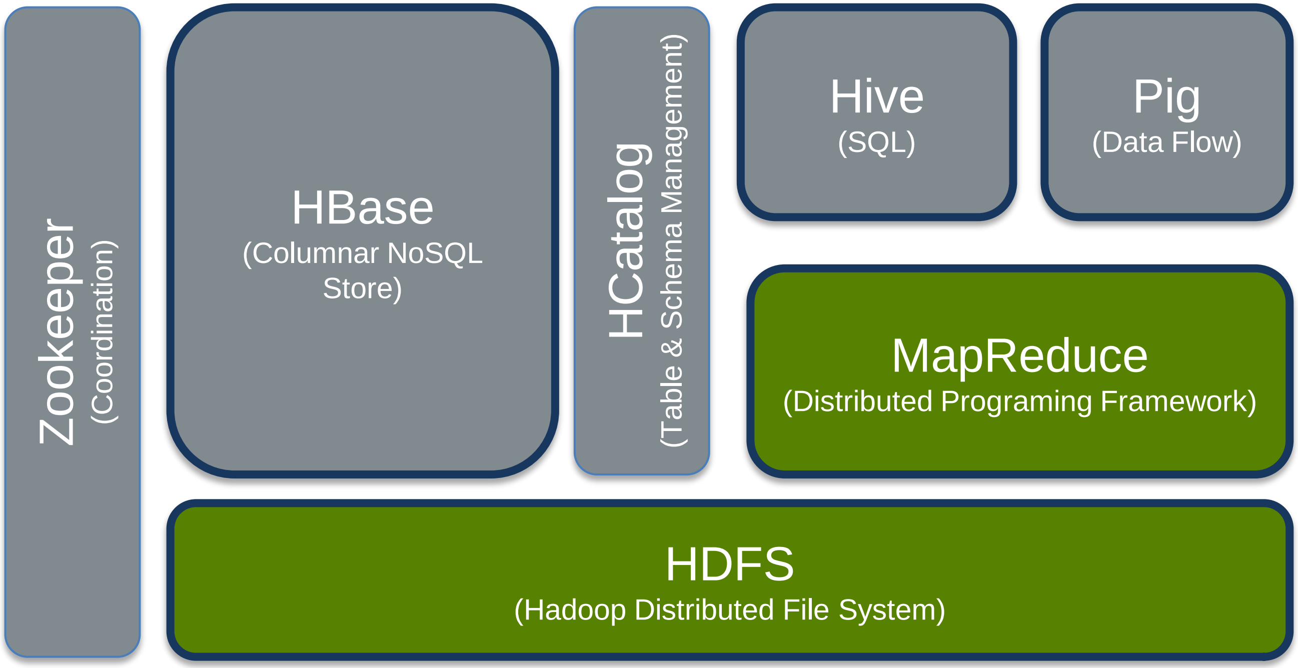
Example: Word Count Reducer

```
public static class Reduce extends MapReduceBase
   implements Reducer<Text, IntWritable, Text, IntWritable> {

   public void reduce(Text key, Iterator<IntWritable> values,
                      OutputCollector<Text, IntWritable> output,
                      Reporter reporter) throws IOException {
     int sum = 0;
     while (values.hasNext()) {
       sum += values.next().get();
     }
     output.collect(key, new IntWritable(sum));
   }
 }
```

Key Hadoop Ecosystem Components

🟩 Core Apache Hadoop ⬜ Related Hadoop Projects



Zookeeper
(Coordination)

HBase
(Columnar NoSQL Store)

HCatalog
(Table & Schema Management)

Hive
(SQL)

Pig
(Data Flow)

MapReduce
(Distributed Programing Framework)

HDFS
(Hadoop Distributed File System)

Hadoop Ecosystem

- **Many other projects in the Hadoop family**
- **Most importantly: HBase**
  - Distributed “NoSQL” column store database
- **Pig: Dataflow language**
- **Hive: SQL-based Data warehouse infrastructure**
- **HCatalog: Meta-data sharing infrastructure**
  - For Pig, Hive, and HBase
- **Zookeeper: Distributed coordination**
- **Oozie: Workflow engine**
- **Mahout: Scalable machine learning and data mining library**
- **The list is growing…**

Hadoop Ecosystem (cont.)

- **Monitoring**
  - Nagios
  - Ganglia
  - Under development – new Apache Incubator project Ambari: Unified Hadoop-centric Monitoring dashboard and Management Console

Tutorial

- **Agenda**
  - –Overview of HDFS architecture
  - –Hardware choices
  - –Rack topology awareness
  - –Federated metadata servers (Hadoop 2.0)
  - –Other Hadoop Improvements
  - –Calculating the probability of data loss

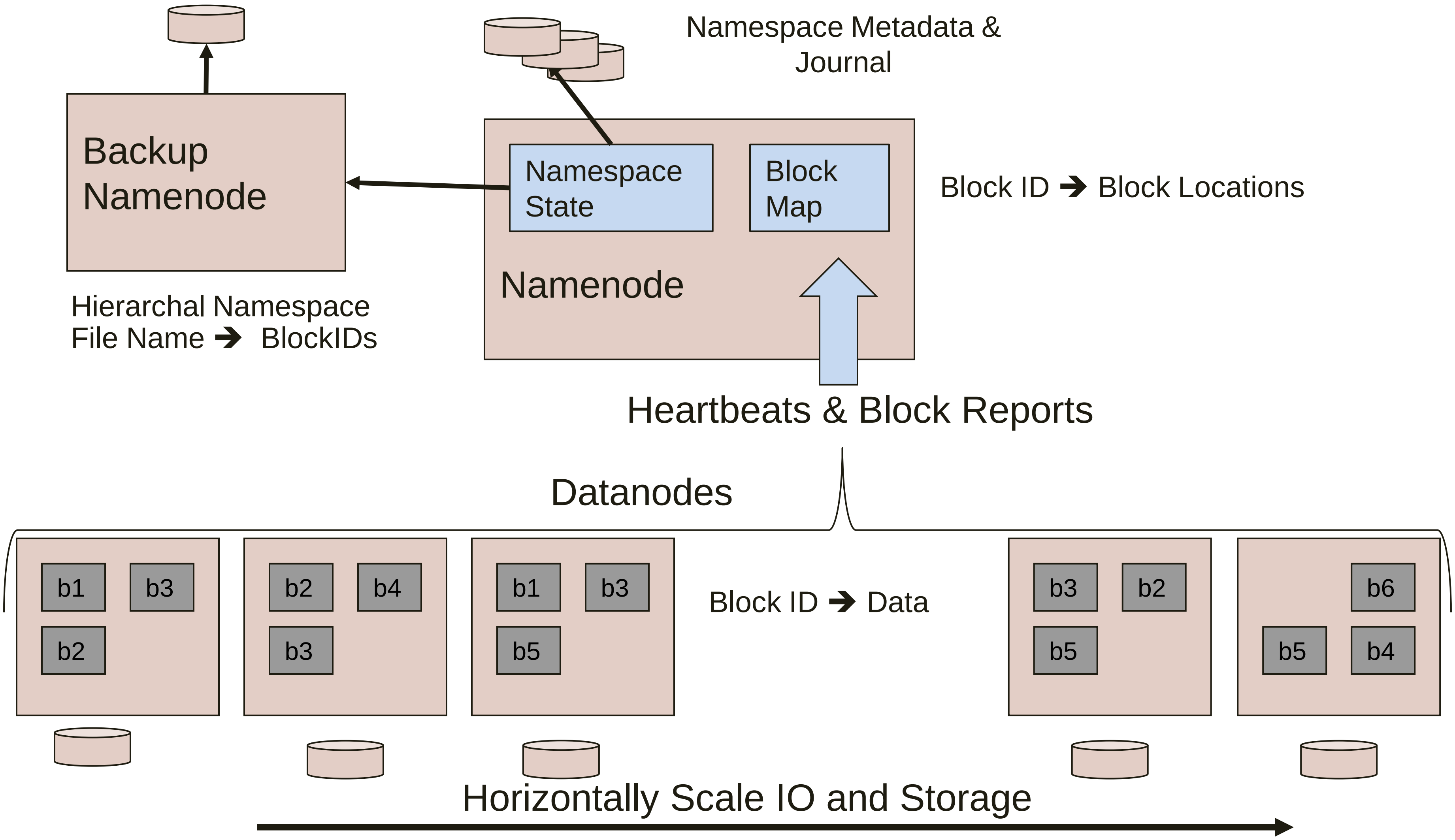
Hardware Selection for New Clusters

- **Every site has a different job mix, varying by**
  - – Type
  - – Size
  - – Frequency
  - – Latency
- **These factors impact**
  - – Processor workload
  - – Memory usage
  - – Storage needs
  - – Inter-node communications
- **Corresponding server characteristics are**
  - – CPU power and core count
  - – RAM size
  - – Disk size, speed, and count
  - – Network architecture (whole data center)

Best Practices for HW Selection

- **Variability of mix makes it very hard to state firm rules**
- **Best practice: Do a pilot program, with quantitative studies of cluster behavior**
  - Start small and gain experience
  - Use Monitoring to measure actual workloads per component
  - Measure masters and slaves separately
  - Identify saturation of the components
  - Tweak and re-measure, get familiar with the trade-offs
- **Remember:**
  - Hadoop was designed to run well on commodity servers and disks
  - Heterogeneous clusters are fine; so as long as your pilot cluster is no more than 10% of the whole, you can merge it in without loss of investment.
- **It is common for the workload to change as an organization's expertise evolves.**
  - Expect to do new pilots from time to time.

Reminder: HDFS Diagram



Namespace Metadata & Journal

Backup Namenode

Namespace State

Block Map

Namenode

Block ID ➔ Block Locations

Hierarchal Namespace
File Name ➔ BlockIDs

Heartbeats & Block Reports

Datanodes

Block ID ➔ Data

Horizontally Scale IO and Storage

Starting Points

- **Typical Hadoop cluster:**
  - One or a few master nodes (HDFS Namenode, MapReduce JobTracker, HBase Master)
    - May co-deploy on a single shared node in low-load clusters, or each have a separate node in high-load clusters.
    - If Backup Namenode is used, it must have separate node
  - Many slave nodes (HDFS DataNodes, MapReduce TaskTrackers, HBase RegionServers)
    - Datanode, TaskTracker, and RegionServer services are usually co-deployed on the same slave nodes.
  - Three servers will run Zookeeper for HBase coordination
    - Each needs 1GB of memory
    - Common to co-deploy with any three slave nodes
  - At least one Gateway server configured with Client software
  - Monitoring and Database servers as preferred for the site

Starting Points (cont.)

- **Typical POC1 hardware**
  - – Single rack with top-of-rack Ethernet switch
  - – 10-20 rack-mount 1U servers
- **Slave nodes:**
  - – Dual quad-core CPUs
  - – 8-24 GB of RAM
  - – 4-6 disk drives of 1 or 2 TB capacity
  - – Dual 1 or 10 Gbps NICs and top-of-rack switch, depending on expected network architecture
- **Master node selection:**
  - – Single shared node, unless you know in advance that Map/Reduce and HBase will both be simultaneously heavily used.
  - – Plenty of CPU and RAM, e.g., 16 cores and 16-32 GB
  - – 2 local disk drives, possibly RAIDed
  - – 1 or 2 shared network drives, preferably HA
- **Cloud alternative:**
  - – Cluster of AWS m1.xlarge or equivalent

Starting Points (cont.)

- **POC2 or small production cluster**
  - Expand the initial test cluster
  - 50 nodes in 2 racks
  - May separate some services previously co-deployed, based on load
  - Secondary Namenode
  - 12 disks per Datanode
  - Start exploring the impact of network architecture choices
  - Grow the variety and specificity of jobs and queries in the job mix
  - Server RAM and CPU may need to grow based on load analysis

- **It takes a while to grow into a thousand-node cluster**
  - Need the data flow, of course
  - Also need the expertise in the Data Analysis or BI team to generate the jobs and queries, and interpret the results
  - Training is advised.

Pilot: How to measure saturation?

- **CPU usage**
- **Disk IOPs and throughput**
- **Network packet counts and throughput**
  - Monitor with tools like Nagios, Ganglia, or other performance monitoring tools available in your data center and network.
  - Monitor while running the actual kinds of query or analysis jobs that are of interest to your team.
  - Scale down the size of the data set proportionally to the size of the pilot cluster.

- **RAM**
  - Note that JVMs typically expand to fill all allocated space
  - So RAM usage, per se, is not an important metric
  - Observe whether swapping occurs – JVM memory size may also be tuned
  - Frequency of stop-the-world GC events are critical – You may need to scan logs to find these events – tune the GC parameters
  - HBase is more memory-intensive on the Datanodes than typical Map/Reduce jobs. Consider providing more RAM for heavy use of HBase.

Disks – general comments

- **Recurring costs of Hadoop nodes (power and a/c) are highly related to number of physical drives**
- **Many types of Map/Reduce jobs may not be limited by spindle count.**
  - Unless monitoring shows disk IOPs or bandwidth saturation, plan to buy the largest disks available rather than the fastest.
  - Spindles do buy more seeks per second.
  - Obviously, some types of jobs do smaller amounts of processing per unit of data, and generate modest amounts of inter-node communication. These jobs will be more sensitive to disk performance and spindle count.
  - Let your pilot project be your guide.
- **Many sites accumulate a great deal of data over time, but only use a fraction of it in the more common jobs. Therefore *active* storage may only be a few percent of total storage.**
  - If a site does not expect to accumulate data, then smaller, faster drives may make sense.

Network Architecture – critical

- **Bandwidth between racks (over-subscription ratio) is very important to Hadoop job performance**
  - Want it between 1:1 and 1:5. 1:10 is max.
  - 10Gbit within rack and only 2x10Gbit uplink doesn't make sense
  - Assume Map/Reduce shuffle randomly uses all-to-all communication
  - And HDFS replication of Reduce results writes cross-rack *again*

Job Tuning

- **Of course you can change the hardware choices to meet your job needs**
- **But you can also tune the jobs to better match your hardware**
- **The way a job is coded or job data is represented, can have large impact on resource balance:**
  - resource cost can be shifted between disk IOPS and CPU by choice of compression scheme or parsing format
  - per-node CPU and disk activity can be traded for inter-node bandwidth, depending on the Map/Reduce strategy.
- **If you have an equal number of jobs that saturate each resource category (“balanced profile”), consider tuning the jobs.**
- **Consider using Vaidya, a performance diagnostic tool for Map/Reduce jobs**

Tutorial

- **Agenda**
  - –Overview of HDFS architecture
  - –Hardware choices
  - –Rack topology awareness
  - –Federated metadata servers (Hadoop 2.0)
  - –Other Hadoop Improvements
  - –Calculating the probability of data loss

Rack topology awareness

- **Built-in feature recognizes the current norm in network topology**
  - Communication within a rack is typically cheaper than inter-rack
  - Entire racks can go out of service in certain network failure modes
- **Performance optimization**
  - Task assignment for efficiency
  - Clients can select “closer” nodes for I/O operations
- **Availability optimization**
  - Replication assures one of three replicas is off-rack
- **Topology representation is pluggable**
  - Default mapping strategy takes an arbitrary executable, such as a script
  - Namenode/JobTracker path specified by HDFS parameter `topology.script.file.name`
  - Takes list of host ids, returns list of rack ids
  - Results are cached, so efficiency is not critical

Tutorial

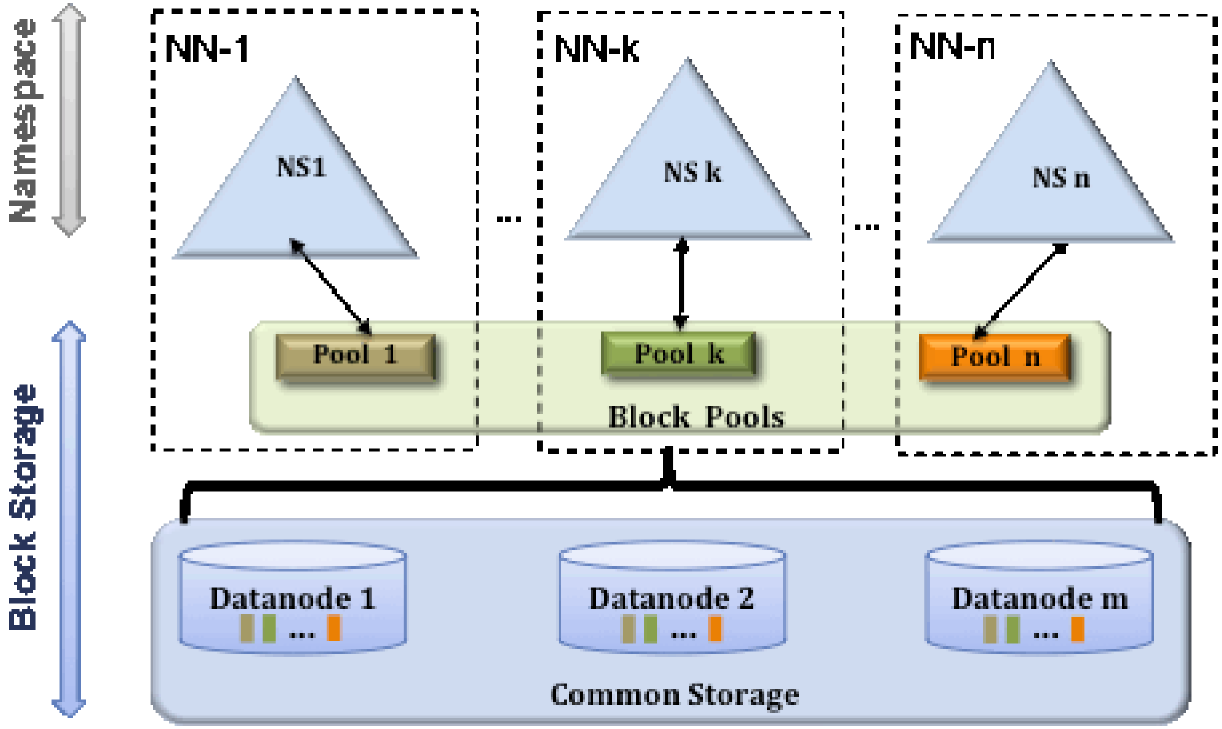
- **Agenda**
  - –Overview of HDFS architecture
  - –Hardware choices
  - –Rack topology awareness
  - –Federated metadata servers (Hadoop 2.0)
  - –Other Hadoop Improvements
  - –Calculating the probability of data loss

Federated metadata servers

- **In our experience, in production use, block size is typically adjusted to large values (64MB – 256MB)**
  - So most files are only 1 or 2 blocks long
  - Load on Namenode is then proportional to number of files, not the quantity of storage used.
- **Namenode with 64GB of memory and 24 cores can handle 100M files, with replication=3, with reasonable performance.**
  - 100M file inodes and path strings
  - 300M-600M block objects
  - Only about 10% of files in active data set for most jobs
- **Some sites are experimenting with even larger VMs, 128GB+**
  - Concern is robustness of JVM at such sizes, especially GC implementations
- **Alternative: Federated Namespaces and Nameservers**

HDFS Federation in v2.0

- **Improved scalability and isolation**
- **Clear separation of Namespace and Block Storage**



HDFS Federation in v2.0 (cont.)

- Allows the namespace to scale horizontally
- Partially alleviates the SPOF problem
- Leaves it up to users to partition the namespace
- Took opportunity to improve the architecture by separating namespace management from block management. Now have block pool management layer.
- Datanodes can participate in multiple block pools, which are separately maintained and reported, so can balance across the whole system – not partitioned set of datanodes with "LUN" problem.
- Block pools are access-isolated.

Tutorial

- **Agenda**
  - –Overview of HDFS architecture
  - –Hardware choices
  - –Rack topology awareness
  - –Federated metadata servers (Hadoop 2.0)
  - –Other Hadoop Improvements
  - –Calculating the probability of data loss

Other Hadoop Improvements in v2.0

- **HA Namenode**
  - Solve the SPOF
  - See Symposium talk tomorrow!

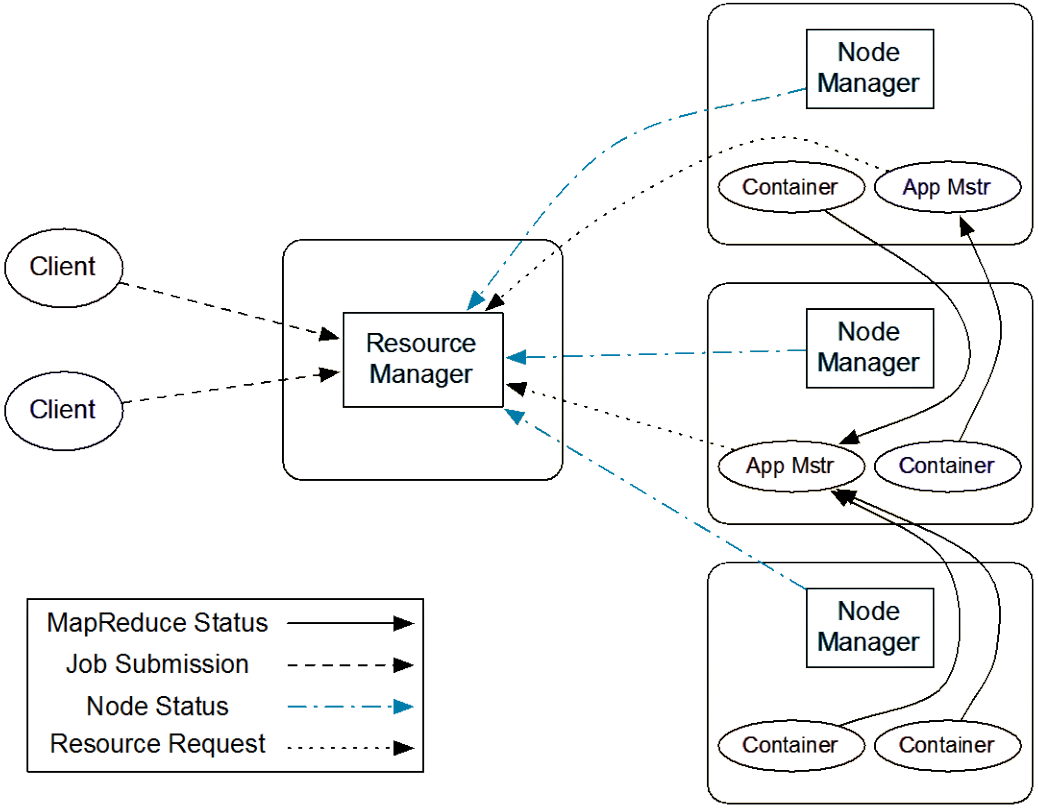
- **MR2 / YARN**

- **Compatible Wire Protocols**
  - On the way to rolling upgrades

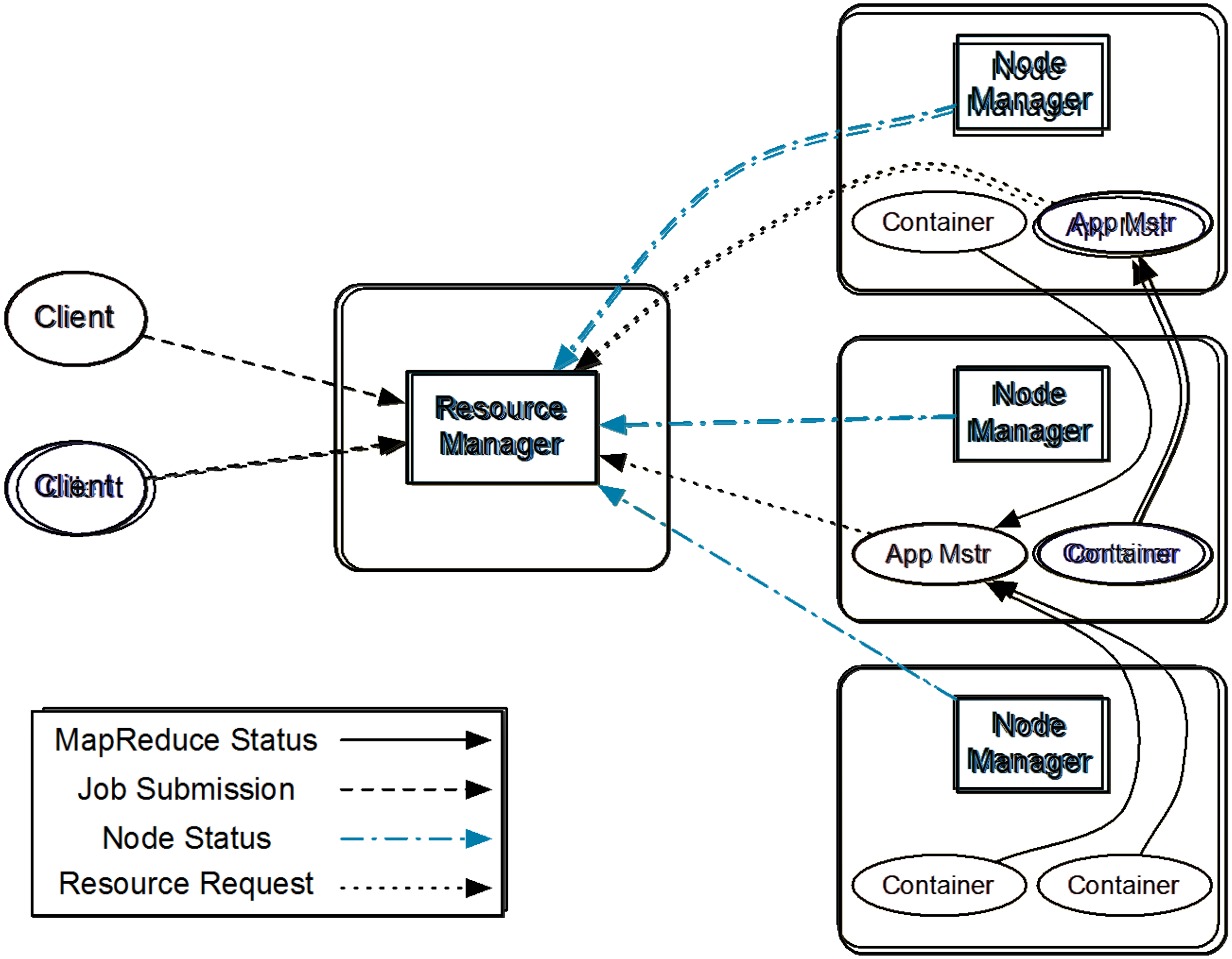
- **Performance**
  - Including big HDFS Write pipeline improvements for HBase

MapReduce2 - YARN

- NextGen Hadoop Data Processing Framework
- Support MR and other paradigms



YARN


Node Manager
Container
App Mstr
Client
Resource Manager
Client
Node Manager
App Mstr
Container
Node Manager
Container
Container
MapReduce Status
Job Submission
Node Status
Resource Request

Performance

- 2x+ across the board
- HDFS read/write
  - CRC32
  - fadvise
  - Shortcut for local reads
- MapReduce
  - Improvements from Terasort record experience (Owen/Arun, 2009)
  - Shuffle 30%+
  - Small Jobs optimization

Tutorial

- **Agenda**
  - –Overview of HDFS architecture
  - –Hardware choices
  - –Rack topology awareness
  - –Federated metadata servers (Hadoop 2.0)
  - –Other Hadoop Improvements
  - –Calculating the probability of data loss
    - –Analysis methodology developed by Rob Chansler of Yahoo!

Probability of Data Loss (1)

- **Suppose our cluster has 4000 nodes, with 12TB each.**
- **If block size is set to 128MB, there are** 384,000,000 **block replicas,**
- **Or 128,000,000 unique blocks, or** 85,333,333 **files (at ~1.5 blocks/file).**

- **Suppose we have a failure of about 1 node per day.**
- **If a node fails,** 96,000 **replicas are immediately unavailable, and the same number of blocks have only 2 replicas remaining.**
- **Each other node has an average of 48 of *one or the* other of those replicas that need re-replication.**
- **So if a second node fails, ~48 blocks will have only one replica remaining available.**

- **Now if a third node fails, the probability that it has the only remaining replica of one of those 48 particular blocks is:** 0.000250125
- **So the probability of losing all replicas of at least one block, in case of 3 simultaneous node failures, is** 0.011938669**, or a little over 1%**

Probability of Data Loss (2)

- **The cool thing is, that even with complete loss of a whole 12TB node, and only allowing 1Mbps of bandwidth per node for re-replication operations, it takes less than an hour (about 50 minutes) to fully re-create 3x redundancy for all blocks in the system.**
- **So 3 nodes would have to fail within a 2 hour time window, in order to have a 1% chance of losing one block.**
- **In practice, with 4000 node clusters in production, we see entire nodes re-replicated in aprx. 2 minutes (25 Mbps per node, effective), not 50 minutes.**

- **More importantly, since v0.20.204, Datanodes have been able to withstand losses of single disk drives without bringing down the entire node. So the loss would be 1TB, not 12TB, the number of at-risk replicas per other node is just 4, not 48, and the number of at-risk replicas per drive in those other nodes is 0.33.**
- **Re-replication time at 1Mbps per node, would be only 22 seconds; so 3 disk drives would have to fail simultaneously within less than a minute, to have an infinitesimal chance of losing one block.**

Probability of Data Loss (3)

- **Even cooler: As you scale up the number of nodes in the cluster, the probability of data loss DECREASES! How often does that happen?**

Credits

For major contributions to Hadoop technology, and help with this presentation:

- **Sanjay Radia and Suresh Srinivas, Hortonworks**
  - Architect and Team Lead, HDFS
  - HA and Federation
- **Owen O’Malley, Hortonworks**
  - Hadoop lead Architect
  - Security, Map/Reduce, system configuration
- **Arun Murthy, Hortonworks**
  - Architect and Team Lead, Map/Reduce
  - M/R2, YARN, etc.
- **Rob Chansler, Yahoo!**
  - Team Lead, HDFS
  - Analysis of Probability of Data Loss

Help getting started

- **Apache Hadoop Projects**
  - http://hadoop.apache.org/
  - http://wiki.apache.org/hadoop/
- **Apache Hadoop Email lists:**
  - common-user@hadoop.apache.org
  - hdfs-user@hadoop.apache.org
  - mapreduce-user@hadoop.apache.org
- **O’Reilly Books**
  - Hadoop, The Definitive Guide
  - HBase, The Definitive Guide
- **Hortonworks, Inc.**
  - Installable Data Platform distribution (100% OSS, conforming to Apache releases)
    - http://hortonworks.com/technology/techpreview/
  - Training and Certification programs
    - http://hortonworks.com/training/
- **Hadoop Summit 2012 (June 13-14, San Jose)**
    - http://hadoopsummit.org/

Thanks for Listening!

Questions?