Aaron T. Myers
Software Engineer, Cloudera
September 7th, 2011

A Little About Me

- atm@cloudera.com
- Hadoop Common and HDFS Committer
- Software Engineer at Cloudera
- Studied CS at Brown (Hi, Panda!)
- Primarily focused on HDFS architecture and Security for Hadoop ecosystem

Presentation Outline

1. Motivations
   - A few use cases
2. What is Hadoop?
   - What it's based on
   - Architecture
3. What isn't Hadoop?
   - Enter: CDH
4. Future directions
5. Discussion

1. Motivations

Use 1: Search Indexing

- eBay's Cassini project
  - Completely rearchitect eBay's search infrastructure
  - ~100 million active users
  - Millions of items listed daily
  - Tens of bids on most items
  - Semi-structured data
    - Product descriptions, feedback, etc.
  - Customized recommendations
  - This is what Hadoop was built for

Use 2: Matchmaking

- eHarmony needs to figure out good matches for its users as soon as possible after they sign up
  - Very short attention spans
  - Need to get it right the first time
- Millions of users provide different sets of information
- Matchmaking really boils down to graph partitioning
- Not quite what Hadoop was designed for, but it'll work

Use 3: Finance

- Hadoop in use at ***big*** financial companies
  - Visa, JP Morgan Chase, B of A, etc.
- Fraud detection
  - Analyze millions of PoS transactions all the time
  - Use machine learning to identify salient features
- Risk modeling
  - Credit history is a very poor formula
  - Look at *every* financial decision a person has ever made

Commonalities in Hadoop Uses

- A lot of data
  - User logs, click stream, transactions, user-generated content
  - 100s of TBs or PBs, easily
- Not necessarily clear which data will be useful
  - Store it all, worry about analysis later (major shift)
  - Often easier to identify schema after the fact
  - Storage in Hadoop is dirt cheap
- Need to keep scaling
  - POC with 10 nodes, scale to 100, 1,000, etc.

2. What is Hadoop?

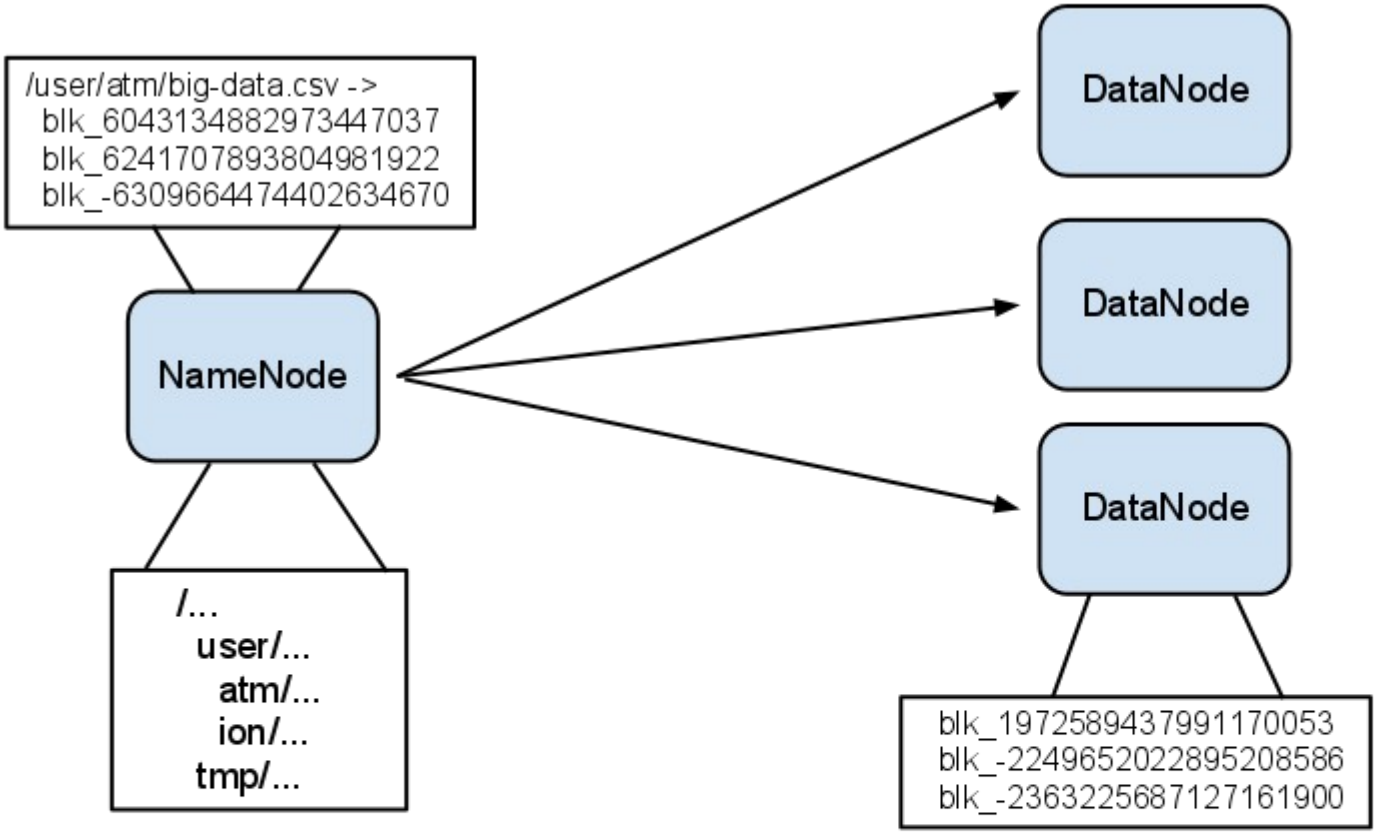
Hadoop is...

- Software for large-scale storage and processing using commodity machines
- Distributed, fault-tolerant, persistent storage
  - HDFS
- Distributed, fault-tolerant compute
  - Map-Reduce
- When taken together, very powerful
- 100% Apache-licensed OSS, developed at the ASF

HDFS

- Hadoop Distributed File System
- Modeled after Google's GFS
- Files made up of potentially many large blocks
  - Default block size 64MB
- A single NameNode (NN) stores FS metadata
- Many DataNodes (DNs) store all the blocks
- Can be configured to be aware of rack placement

HDFS (Continued)



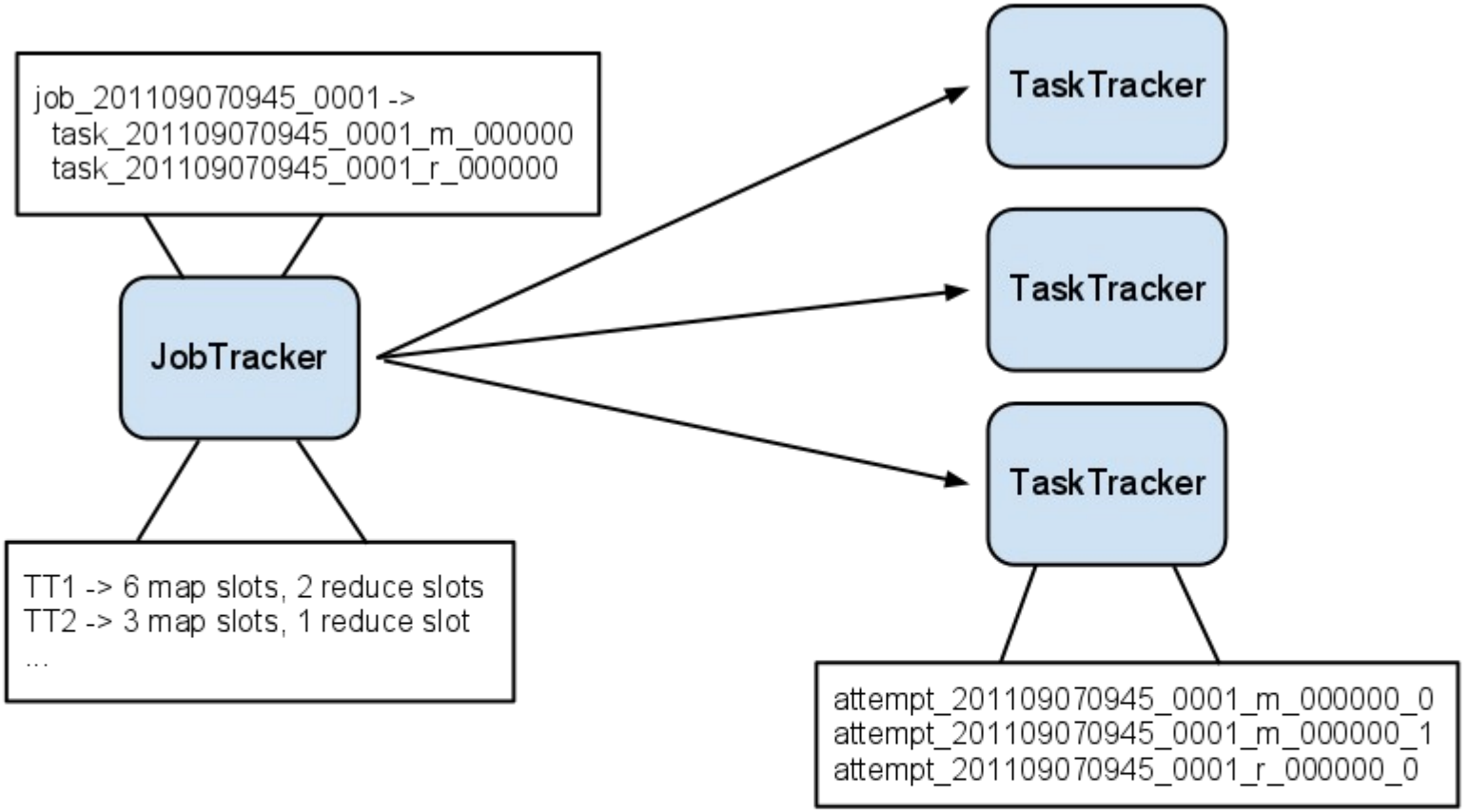
HDFS (Continued)

- For durability/locality, every block replicated
  - By default, replicated 3 times
  - Can be configured per-file by clients
  - If cluster is rack-aware, place one replica on one rack, two on another
- Only metadata goes to the NN
  - Clients read/write directly from/to the DN(s) which hold the blocks they want
- All blocks are checksummed

Map-Reduce

- Distributed processing framework for Hadoop
- Modeled after Google's MapReduce
- A “job” is made up of many “tasks” which are made up of potentially several “task attempts”
- A single JobTracker (JT) handles resource assignment and tracking job life cycle (tasks)
- Many TaskTrackers (TTs) actually execute task attempts

Map-Reduce (Continued)



Map-Reduce + HDFS

- Tasks correspond roughly 1:1 with blocks
- So, collocate the TTs and DNs
- When selecting where to run a task, the JT asks the NN which nodes have the block(s) the task is going to operate on
  - Ship the code to the data
- Node-local tasks run faster than rack-local
- Rack-local tasks run **much** faster than not

3. What isn't Hadoop?

Hadoop is great, but...

- Map-Reduce is a very low-level paradigm
  - Implementing a join across two data sets is a common operation, and kind of a pain to do
- A Map-Reduce job does not an analysis make
  - Generally need a series, perhaps a DAG, of MR jobs to accomplish a business goal
- Data doesn't usually get written directly to HDFS
  - Data ingest is a serious problem
  - Data exists as log files, custom event streams, in traditional DBMSes, etc.

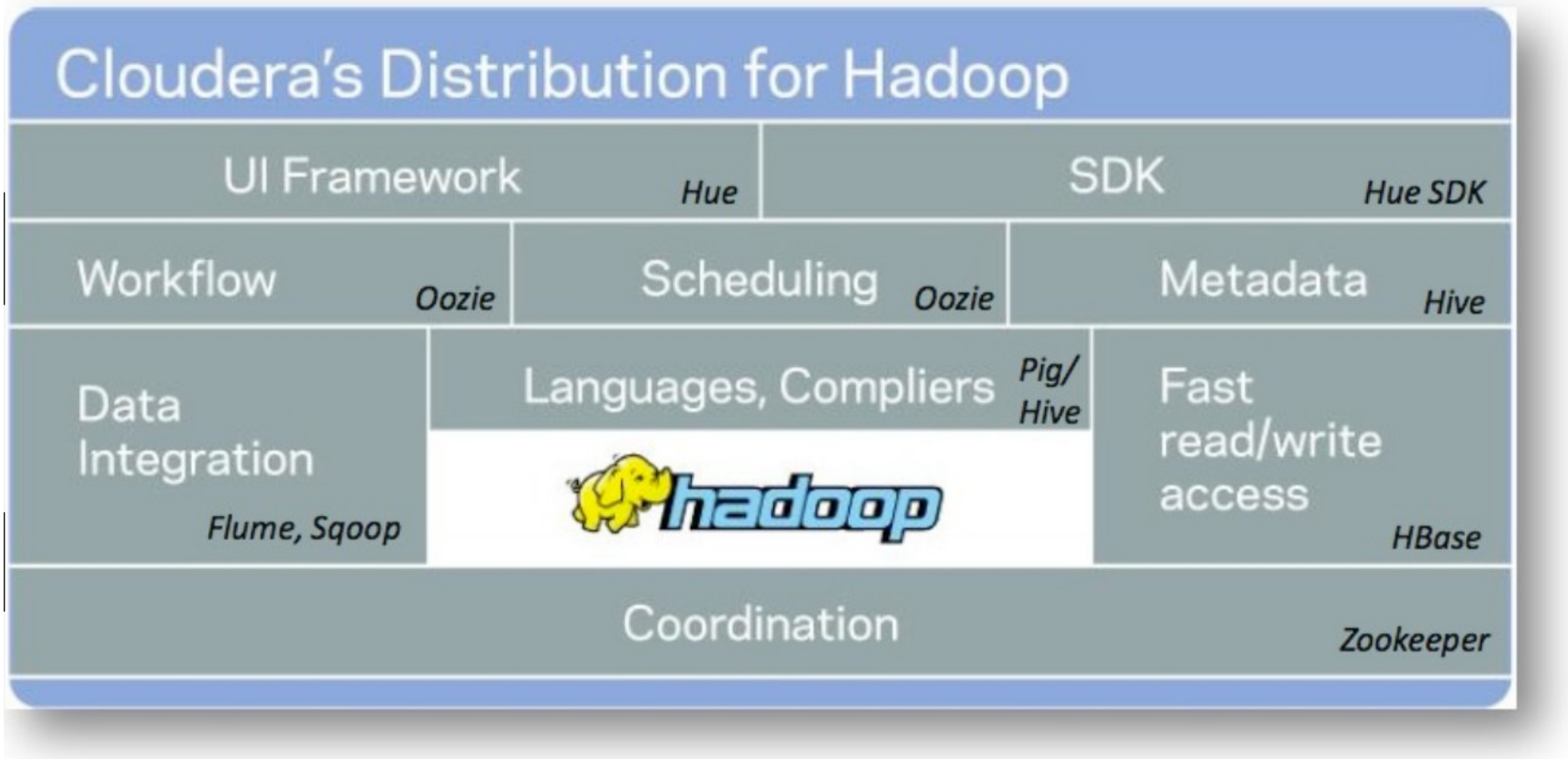
Hadoop is great, but...

- HDFS stores arbitrary files
  - Selecting a good storage format is an exercise worth having
- HDFS random access is slow
  - Optimized for large, streaming reads/writes
  - No support for random writes - append-only
  - If you want to serve small, random reads/writes in real-time, you need something else
- Hadoop has no GUI
  - Well, nothing my mom would find useful, anyway

Hadoop is like the Linux Kernel

- Everyone runs it, few care that they do
- You need a lot more than the kernel
- Sure, you could write programs to make system calls...

Enter: CDH


Cloudera's Distribution for Hadoop
UI Framework
Hue
SDK
Hue SDK
Workflow
Oozie
Scheduling
Oozie
Metadata
Hive
Data Integration
Flume, Sqoop
Languages, Compliers
Pig/ Hive
hadoop
Fast read/write access
HBase
Coordination
Zookeeper

Higher-level languages: Hive

- Uses a SQL-like language called HiveQL
  - SELECT * FROM Users, Pages WHERE...
- Also includes metadata storage, separate from HDFS
  - The Hive “metastore”
- A single Hive query gets translated into potentially several MR jobs
- Query plan capable of taking advantage of features of Hadoop

Higher-level languages: Pig

- DSL which gets compiled into straight MR
- Abstracts away the complexity of writing MR
- Oriented toward those who are already familiar with programming languages

Pig: An Example

top_5.pig

```
users = load 'users.csv' as (username: chararray, age: int);
users_1825 = filter users by age >= 18 and age <= 25;

pages = load 'pages.csv' as (username: chararray, url: chararray);

joined = join users_1825 by username, pages by username;
grouped = group joined by url;
summed = foreach grouped generate group as url, COUNT(joined) AS views;
sorted = order summed by views desc;
top_5 = limit sorted 5;

store top_5 into 'top_5_sites.csv';
```

Source: Dmitriy Royaboy – Hadoop at Twitter

Pig: An Example

Source: Dmitriy Royaboy – Hadoop at Twitter

Workflow: Oozie

- When just one MR job won't do
- Lets one specify a DAG of steps to perform to complete a task
  - Run this Hive query, then run these MR jobs, then wait for this data to show up in the FS...
- Allows periodic scheduling of jobs, or triggered jobs

Data Ingest: Flume

- Flexible, reliable scalable system for collecting streaming data
- Flume “agents” at data sources
- Flume “sinks” at data destination(s)
- Flume “collectors” in between
- API for custom sources/sinks
- Lots of out-of-the-box sources/sinks
  - e.g. arbitrary log tailing, syslog events

Data Ingest: Sqoop

- Tool for efficiently transferring data in bulk between Hadoop and structured data stores
- Usually run an MR job to do bulk import/load/export
  - Great way to take down a DB machine
- Provides a pluggable “connector” mechanism to allow Sqoop to work with arbitrary DBs

Storage Format: Avro

- Efficient data (de)serialization system
  - Where efficient = fast, compact
  - Think Thrift, Protobufs, etc.
- Allows for rich data structures
- Allows for flexible, evolving schemas
- A splittable, compressible container file format
- Libraries for reading/writing from most popular programming languages

Fast random access: HBase

- Distributed, versioned, sparse, column-oriented, multidimensional, sorted map
  - Modeled after Google's BigTable
  - Supports billions of rows X millions of columns
- On each DataNode in a Hadoop cluster, also run an HBase RegionServer
  - RS serves HBase regions stored in HDFS on that node
  - Also acts as a big cache

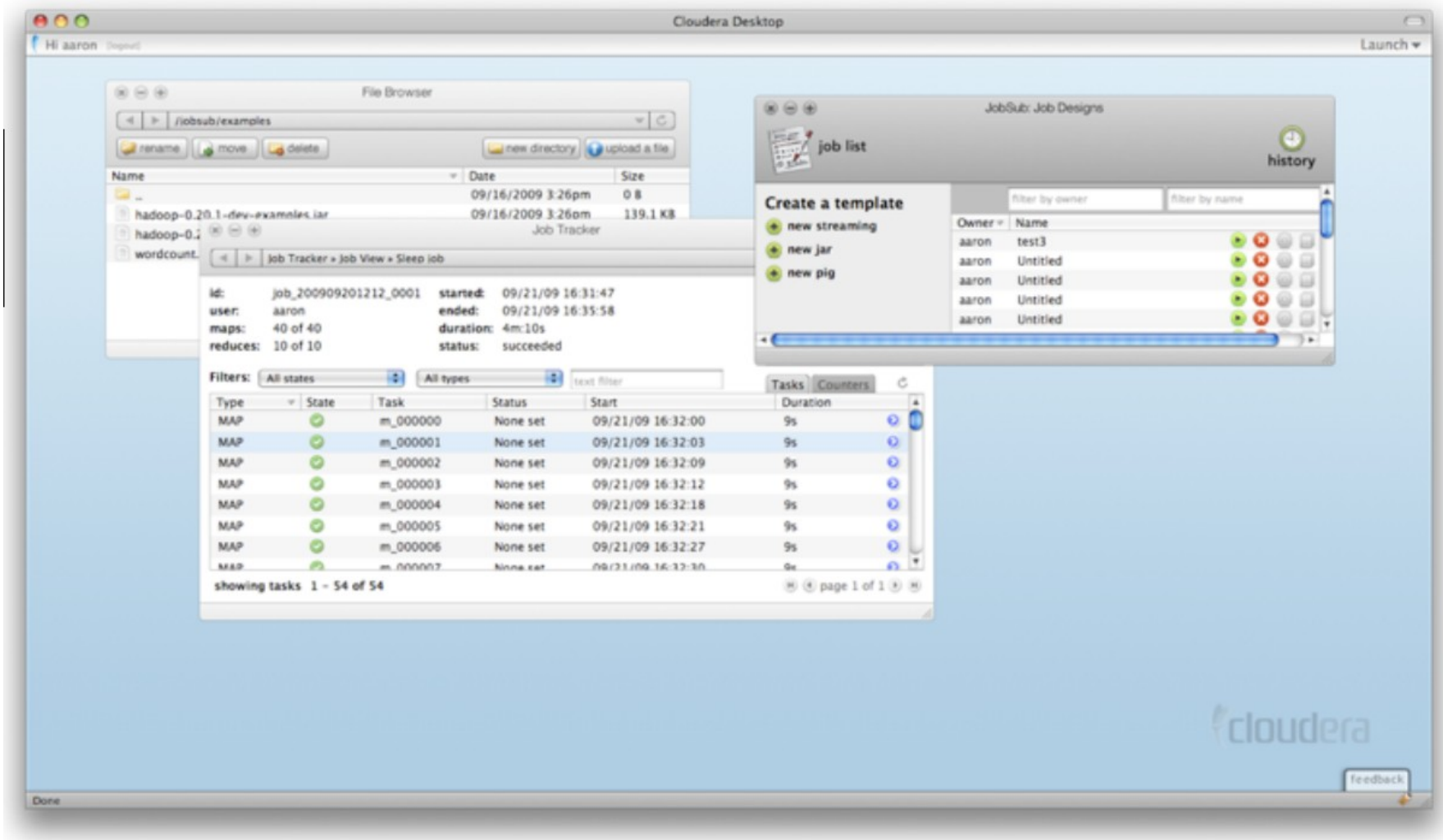
Mutability of records: HBase

- HDFS is append-only
  - So how do we update individual “records” ?
  - HBase to the rescue
- To write, append an update with a higher version
- On read, only read the highest version
  - Or, exploit this fact to see a historical view of some value

User Interface: Hue

- A decent GUI is critical for mainstream adoption
- Should be able to run jobs, create Hive queries, view files, create Oozie workflows, etc.

User Interface: Hue



4. Future Directions

(Read: potential class projects)

Alternative Processing Frameworks

- Lots of algos can be adapted to run on MR
  - But some are tricky
- Hadoop trunk now has re-architected processing system
  - Theoretically supports alternative computing paradigms beside MR, e.g. BSP, Spark, Pregel, etc.
  - But, there are none written yet!
  - Matei is working on Spark

More Resource Awareness

- Until recently, MR scheduling in Hadoop was slot-based
  - Distinct slots for map vs. reduce
  - Very difficult to get full utilization
- Hadoop trunk now has “YARN”
  - “Yet Another Resource Negotiator”
  - Theoretically supports using other resources to affect scheduling decisions
  - But, only memory is implemented at the moment

Real-time querying

- Hive and Pig have interactive modes
  - But, the very fastest Hadoop job takes ~30 seconds to run (not very interactive)
- Much of the time there are spare resources in a Hadoop cluster
  - It would be great if these could somehow be utilized to run queries we expect to be quick

Track Disks, Not Nodes

- For durability, data is replicated to different nodes on different racks
- Batches of hardware tend to fail at around the same time (or failure rate goes up dramatically)
- It would be great to include drive batch information in replication policy

Separate Block Map / Namespace

- Two functions of JT were recently decoupled in MR
- The two functions of the NN could likely also be decoupled
- Allow for greater scalability, maybe performance

Consistent HDFS snapshots

- Data in a Hadoop cluster is usually so large as to be infeasible to create backups
  - Unless you happen to have a spare cluster
  - Even if you did, it's non-trivial to ship backups of a changing file system elsewhere
- Ideally HDFS would support creating moment-in-time consistent snapshots of the FS
  - Perhaps made easier by the fact that it's append-only

HDFS Event Notifications

- HDFS equivalent of Linux inotify(7)
- Would make a lot of things easier
  - Oozie triggers
  - Adding new Hive tables when data arrives
  - ...

Security, everywhere

- Hadoop only recently added strong authentication
  - Before this, no one in Hadoop ecosystem bothered much with authorization
- Still very, very early days of authorization mechanisms

5. Discussion

(Ask me questions)

Email: atm@cloudera.com
Twitter: @atm