Hadoop and Pig @Twitter
Kevin Weil -- @kevinweil
Analytics Lead, Twitter
Agenda

- Hadoop Overview
- Pig: Rapid Learning Over Big Data
- Data-Driven Products
- Hadoop/Pig and Analytics
My Background

- Mathematics and Physics at Harvard, Physics at Stanford
- **Tropos Networks** (city-wide wireless): mesh routing algorithms, GBs of data
- **Cooliris** (web media): Hadoop and Pig for analytics, TBs of data
- **Twitter**: Hadoop, Pig, HBase, Cassandra, machine learning, visualization, social graph analysis, soon to be PBs data
Agenda

- Hadoop Overview
- Pig: Rapid Learning Over Big Data
- Data-Driven Products
- Hadoop/Pig and Analytics
Data is Getting Big

- NYSE: 1 TB/day
- Facebook: 20+ TB compressed/day
- CERN/LHC: 40 TB/day (15 PB/year)
- And growth is accelerating
- Need multiple machines, horizontal scalability
Hadoop

- Distributed file system (*hard to store a PB*)
- Fault-tolerant, handles replication, node failure, etc
- MapReduce-based parallel computation (*even harder to process a PB*)
- Generic key-value based computation interface allows for wide applicability
Hadoop

- **Open source**: top-level Apache project
- **Scalable**: Y! has a 4000-node cluster
- **Powerful**: sorted a TB of random integers in 62 seconds

- **Easy Packaging**: Cloudera RPMs, DEBs
MapReduce Workflow

- Challenge: how many tweets per user, given tweets table?
- Input: key=row, value=tweet info
- Map: output key=user_id, value=1
- Shuffle: sort by user_id
- Reduce: for each user_id, sum
- Output: user_id, tweet count
- With 2x machines, runs 2x faster
MapReduce Workflow

- Challenge: how many tweets per user, given tweets table?
- Input: key=row, value=tweet info
- Map: output key=user_id, value=1
- Shuffle: sort by user_id
- Reduce: for each user_id, sum
- Output: user_id, tweet count
- With 2x machines, runs 2x faster
MapReduce Workflow

- Challenge: how many tweets per user, given tweets table?
- Input: key=row, value=tweet info
- Map: output key=user_id, value=1
- Shuffle: sort by user_id
- Reduce: for each user_id, sum
- Output: user_id, tweet count
- With 2x machines, runs 2x faster
MapReduce Workflow

- Challenge: how many tweets per user, given tweets table?
- Input: key=row, value=tweet info
- Map: output key=user_id, value=1
- Shuffle: sort by user_id
- Reduce: for each user_id, sum
- Output: user_id, tweet count
- With 2x machines, runs 2x faster
MapReduce Workflow

- Challenge: how many tweets per user, given tweets table?
- Input: key=row, value=tweet info
- Map: output key=user_id, value=1
- Shuffle: sort by user_id
- Reduce: for each user_id, sum
- Output: user_id, tweet count
- With 2x machines, runs 2x faster
MapReduce Workflow

- Challenge: how many tweets per user, given tweets table?
- Input: key=row, value=tweet info
- Map: output key=user_id, value=1
- Shuffle: sort by user_id
- Reduce: for each user_id, sum
- Output: user_id, tweet count
- With 2x machines, runs 2x faster
MapReduce Workflow

- Challenge: how many tweets per user, given tweets table?
- Input: key=row, value=tweet info
- Map: output key=user_id, value=1
- Shuffle: sort by user_id
- Reduce: for each user_id, sum
- Output: user_id, tweet count
- With 2x machines, runs 2x faster
But...

- Analysis typically in Java
- Single-input, two-stage data flow is rigid
- Projections, filters: custom code
- Joins are lengthy, error-prone
- Hard to manage n-stage jobs
- Exploration requires compilation!
Agenda

- Hadoop Overview
- Pig: Rapid Learning Over Big Data
- Data-Driven Products
- Hadoop/Pig and Analytics
Enter Pig

- High level language
- Transformations on sets of records
- Process data one step at a time
- Easier than SQL?

- Top-level Apache project
Why Pig?

- Because I bet you can read the following script.
A Real Pig Script

```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';
```
Now, just for fun...

- The same calculation in vanilla MapReduce
No, seriously.
Pig Democratizes Large-scale Data Analysis

- The Pig version is:
  - 5% of the code
  - 5% of the development time
  - Within 25% of the execution time
  - Readable, reusable
One Thing I’ve Learned

- It’s easy to answer questions
- It’s hard to ask the right questions
- Value the system that promotes innovation and iteration

Friday, July 23, 2010
Agenda

- Hadoop Overview
- Pig: Rapid Learning Over Big Data
- Data-Driven Products
- Hadoop/Pig and Analytics
MySQL, MySQL, MySQL

- We all start there.
- But MySQL is not built for analysis.
- `select count(*) from users? Maybe.`
- `select count(*) from tweets? Uh...`
- Imagine joining them.
- And grouping.
- Then sorting.
Non-Pig Hadoop at Twitter

- Data Sink via Scribe
- Distributed Grep
- A few performance-critical, simple jobs
- People Search
People Search?

- First real product built with Hadoop
- “Find People”
- Old version: offline process on a single node
- New version: complex graph calculations, hit internal network services, custom indexing
  - Faster, more reliable, more observable
People Search

- Import user data into HBase
- Periodic MapReduce job reading from HBase
  - Hits FlockDB, other internal services in mapper
- Custom partitioning
- Data sucked across to sharded, replicated, horizontally scalable, in-memory, low-latency Scala service
  - Build a trie, do case folding/normalization, suggestions, etc
Agenda

- Hadoop Overview
- Pig: Rapid Learning Over Big Data
- Data-Driven Products
- Hadoop/Pig and Analytics
Order of Operations

- Counting
- Correlating
- Research/Algorithmic Learning
Counting

‣ How many requests per day?
‣ What’s the average latency? 95% latency?
‣ What’s the response code distribution?
‣ How many searches per day? Unique users?
‣ What’s the geographic breakdown of requests?
‣ How many tweets? From what clients?
‣ How many signups? Profile completeness?
‣ How many SMS notifications did we send?
Correlating

› How does usage differ for mobile users?
› ... for desktop client users (Tweetdeck, etc)?
› Cohort analyses
› What services fail at the same time?
› What features get users hooked?
› What do successful users do often?
› How does tweet volume change over time?
Research

› What can we infer from a user’s tweets?
› ... from the tweets of their followers? followees?
› What features tend to get a tweet retweeted?
› ... and what influences the retweet tree depth?
› Duplicate detection, language detection
› What graph structures lead to increased usage?
› Sentiment analysis, entity extraction
› User reputation
If We Had More Time...

- HBase
- LZO compression and Hadoop
- Protocol buffers
- Our open source: hadoop-lzo, elephant-bird
- Analytics and Cassandra
Questions?

Follow me at twitter.com/kevinweil