Hadoop and Cascading

Christopher Curtin

About Me

- 19+ years in Technology
- Background in Factory Automation, Warehouse Management and Food Safety system development before Silverpop
- CTO of Silverpop
- Silverpop is the world's only provider of both email marketing and marketing automation solutions specifically tailored to the unique needs of B2C and B2B marketers.

Silverpop®

NOSQL

- SQL/RDBMS have their place
- Lots of things we couldn't do without one
- However …

Silverpop®

Not all data is a nail!

Silverpop®

To quote Don Brown

- “pig makes easy things really easy. cascading makes the hard stuff possible”

What is Map/Reduce?

- Pioneered by Google
- Parallel processing of large data sets across many computers
- Highly fault tolerant
- Splits work into two steps
  - Map
  - Reduce

Map

- Identifies what is in the input that you want to process
- Can be simple: occurrence of a word
- Can be difficult: evaluate each row and toss those older than 90 days or from IP Range 192.168.1.*
- Output is a list of name/value pairs
- Name and value do not have to be primitives

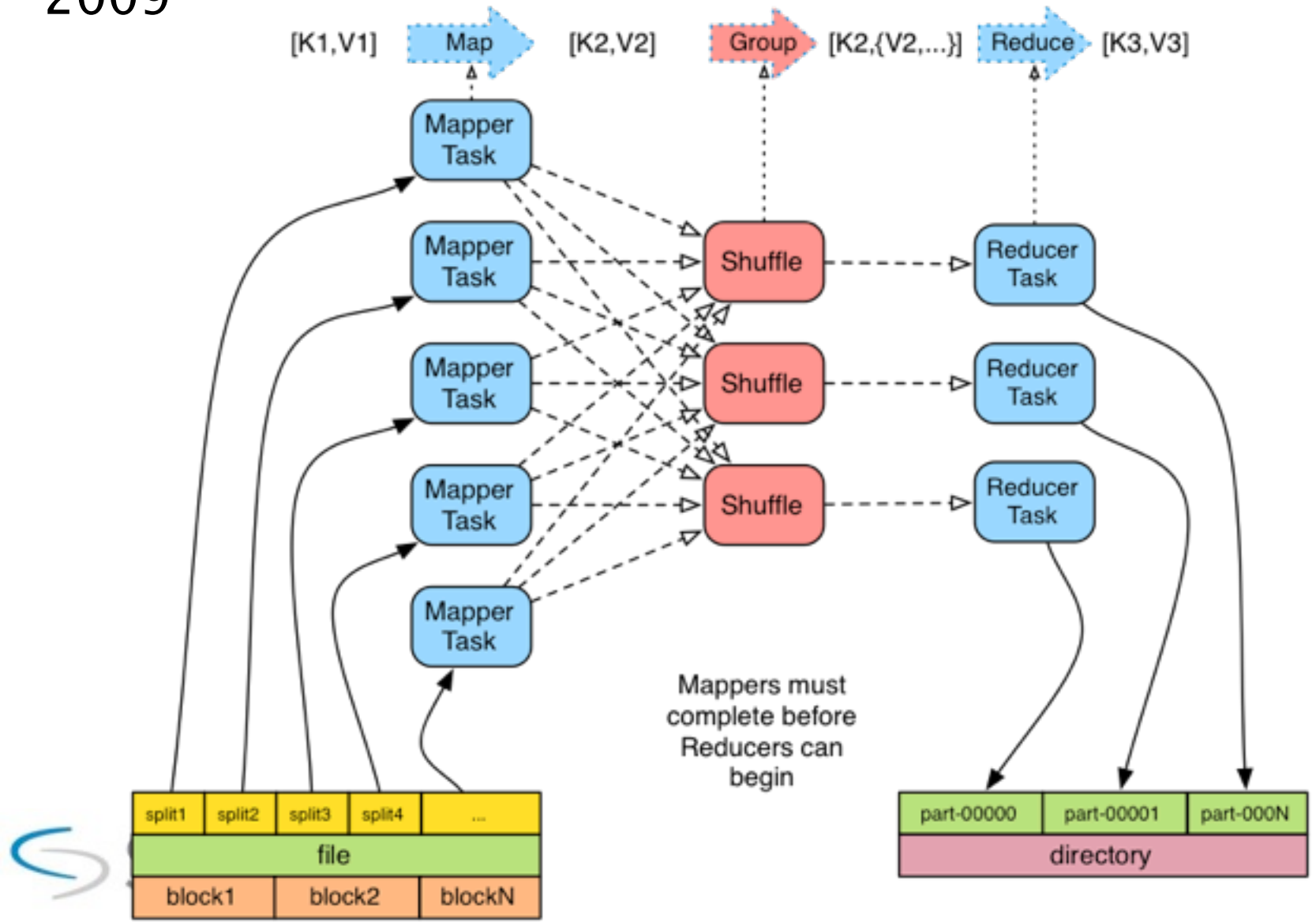
Reduce

- Takes the name/value pairs from the Map step and does something useful with them
- Map/Reduce Framework determines which Reduce instance to call for which Map values so a Reduce only ‘sees’ one set of ‘Name’ values
- Output is the ‘answer’ to the question
- Example: bytes streamed by IP address from Apache logs

Hadoop

- Apache's Map/Reduce framework
- Apache License
- Yahoo! Uses a version and they release their enhancements back to the community

Runtime Distribution © Concurrent 2009



Getting Started with Map/

- First challenge: real examples
- Second challenge: when to map and when to reduce?
- Third challenge: what if I need more than one of each? How to coordinate?
- Fourth challenge: non-trivial business logic

Cascading

- Open Source
- Puts a wrapper on top of Hadoop
- And so much more …

Main Concepts

- Tuple
- Operations
- Pipes
- Flows

Tuple

- A single ‘row’ of data being processed
- Each column is named
- Can access data by name or position

Operations

- Define what to do on the data
- “Relational”-like operations (Can I say that here?)
- Each – for each “tuple” in data do this to it
- Group – similar to a ‘group by’ in SQL
- CoGroup – joins of tuple streams together
- Every – for every key in the Group or CoGroup do this

Operations – advanced

- Each operations allow logic on the tuple, such a parsing dates, creating new attributes etc.
- Every operations allow you to iterate over the ‘group’ of tuples to do non-trivial operations.
- Both allow multiple operations in same function, so no nested function calls!

Silverpop®

Pipes

- Pipes tie Operations together
- Pipes can be thought of as ‘tuple streams’
- Pipes can be split, allowing parallel execution of Operations

Silverpop®

Example Operation

```
RowAggregator aggr = new RowAggregator(row);

Fields groupBy = new Fields
   (MetricColumnDefinition.RECIPIENT_ID_NAME);

Pipe formatPipe = new Each("reformat_“ new Fields
   ("line"), a_sentFile);

formatPipe = new GroupBy(formatPipe, groupBy);
formatPipe = new Every(formatPipe, Fields.ALL, aggr);
```

Silverpop®

Flows

- Flows are reusable combinations of Taps, Pipes and Operations
- Allows you to build library of functions
- Flows are where the Cascading scheduler comes into play

Cascading Scheduler

- Once the Flows and Cascades are defined, looks for dependencies
- When executed, tells Hadoop what Map, Reduce or Shuffle steps to take based on what Operations were used
- Knows what can be executed in parallel
- Knows when a step completes what other steps can execute

Dynamic Flow Creation

- Flows can be created at run time based on inputs.
- 5 input files one week, 10 the next, Java code creates 10 Flows instead of 5
- Group and Every don't care how many input Pipes

Silverpop®

Dynamic Tuple Definition

- Each operations on input Taps can parse text lines into different Fields
- So one source may have 5 inputs, another 10
- Each operations can used meta data to know how to parse
- Can write Each operations to output common Tuples
- Every operations can output new Tuples as well

Silverpop®

Mixing non-Hadoop code

- Cascading allows you to mix regular java between Flows in a Cascade
- So you can call out to databases, write intermediates to a file etc.
- We use it to load meta data about the columns in the source files
- Can be run via a daemon
  - Listen on a JMS queue
  - Schedule job based on job definition

Real Example

Real Example

- For the hundreds of mailings sent last year

Silverpop®

Real Example

- For the hundreds of mailings sent last year
- To millions of recipients

Silverpop®

Real Example

- For the hundreds of mailings sent last year
- To millions of recipients
- Show me who opened, how often

Silverpop®

Real Example

- For the hundreds of mailings sent last year
- To millions of recipients
- Show me who opened, how often
- Break it down by how long they have been a subscriber

Silverpop®

Real Example

- For the hundreds of mailings sent last year
- To millions of recipients
- Show me who opened, how often
- Break it down by how long they have been a subscriber
- And their Gender

Silverpop

Real Example

- For the hundreds of mailings sent last year
- To millions of recipients
- Show me who opened, how often
- Break it down by how long they have been a subscriber
- And their Gender
- And the number of times clicked on the offer

Silverpop®

RDBMS solution

- Lots of million+ row joins
- Lots of million+ row counts
- Temporary tables since we want multiple answers
- Lots of memory
- Lots of CPU and I/O
- $$ becomes bottleneck to adding more rows or more clients to same logic

Cascading Solution

- Let Hadoop parse input files
- Let Hadoop group all inputs by recipient's email
- Let Cascading call Every functions to look at all rows for a recipient and 'flatten' data (one row per recipient)
- Split 'flattened' data Pipes to process in parallel: time in list, gender, clicked on links
- Bandwidth to export data from RDBMS becomes bottleneck

Silverpop

Example Continued

- Adding a new 'pivot' means:
  - add a new Pipe to existing flow
  - Define the new Aggregation function if not something common
  - Defining the new output Tap
- Dynamic Flow creation allows me to do this in a function and configure 'outputs' from a well known set
  - Desired Results defined per job run

Silverpop

Another Example: ISP Bandwidth

- Operations would like to know what ISPs do we send the most email to?
- # of recipients can be obtained from the database easily
- Bandwidth can't

Bandwidth Example

- Daily load the logs from our MTA servers into Hadoop
- Run a Cascading job to aggregate by destination IP, # of messages, # of bytes
- Save daily into HDFS ‘intermediate’
- Weekly run a Cascading job against the daily files to determine weekly usage, top domains, change in domain ranking

Pros and Cons

- Pros
  - Mix java between map/reduce steps
  - Don't have to worry about when to map, when to reduce
  - Don't have to think about dependencies or how to process
  - Data definition can change on the fly
- Cons
  - Level above Hadoop – sometimes 'black magic'
  - Data must (should) be outside of database to get most concurrency

Resources

- Me: ccurtin@silverpop.com
  @ChrisCurtin
- Chris Wensel: @cwensel
- Web site: www.cascading.org
- Mailing list off website
- AWSome Atlanta Group: http://www.meetup.com/awsomeatlanta/
- O'Reilly Hadoop Book:
- http://oreilly.com/catalog/