

Moving Faster: Why Intent Media Chose Cascalog for Data Processing and Machine Learning

Kurt Schrader
May 20, 2014

Overview

- History of data processing at Intent Media
- “Hello World” in various data processing languages
- Cascalog overview
- The future

Who am I?

Kurt Schrader
CTO

kurt@intentmedia.com

Office: 646-358-1240

Mobile: 212-203-1314

180 Varick Street, Suite 936, New York, NY 10014

<intent>
MEDIA

[Change search](#)New York, NY (NYC) to San Francisco, CA (SFO)
San Francisco, CA (SFO) to New York, NY (NYC)Wed, May 21, 2014
Sat, May 24, 2014Flexible Dates?
[Search +/- 3 days](#)Don't Miss Out! Save up to \$525* by booking your Flight and Hotel together [Search now](#)

LOWEST PRICE

\$683.50

[Hide Matrix](#)

	US Airways	Multiple Airlines	American Airlines	Delta Air Lines	jetBlue	Virgin America	United Airlines	Sun Country Airlines
Non-stop	\$966.00	\$972.99	\$966.00	\$966.00	\$966.00	\$1,078.00	\$976.00	
1+ stops	\$683.50	\$793.99	\$917.00	\$977.00	\$1,133.00	\$970.00	\$980.00	\$1,008.00

Refine Results

[Select flight times](#)

Outbound

Take-off: 5:00 AM - 9:00 PM

[Show landing time](#)

Inbound

Take-off: Sat 12:15 AM - Sun 12:00 AM

[Show landing time](#)[Select stops](#)☒ Any☐ Non-stop[\\$966](#)☐ 1 stop[\\$683](#)[Select price range](#)☒ Any☐ \$683-\$850 (85)☐ \$850-\$970 (84)☐ \$970-\$980 (179)☐ \$980-\$1,020 (75)☐ \$1,020-\$1,440 (70)☐ \$1,440-\$2,000 (2)[Select airlines](#)☒ Any

Matching Results: 495

Sort by: [Lowest Price](#)Additional [baggage fees](#) may apply. Act Fast! Only [1 ticket](#) left at this price!

\$683.50

Total cost

[Select](#)☐ Select this departure

Leave Wed, May 21

1:50 PM
Newark [EWR](#)6:58 PM
San Francisco [SFO](#)1 stop
8hr 8minUS Airways 660 / 686
[Seat map](#)☐ Select this return

Return Sat, May 24

10:25 PM
San Francisco
[SFO](#)9:33 AM
Newark [EWR](#)1 stop
8hr 8minUS Airways 898 / 1984
[Seat map](#)

This is an overnight flight, which will arrive one (1) day later.

Earn \$6.84

[Don't Miss Out! Save up to \\$525* by booking your Flight and Hotel together](#)

Run New York - San Francisco on May 21 - May 24 on these other sites:

Best Price Guaranteed on Flights to San Francisco
Flying to SFO? Book Direct and Save on Flights to SFO!
Book flight + hotel together and save!

New York to San Francisco

May 21 to May 24

1 traveler

Clicking these links will open a new window but won't affect your current results page.

Data Science at IM

- Builds models from terabytes of data:
 - Propensity to buy
 - Propensity to click on an offer
- Been learning about “how to do” data science for five years

History of Data Processing at Intent Media

Originally: Hadoop Java API

Java API

Hadoop Map Reduce

Hadoop DFS

Hadoop Java API Example

Example: Anagram Mapper in Java

```
public class AnagramMapper extends MapReduceBase implements
    Mapper<LongWritable, Text, Text, Text> {

    private Text sortedText = new Text();
    private Text originalText = new Text();

    public void map(LongWritable key, Text value,
        OutputCollector<Text, Text> outputCollector, Reporter reporter)
        throws IOException {

        String word = value.toString();
        char[] wordChars = word.toCharArray();
        Arrays.sort(wordChars);
        String sortedWord = new String(wordChars);
        sortedText.set(sortedWord);
        originalText.set(word);
        outputCollector.collect(sortedText, originalText);
    }
}
```

Sort
characters



beta -> abet, beat -> abet

Example: Anagram Reducer in Java

```
public class AnagramReducer extends MapReduceBase implements Reducer<Text, Text, Text, Text> {

    private Text outputKey = new Text();
    private Text outputValue = new Text();

    public void reduce(Text anagramKey, Iterator<Text> anagramValues,
        OutputCollector<Text, Text> results, Reporter reporter) throws IOException {
        String output = "";
        while(anagramValues.hasNext())
        {
            Text anagam = anagramValues.next();
            output = output + anagam.toString() + "~";
        }
        StringTokenizer outputTokenizer = new StringTokenizer(output, "~");
        if(outputTokenizer.countTokens()>=2)
        {
            output = output.replace("~", ",");
            outputKey.set(anagramKey.toString());
            outputValue.set(output);
            results.collect(outputKey, outputValue);
        }
    }
}
```


Downsides of Java API

- Hard to write
 - Need to think in “map -> reduce”
- Hard to test
- Hard to read and understand when you go back to it in the future
- Too low level

“There's no problem in Computer Science
that can't be solved by adding another layer
of abstraction”

Another Layer of Abstraction

?

Java

Hadoop Map Reduce

Hadoop DFS

Apache Pig

Pig

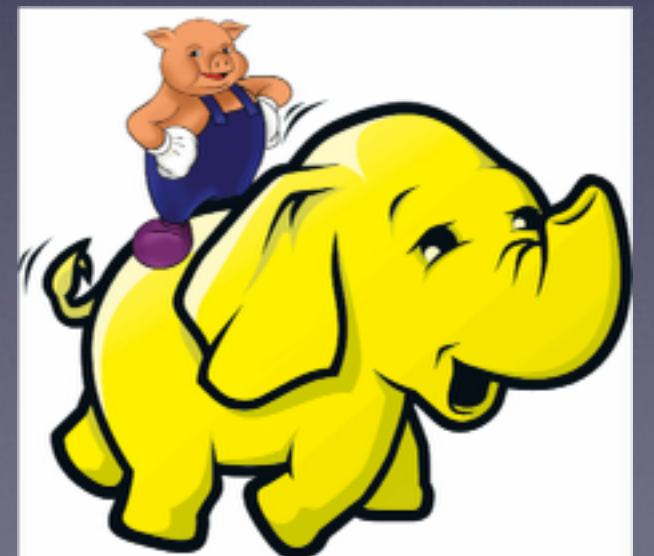
Java

Hadoop Map Reduce

Hadoop DFS

Apache Pig

- Pig is a higher level declarative language that reduces to map/reduce queries
- Simpler to reason about
- Allows for User Defined Functions



```
-- NORMALIZE DATA
--
signals = FOREACH btl_training GENERATE $2..;
grouped_data = GROUP signals ALL;
vars = FOREACH grouped_data GENERATE TupleVar(signals) AS tuple_variance;
```

User
Defined
Function



©

©

Pig Downsides

- Defines its own language
- Custom operations in Python, Java, etc
- Hard to unit test

Cascading

Cascading

Java

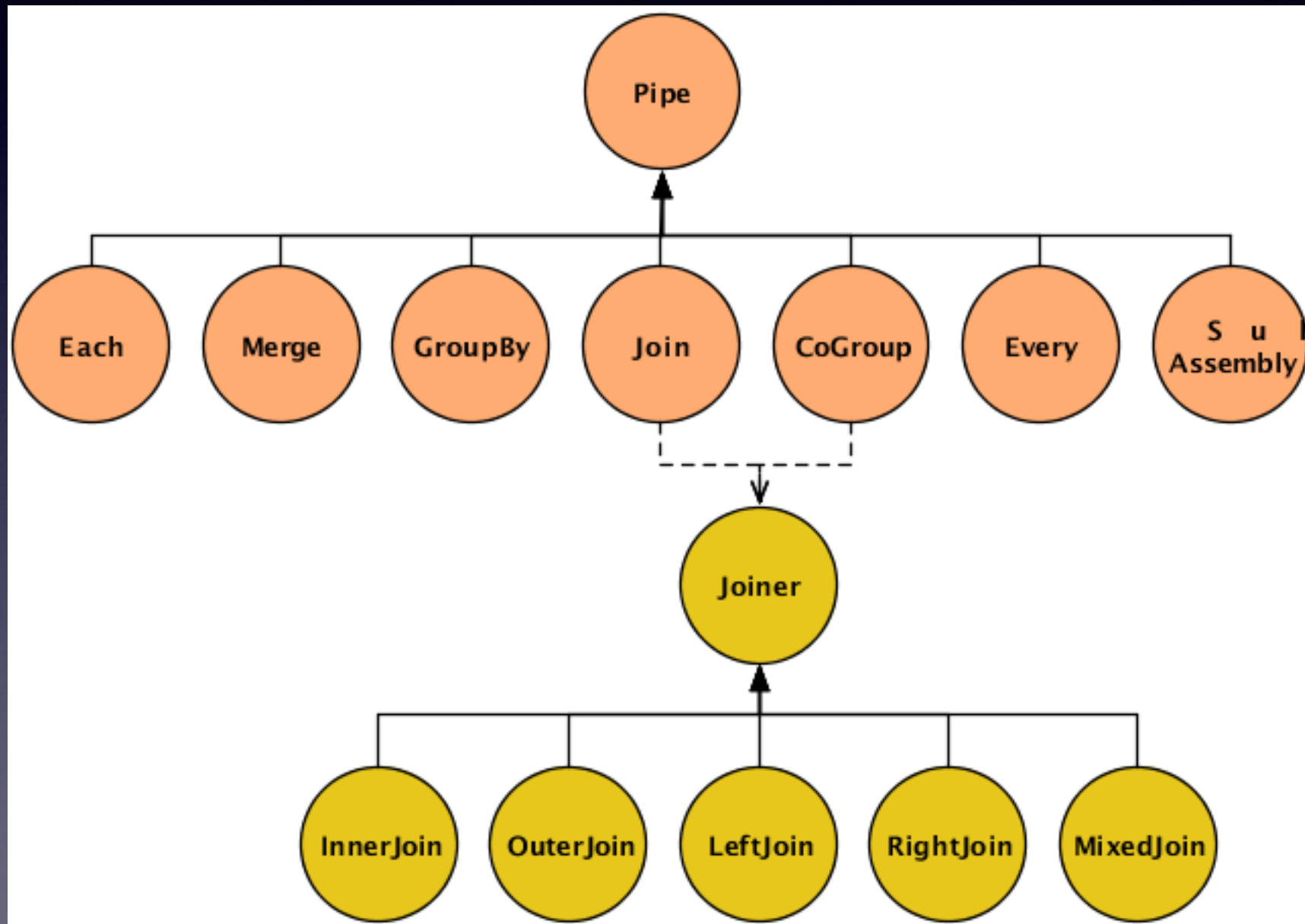
Hadoop Map Reduce

Hadoop DFS

Cascading overview

- Java data processing library
 - Thinks of your data as a stream
 - Uses a taps, pipes, and sinks abstraction
- Built in functions for common tasks

Built in Operations



Taps

```
public class
    Main
    {
    public static void
    main( String[] args )
    {
        String docPath = args[ 0 ];
        String wcPath = args[ 1 ];

        Properties properties = new Properties();
        AppProps.setApplicationJarClass( properties, Main.class );
        HadoopFlowConnector flowConnector = new HadoopFlowConnector( properties );

        // create source and sink taps
        Tap docTap = new Hfs( new TextDelimited( true, "\t" ), docPath );
        Tap wcTap = new Hfs( new TextDelimited( true, "\t" ), wcPath );

        // specify a regex operation to split the "document" text lines into a token stream
        Fields token = new Fields( "token" );
        Fields text = new Fields( "text" );
        RegexSplitGenerator splitter = new RegexSplitGenerator( token, "[ \\[\\]\\(\\),\\.]" );
        // only returns "token"
        Pipe docPipe = new Each( "token", text, splitter, Fields.RESULTS );

        // determine the word counts
        Pipe wcPipe = new Pipe( "wc", docPipe );
        wcPipe = new GroupBy( wcPipe, token );
        wcPipe = new Every( wcPipe, Fields.ALL, new Count(), Fields.ALL );

        // connect the taps, pipes, etc., into a flow
        FlowDef flowDef = FlowDef.flowDef()
            .setName( "wc" )
            .addSource( docPipe, docTap )
            .addTailSink( wcPipe, wcTap );

        // write a DOT file and run the flow
        Flow wcFlow = flowConnector.connect( flowDef );
        wcFlow.writeDOT( "dot/wc.dot" );
        wcFlow.complete();
    }
}
```

Pipes

“There's no problem in Computer Science
that can't be solved by adding another layer
of abstraction”

Cascalog

Cascalog

Cascading

Java

Hadoop Map Reduce

Hadoop DFS

Clojure

Lisp that runs on the JVM

(and the CLR, and on Javascript)

“Our hypothesis was that if we wrote our software in Lisp, we'd be able to get features done faster than our competitors, and also to do things in our software that they couldn't do.”

—Paul Graham

Cascalog example

```
(defn generate-model-data []
  (let [get-word-vector (mapfn [attributes]
                              (when attributes
                                [(vec (.split attributes "\\s+"))]))
        to-binary (mapfn [purchase-amount]
                        (if (and purchase-amount (> purchase-amount 0)) 1 -1))]
    (?<- (stdout) [?impression-id ?word-split ?did-purchase]
      (ad-copy ?impression-id ?ad-copy)
      (clicks !!click-id ?impression-id !!purchase-amount)
      (get-word-vector ?ad-copy :> ?word-split)
      (c/sum !!purchase-amount :> ?purchase-amount-sum)
      (to-binary ?purchase-amount-sum :> ?did-purchase))))
```

Datalog

“Datalog is a truly declarative logic programming language that syntactically is a subset of Prolog.”

Hello World

Word Count

Word Count SQL

```
SELECT    word, COUNT(*)  
FROM      words  
GROUP BY word
```


Word Count Java Hadoop

< Hello, 1>
< World, 1>
< Bye, 1>
< World, 1>

```
public class WordCount {

    public static class Map extends 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, Context context) throws IOException, InterruptedException {
            String line = value.toString();
            StringTokenizer tokenizer = new StringTokenizer(line);
            while (tokenizer.hasMoreTokens()) {
                word.set(tokenizer.nextToken());
                context.write(word, one);
            }
        }
    }

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

        public void reduce(Text key, Iterable<IntWritable> values, Context context)
            throws IOException, InterruptedException {
            int sum = 0;
            for (IntWritable val : values) {
                sum += val.get();
            }
            context.write(key, new IntWritable(sum));
        }
    }

    public static void main(String[] args) throws Exception {
        Configuration conf = new Configuration();

        Job job = new Job(conf, "wordcount");

        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);

        job.setMapperClass(Map.class);
        job.setReducerClass(Reduce.class);

        job.setInputFormatClass(TextInputFormat.class);
        job.setOutputFormatClass(TextOutputFormat.class);

        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        job.waitForCompletion(true);
    }
}
```

< Hello, 1>
< World, 2>
< Bye, 1>

Word count PIG

```
A = load '/tmp/alice.txt';  
B = foreach A generate flatten(TOKENIZE((chararray)$0)) as word;  
C = filter B by word matches '\\w+';  
D = group C by word;  
E = foreach D generate COUNT(C), group;  
store E into '/tmp/alice_wordcount';
```

Word Count Cascading

```
// define source and sink Taps.
Scheme sourceScheme = new TextLine( new Fields( "line" ) );
Tap source = new Hfs( sourceScheme, inputPath );

Scheme sinkScheme = new TextLine( new Fields( "word", "count" ) );
Tap sink = new Hfs( sinkScheme, outputPath, SinkMode.REPLACE );

// the 'head' of the pipe assembly
Pipe assembly = new Pipe( "wordcount" );

// For each input Tuple
// parse out each word into a new Tuple with the field name "word"
// regular expressions are optional in Cascading
String regex = "(?!\\pL)(?=\\pL)[^ ]*(?<=\\pL)(?!\\pL)";
Function function = new RegexGenerator( new Fields( "word" ), regex );
assembly = new Each( assembly, new Fields( "line" ), function );

// group the Tuple stream by the "word" value
assembly = new GroupBy( assembly, new Fields( "word" ) );

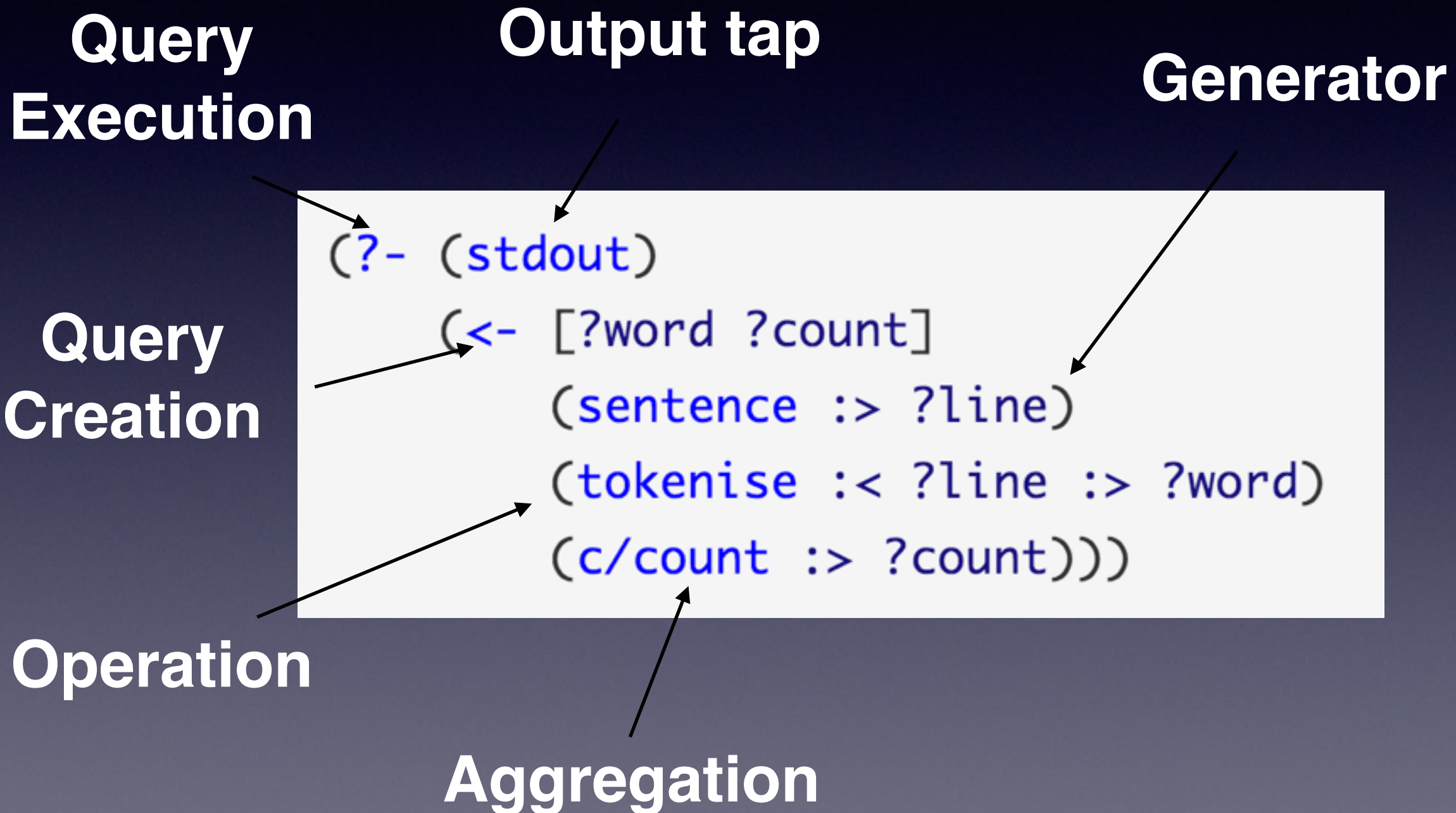
// For every Tuple group
// count the number of occurrences of "word" and store result in
// a field named "count"
Aggregator count = new Count( new Fields( "count" ) );
assembly = new Every( assembly, count );

// initialize app properties, tell Hadoop which jar file to use
Properties properties = new Properties();
FlowConnector.setApplicationJarClass( properties, Main.class );

// plan a new Flow from the assembly using the source and sink Taps
// with the above properties
FlowConnector flowConnector = new FlowConnector( properties );
Flow flow = flowConnector.connect( "word-count", source, sink, assembly );

// execute the flow, block until complete
flow.complete();
```


Word Count Cascalog



Tokenize

```
(def/defmapcatfn tokenise [line]
  "reads in a line of string and splits it by a regular expression"
  (clojure.string/split line #"[\[\]\(\),.\s]+"))
```

Cascalog overview

(Credit to Jon Sondag, Head of Data Science, Intent Media)

<https://github.com/johnnywalleye/nyc-clj-meetup-apr-14>

Real example

```
(defn generate-model-data []
  (let [get-word-vector (mapfn [attributes]
    (when attributes
      [(vec (.split attributes "\\s+"))]))
    to-binary (mapfn [purchase-amount]
      (if (and purchase-amount (> purchase-amount 0)) 1 -1))]
    (?<- (stdout) [?impression-id ?word-split ?did-purchase]
      (ad-copy ?impression-id ?ad-copy)
      (clicks !!click-id ?impression-id !!purchase-amount)
      (get-word-vector ?ad-copy :> ?word-split)
      (c/sum !!purchase-amount :> ?purchase-amount-sum)
      (to-binary ?purchase-amount-sum :> ?did-purchase))))
```

Pre-aggregation
(Generators)

["impression-1" "buy this product"]	["click-1" "impression-3" 0]
["impression-2" "great deal"]	["click-2" "impression-2" 0]
["impression-3" "cheap sale"]	["click-3" "impression-2" 100]
["impression-4" "cheap sale"]	

Real example

```
(defn generate-model-data []
  (let [get-word-vector (mapfn [attributes]
    (when attributes
      [(vec (.split attributes "\\s+"))]))
    to-binary (mapfn [purchase-amount]
      (if (and purchase-amount (> purchase-amount 0)) 1 -1))]
    (?<- (stdout) [?impression-id ?word-split ?did-purchase]
      (ad-copy ?impression-id ?ad-copy)
      (clicks !!click-id ?impression-id !!purchase-amount)
      (get-word-vector ?ad-copy :> ?word-split)
      (c/sum !!purchase-amount :> ?purchase-amount-sum)
      (to-binary ?purchase-amount-sum :> ?did-purchase))))
```

Pre-aggregation
(Join)

```
[["impression-1" "buy this product" nil nil]
 ["impression-2" "great deal" "click-2" 0]
 ["impression-2" "great deal" "click-3" 100]
 ["impression-3" "cheap sale" "click-1" 0]
 ["impression-4" "cheap sale" nil nil]]
```


Real example


```
(defn generate-model-data []
  (let [get-word-vector (mapfn [attributes]
    (when attributes
      [(vec (.split attributes "\\s+"))]))
    to-binary (mapfn [purchase-amount]
      (if (and purchase-amount (> purchase-amount 0)) 1 -1))]
    (?<- (stdout) [?impression-id ?word-split ?did-purchase]
      (ad-copy ?impression-id ?ad-copy)
      (clicks !!click-id ?impression-id !!purchase-amount)
      (get-word-vector ?ad-copy :> ?word-split)
      (c/sum !!purchase-amount :> ?purchase-amount-sum)
      (to-binary ?purchase-amount-sum :> ?did-purchase))))
```

Pre-aggregation
(Operations)

```
[["impression-1" ["buy" "this" "product"] nil nil]
["impression-2" ["great" "deal"] "click-2" 0]
["impression-2" ["great" "deal"] "click-3" 100]
["impression-3" ["cheap" "sale"] "click-1" 0]
["impression-4" ["cheap" "sale"] nil nil]]
```

Real example

```
(defn generate-model-data []
  (let [get-word-vector (mapfn [attributes]
    (when attributes
      [(vec (.split attributes "\\s+"))]))
    to-binary (mapfn [purchase-amount]
      (if (and purchase-amount (> purchase-amount 0)) 1 -1))]
    (?<- (stdout) [?impression-id ?word-split ?did-purchase]
      (ad-copy ?impression-id ?ad-copy)
      (clicks !!click-id ?impression-id !!purchase-amount)
      (get-word-vector ?ad-copy :> ?word-split)
      (c/sum !!purchase-amount :> ?purchase-amount-sum)
      (to-binary ?purchase-amount-sum :> ?did-purchase))))
```



Aggregation

```
[["impression-1" ["buy" "this" "product"] 0]
["impression-2" ["great" "deal"] 100]
["impression-3" ["cheap" "sale"] 0]
["impression-4" ["cheap" "sale"] 0]]
```

Real example

```
(defn generate-model-data []
  (let [get-word-vector (mapfn [attributes]
    (when attributes
      [(vec (.split attributes "\\s+"))]))
    to-binary (mapfn [purchase-amount]
      (if (and purchase-amount (> purchase-amount 0)) 1 -1))]
    (?<- (stdout) [?impression-id ?word-split ?did-purchase]
      (ad-copy ?impression-id ?ad-copy)
      (clicks !!click-id ?impression-id !!purchase-amount)
      (get-word-vector ?ad-copy :> ?word-split)
      (c/sum !!purchase-amount :> ?purchase-amount-sum)
      (to-binary ?purchase-amount-sum :> ?did-purchase)))) ←
```

Post-aggregation
(Operations)

```
[["impression-1" ["buy" "this" "product"] -1]
 ["impression-2" ["great" "deal"] 1]
 ["impression-3" ["cheap" "sale"] -1]
 ["impression-4" ["cheap" "sale"] -1]]
```


Demo

Built-in Filter Operations

- first-n
- limit
- limit-rank
- fixed-sample
- fixed-sample-agg
- re-parse

Built-in Agg Operations

- avg
- count/!count
- distinct-count
- max
- min
- sum

Built-in Higher Order Functions

- all
- any
- comp
- each
- negate
- partial

Workflow

- On a sampled dataset:
 - Unit test the individual functions
 - End-to-end test the workflow
- Then, test on the cluster

Checkpoint

```
(defn run-workflow [input-path-1 input-path-2 workflow-output-path]
  (let [workflow-tmp-path "/tmp/generate-model-workflow"]
    (with-job-conf { "mapred.reduce.tasks" 500
                    "mapred.output.compress" "true"
                    "mapred.output.compression.type" "BLOCK"}
      (workflow [workflow-tmp-path]
        step-1 ([:deps []
                  :tmp-dirs step-1-path]
                (?- (hfs-seqfile step-1-path)
                    (query-1 input-path-1)))
        step-2 ([:deps []
                  :tmp-dirs step-2-path]
                (?- (hfs-seqfile step-2-path)
                    (query-2 input-path-2)))
        step-3 ([:deps [step-1 step-2]
                  :tmp-dirs step-3-path]
                (?- (hfs-seqfile step-3-path)
                    (query-3 step-1-path step-2-path)))
        step-4 ([:deps [step-3]
                  :tmp-dirs step-4-path]
                (?- (make-output-tap workflow-output-path)
                    (query-4 step-3-path)))))))
```

The Future



How can we move
even faster?

“Hadoop is the EJB of data processing”

–Dave Thomas

What's next?

Cascalog

Cascading

Java

?

Hadoop DFS

Cascalog 2.0

Cascalog 2.0 Backends

- Spark
- Storm
- Cascading

Cascalog 2.0 Backends



Run programs up to 100x faster than Hadoop
MapReduce in memory, or 10x faster on disk.

Cascading 3.0

Cascading 3.0 backends

- Spark
- Storm
- Tez
- MapReduce

The Future

- Data processing time will continue to decrease, hopefully by orders of magnitude
- We'll be able to write our data processing code at a high level of abstraction and let the system handle the complexity underneath

Questions?