Big Data, Small Code

Using Java 8 and Apache Crunch to quickly develop concise, efficient, and testable data pipelines

About me

Worked with Hadoop and big data for 5 years.

First at Last.fm, then Spotify, then Soundcloud

Working on improving the developer experience for data

Committer on Apache Crunch project

Music producer and artist: http:/thewit.ch/portfolio

davw@apache.org // @dawhiting

State of the Big Data world 2016

The hype

Spark, machine learning, real-time, advanced visualisation, event sourcing

The reality

Hadoop MapReduce jobs running periodically for ETL, aggregation and analytics purposes make up the vast majority of jobs at major organisations using big data.

MapReduce is not dead

The “boring” end of the pipeline

- Filtering
- Transforming
- Splitting
- Correlating
- Monitoring
- Aggregating

Apache Crunch

Apache Crunch

“The Apache Crunch Java library provides a framework for writing, testing, and running MapReduce pipelines. Its goal is to make pipelines that are composed of many user-defined functions simple to write, easy to test, and efficient to run.”*

- Functional
- Type-safe
- Runs as MapReduce*
- Zero boilerplate
- Easy testing of pipeline logic
- Used at many major organisations

* not strictly tied to MapReduce, also can be run in memory or on Spark

Approach

Apache Crunch considers your data as immutable lazily-evaluated “collections” of records, which represent data on HDFS or data in the process of being transformed.

By applying functional transformations to these collections, you can obtain modified collections, and when you have the data you need, you can write it back to HDFS or elsewhere.

Should be familiar to anyone who’s used Spark’s RDDs or the Java 8 stream API.

```
Pipeline crunch = new MRPipeline(WordCountJob.class);
crunch.read(From.textFile("/path/on/hdfs"))
      .parallelDo(new DoFn<String, String>() {
          public void process(String s, Emitter<String> emitter) {
              for (String word: s.split(" ")) {
                  emitter.emit(word);
              }
          }
      }, strings())
      .count()
      .parallelDo(new MapFn<Pair<String, Long>, String>() {
          public String map(Pair<String, Long> wordCount) {
              return wordCount.first() + ":" + wordCount.second();
          }
      }, strings())
      .write(To.textFile("/path/to/output"));
crunch.done();
```

```
Pipeline crunch = new MRPipeline(WordCountJob.class);
crunch.read(From.textFile("/path/on/hdfs"))
      .parallelDo(

                                  s.split(" ")
                  emitter.emit(word)


                  )
      .count()
      .parallelDo(

                     wordCount.first() + ":" + wordCount.second()

                  )
      .write(To.textFile("/path/to/output"));
crunch.done();
```

Java 8 + Crunch = Crunch Lambda

```
Pipeline crunch = new MRPipeline(WordCountJobLambda.class);
Lambda.wrap(
  crunch.read    (From.textFile("/path/on/hdfs")))
        .flatMap(line -> Arrays.stream(line.split(" ")), strings())
        .count   ()
        .map     (wc -> wc.first() + ":" + wc.second(), strings())
        .write   (To.textFile("/path/to/output"));
crunch.done();
```

The Crunch data model

Collection

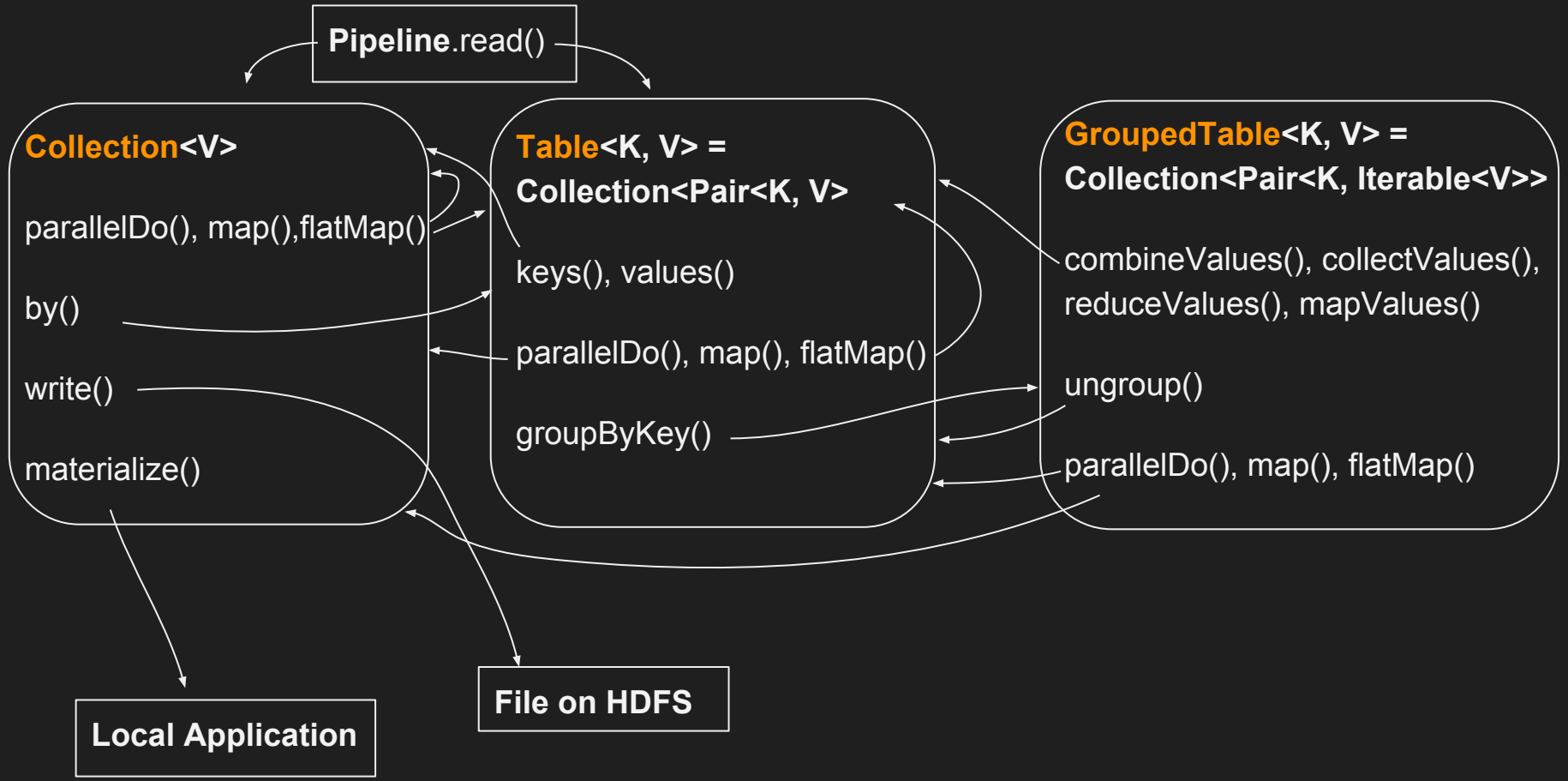
some_value_data_1
some_value_data_2
some_value_data_3
some_value_data_4
some_value_data_5
some_value_data_6

Table

key A	some_value_data_1
key A	some_value_data_2
key A	some_value_data_3
key B	some_value_data_4
key C	some_value_data_5
key C	some_value_data_6

GroupedTable

key A	some_value_data_1 some_value_data_2 some_value_data_3
key B	some_value_data_4
key C	some_value_data_5 some_value_data_6


Pipeline.read()
Collection<V>
parallelDo(), map(),flatMap()
by()
write()
materialize()
Table<K, V> =
Collection<Pair<K, V>
keys(), values()
parallelDo(), map(), flatMap()
groupByKey()
GroupedTable<K, V> =
Collection<Pair<K, Iterable<V>>
combineValues(), collectValues(),
reduceValues(), mapValues()
ungroup()
parallelDo(), map(), flatMap()
Local Application
File on HDFS

What if I’m not running a word-count company?

Example: Monitoring event counts

```
Pipeline crunch = new MRPipeline(EventStatsJob.class);
PushGateway pushGateway = new PushGateway(PUSH_GATEWAY_ADDRESS);
Gauge gauge = Gauge.build()
    .name("event_count")
    .labelNames("event_type", "application")
    .create();
LTable<Long, Event> events = readEvents(crunch, dataPath);
events.values()
      .map(StatsKey::fromEvent, StatsKey.pType())
      .count()
      .materialize()
      .forEach(statsRecord ->
          gauge.labels(
                  statsRecord.first().eventType,
                  statsRecord.first().application)
               .set(statsRecord.second()));

pushGateway.push(gauge, "event_stats_job");
```

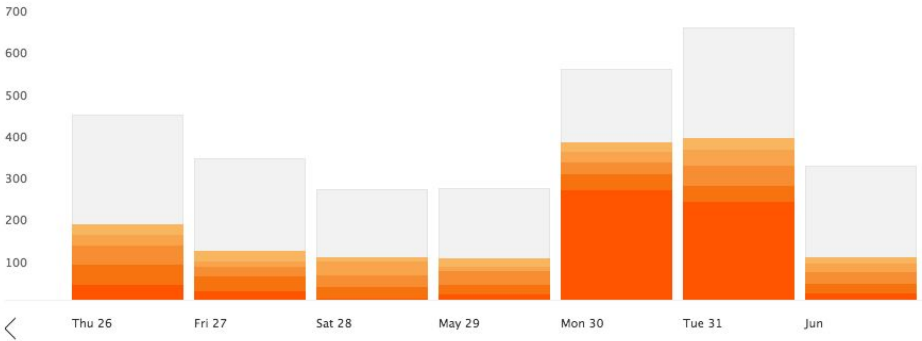
Example: SoundCloud Stats Toplists

Home | Collection | Search | Upload | Demoscene Time Mac...

Stats

2,858	62	2	2	1	0
▶ Plays	♥ Likes	Comments	Reposts	Downloads	RSS downloads

Last 7 days
26 May - 1 Jun, 2016



Rescue	591
Protect & Survive	225
Kernel Panic	220
Artillery	154
Airship	140
Other tracks	1,528

Most played tracks

Rescue
591

2	Protect & Survive	225
3	Kernel Panic	220
4	Artillery	154
5	Airship	140
6	Satellite	111
7	Cavalry	97
8	Disk Zero	95
9	Jetpack	83
10	Stealth	80

Top countries | Top cities

United States
890

2	United Kingdom	212
3	France	162
4	Germany	157
5	Canada	154
6	Peru	130
7	Sweden	102
8	Switzerland	87
9	Finland	69
10	Ukraine	65

Who played the most

Zorobabel
129

2	Alarax	78
3	AUXelerate/Neocleora	50
4	sample0327	44
5	Jakepearl42	36
6	luis camilo velazques	32
7	Chiroptera-Boy	30
8	Asen Jechev	29
9	Rent Kaos	24
10	rokkis	24

Websites | Apps | RSS

soundcloud.com
169

2	www.google.com	91
3	www.facebook.com	67
4	w.soundcloud.com/player/?url=http	41
5	w.soundcloud.com/player/?url=http	31
6	www.reddit.com/r/overwatch/	31
7	www.reddit.com/r/overwatch	30
8	w.soundcloud.com/player/?referrer	26
9	m.soundcloud.com	21
10	www.google.it	20

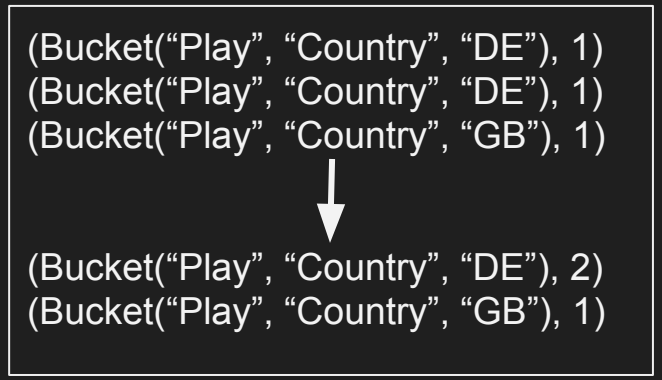
```
LTable<Bucket, Long> summedBuckets =
  events
    .flatMap(this::createFactsFromEvent,
             tableOf(proto(Bucket.class), longs()))
    .groupByKey()
    .reduceValues((a, b) -> a + b)
    .filterByValue(count -> count > 0L); // Remove negative counts

return summedBuckets
    .map(this::remapDimensionKeys,
         tableOf(proto(FactKey.class), proto(TopListPair.class)))
    .groupByKey()
    .mapValues(rowSet -> findTopK(k, rowSet),
               collections(proto(TopListPair.class)));
```

```
LTable<Bucket, Long> summedBuckets =
  events
    .flatMap(this::createFactsFromEvent,
             tableOf(proto(Bucket.class), longs()))
    .groupByKey()
    .reduceValues((a, b) -> a + b)
    .filterByValue(count -> count > 0L); // Remove negative counts

return summedBuckets
    .map(this::remapDimensionKeys,
         tableOf(proto(FactKey.class), proto(TopListPair.class)))
    .groupByKey()
    .mapValues(rowSet -> findTopK(k, rowSet),
               collections(proto(TopListPair.class)));
```

SoundPlayed(“David”, “DE”, “Web”)

(Bucket(“Play”, “Country”, “DE”), 1)
(Bucket(“Play”, “Client”, “Web”), 1)
(Bucket(“Play”, “User”, “David”), 1)

```
LTable<Bucket, Long> summedBuckets =
  events
    .flatMap(this::createFactsFromEvent,
             tableOf(proto(Bucket.class), longs()))
    .groupByKey()
    .reduceValues((a, b) -> a + b)
    .filterByValue(count -> count > 0L); // Remove negative counts

return summedBuckets
    .map(this::remapDimensionKeys,
         tableOf(proto(FactKey.class), proto(TopListPair.class)))
    .groupByKey()
    .mapValues(rowSet -> findTopK(k, rowSet),
               collections(proto(TopListPair.class)));
```



```
LTable<Bucket, Long> summedBuckets =
  events
    .flatMap(this::createFactsFromEvent,
             tableOf(proto(Bucket.class), longs()))
    .groupByKey()
    .reduceValues((a, b) -> a + b)
    .filterByValue(count -> count > 0L); // Remove negative counts

return summedBuckets
    .map(this::remapDimensionKeys,
         tableOf(proto(FactKey.class), proto(TopListPair.class)))
    .groupByKey()
    .mapValues(rowSet -> findTopK(k, rowSet),
               collections(proto(TopListPair.class)));
```

```
LTable<Bucket, Long> summedBuckets =
  events
    .flatMap(this::createFactsFromEvent,
             tableOf(proto(Bucket.class), longs()))
    .groupByKey()
    .reduceValues((a, b) -> a + b)
    .filterByValue(count -> count > 0L); // Remove negative counts

return summedBuckets
    .map(this::remapDimensionKeys,
         tableOf(proto(FactKey.class), proto(TopListPair.class)))
    .groupByKey()
    .mapValues(rowSet -> findTopK(k, rowSet),
               collections(proto(TopListPair.class)));
```

(Bucket(“Play”, “Country”, “DE”), 2)

(
FactKey(“Play”, “Country”),
TopListPair(“DE”, 2)
)

```
LTable<Bucket, Long> summedBuckets =
  events
    .flatMap(this::createFactsFromEvent,
             tableOf(proto(Bucket.class), longs()))
    .groupByKey()
    .reduceValues((a, b) -> a + b)
    .filterByValue(count -> count > 0L); // Remove negative counts

return summedBuckets
    .map(this::remapDimensionKeys,
         tableOf(proto(FactKey.class), proto(TopListPair.class)))
    .groupByKey()
    .mapValues(rowSet -> findTopK(k, rowSet),
               collections(proto(TopListPair.class)));
```

```
public Collection<TopListPair> findTopK(int k, Stream<TopListPair> input) {
   SortedSet<TopListPair> set = new TreeSet<>(
           Comparator.comparingLong(TopListPair::getCount)
                   .reversed()
                   .thenComparing(TopListPair::getDimensionValue));
   input.forEach(pair -> {
       set.add(pair);
       if (set.size() > k) {
           set.remove(set.last());
       }
   });
   return set;
}
```

Impact at SoundCloud

- Complex legacy MapReduce job graphs replaced with simple Crunch jobs
- 4 teams now using Crunch
- Developers report increased productivity
- Jobs with no tests now have tests

Future

Future

- Crunch is already feature-complete and stable
- Apache Beam is coming
- Crunch -> Beam should be a smooth transition
- Beam on MapReduce via Crunch?

Summary

- MapReduce isn’t going away soon
- Crunch is a pragmatic way to keep working with it
- Java 8 makes functional programming easier
- Crunch Lambda + Java 8 = <3

Learn More

- crunch.apache.org
- developers.soundcloud.com/blog
- radicaljava.com
- tinyurl.com/bbuzz-crunch

Demoscene Time Machine - Gravity

“Drawing inspiration from 90s video game soundtracks and the demo and tracker scenes of the early years of the internet, Demoscene Time Machine’s Gravity combines classic 8-bit sounds and melodies with modern electronic music styles to create a sound which is both nostalgic and forward-thinking.” - Echoetic

“Takes me back to those wondrous days when I was a fresh-faced kid experiencing the magic of the early internet.” - ChipWIN

Out now on www.demoscenetimemachine.com and all good streaming and download stores.