SPARK AT OOYALA

EVAN CHAN
AUG 29, 2013

OOYALA

Who is this guy?

- Staff Engineer, Compute and Data Services, Ooyala
- Building multiple web-scale real-time systems on top of C*, Kafka, Storm, etc.
- Scala/Akka guy
- Very excited by open source, big data projects
- @evanfchan

Agenda

- Ooyala and Big Data
- What problem are we trying to solve?
- Spark and Shark
- Our Spark/Cassandra Architecture
- Spark Job Server

OOYALA AND BIG DATA

OOYALA

Powering personalized video experiences across all screens.

COMPANY OVERVIEW

Founded in 2007

Commercially launch in 2009

230+ employees in Silicon Valley, LA, NYC, London, Paris, Tokyo, Sydney & Guadalajara

Global footprint, 200M unique users, 110+ countries, and more than 6,000 websites

Over 1 billion videos played per month and 2 billion analytic events per day

25% of U.S. online viewers watch video powered by Ooyala

TRUSTED VIDEO PARTNER

CUSTOMERS

- Slashdot
- Bloomberg
- ESPN
- STAR
- MIRAMAX
- Geeknet
- Victoria's Secret
- DELL
- The North Face
- General Mills
- REI
- Whole Foods Market
- Telegraph
- The Washington Post
- Arsenal

STRATEGIC PARTNERS

- Panasonic
- ERICSSON
- MOTOROLA
- LG
- YAHOO! JAPAN
- Telefónica
- Akamai
- Microsoft

We have a large Big Data stack

- > 250GB of fresh logs every day
- Total of 28TB of data managed over ~200 Cassandra nodes
- Traditional stack: Hadoop, Ruby, Cassandra, Ruby...
- Real-time stack: Kafka, Storm, Scala, Cassandra
- New stack: Kafka, Akka, Cassandra, Spark, Scala/Go

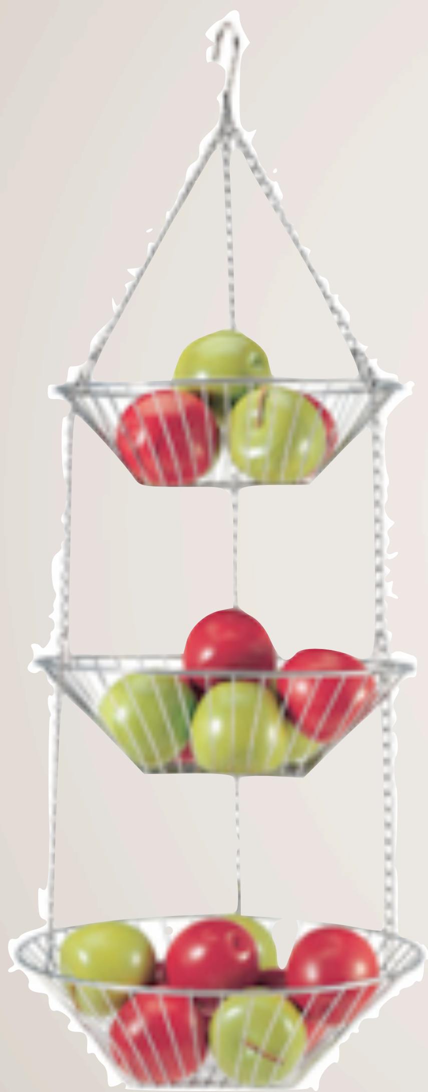
Becoming a big Spark user...

- Started investing in Spark beginning of 2013
- 2 teams of developers doing stuff with Spark
- Actively contributing to Spark developer community
- Deploying Spark to a large (>100 node) production cluster
- Spark community very active, huge amount of interest

WHAT PROBLEM ARE WE TRYING TO SOLVE?

From mountains of raw data...

To nuggets of truth...



- Quickly
- Painlessly
- At scale?

TODAY: PRECOMPUTED AGGREGATES

- Video metrics computed along several high cardinality dimensions
- Very fast lookups, but inflexible, and hard to change
- Most computed aggregates are never read
- What if we need more dynamic queries?
    - Top content for mobile users in France
    - Engagement curves for users who watched recommendations
    - Data mining, trends, machine learning

THE STATIC - DYNAMIC CONTINUUM

100% Precomputation

- Super fast lookups
- Inflexible, wasteful
- Best for 80% most common queries

100% Dynamic

- Always compute results from raw data
- Flexible but slow

WHERE WE WANT TO BE

Partly dynamic

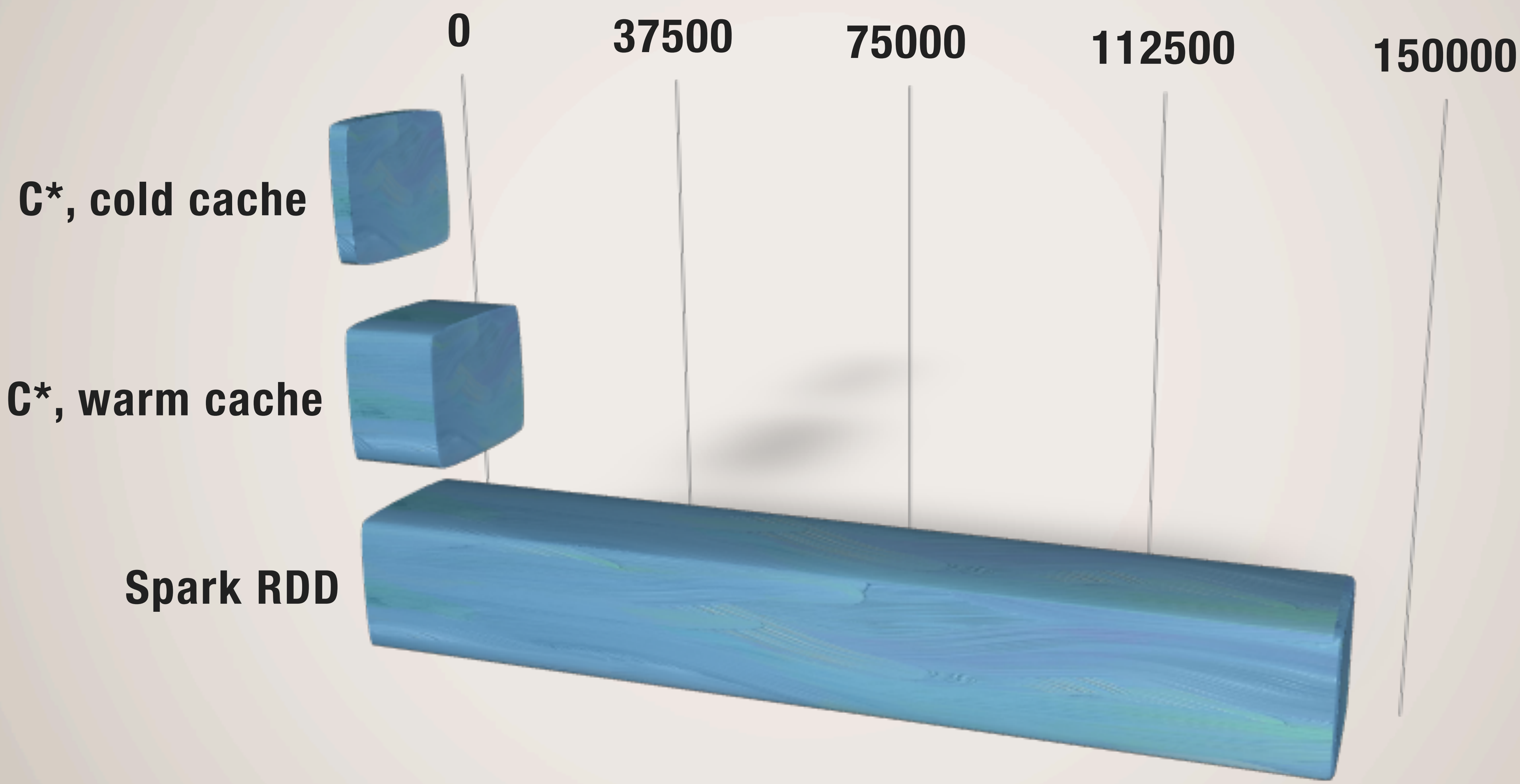
- Pre-aggregate most common queries
- Flexible, fast dynamic queries
- Easily generate many materialized views

INDUSTRY TRENDS

- Fast execution frameworks
  - Impala
- In-memory databases
  - VoltDB, Druid
- Streaming and real-time
- Higher-level, productive data frameworks

WHY SPARK?

THROUGHPUT: MEMORY IS KING



6-node C*/DSE 1.1.9 cluster,
Spark 0.7.0

Spark cached RDD 10-50x faster than raw Cassandra

DEVELOPERS LOVE IT

- “I wrote my first aggregation job in 30 minutes”
- High level “distributed collections” API
- No Hadoop cruft
- Full power of Scala, Java, Python
- Interactive REPL shell

SPARK VS HADOOP WORD COUNT

```
file = spark.textFile("hdfs://...")

file.flatMap(line => line.split(" "))
    .map(word => (word, 1))
    .reduceByKey(_ + _)
```

```
package org.myorg;

import java.io.IOException;
import java.util.*;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.*;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

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);
 }

}
```

ONE PLATFORM TO RULE THEM ALL

Bagel - Pregel on Spark

SHARK

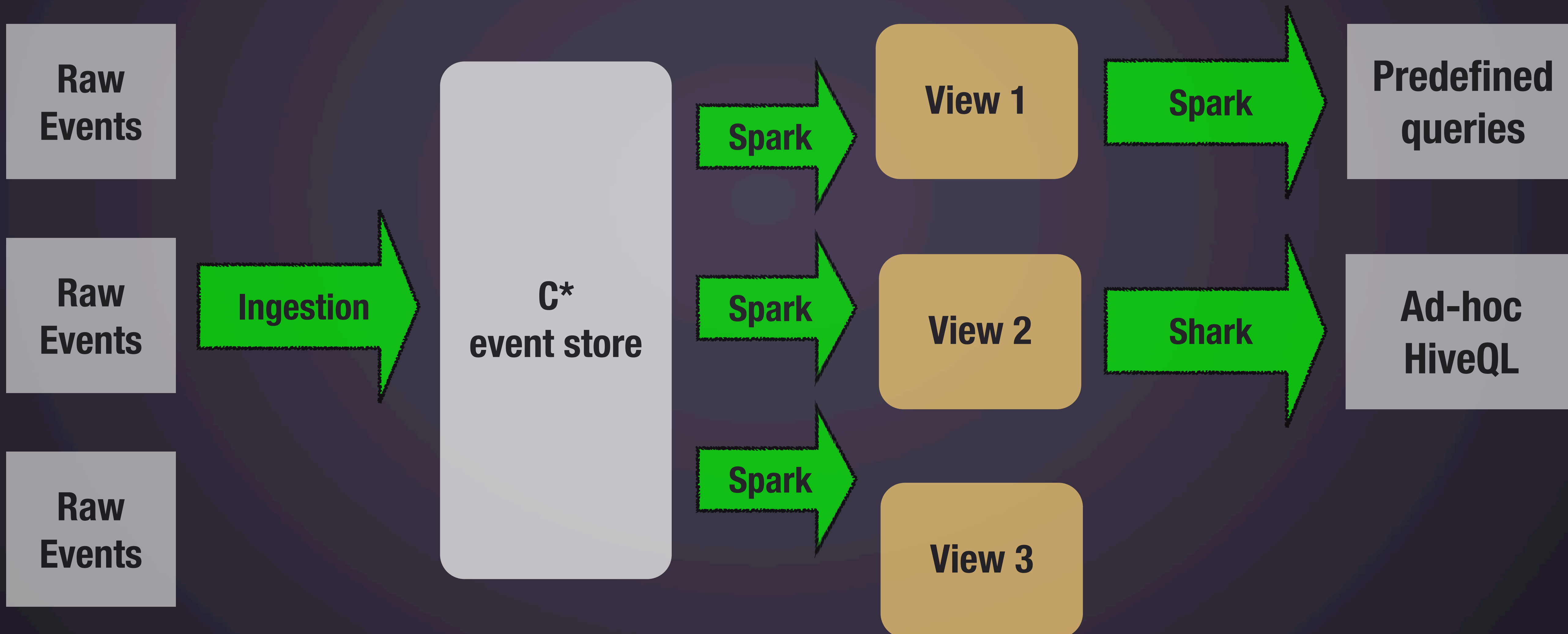
HIVE on Spark

Spark Streaming - discretized stream processing

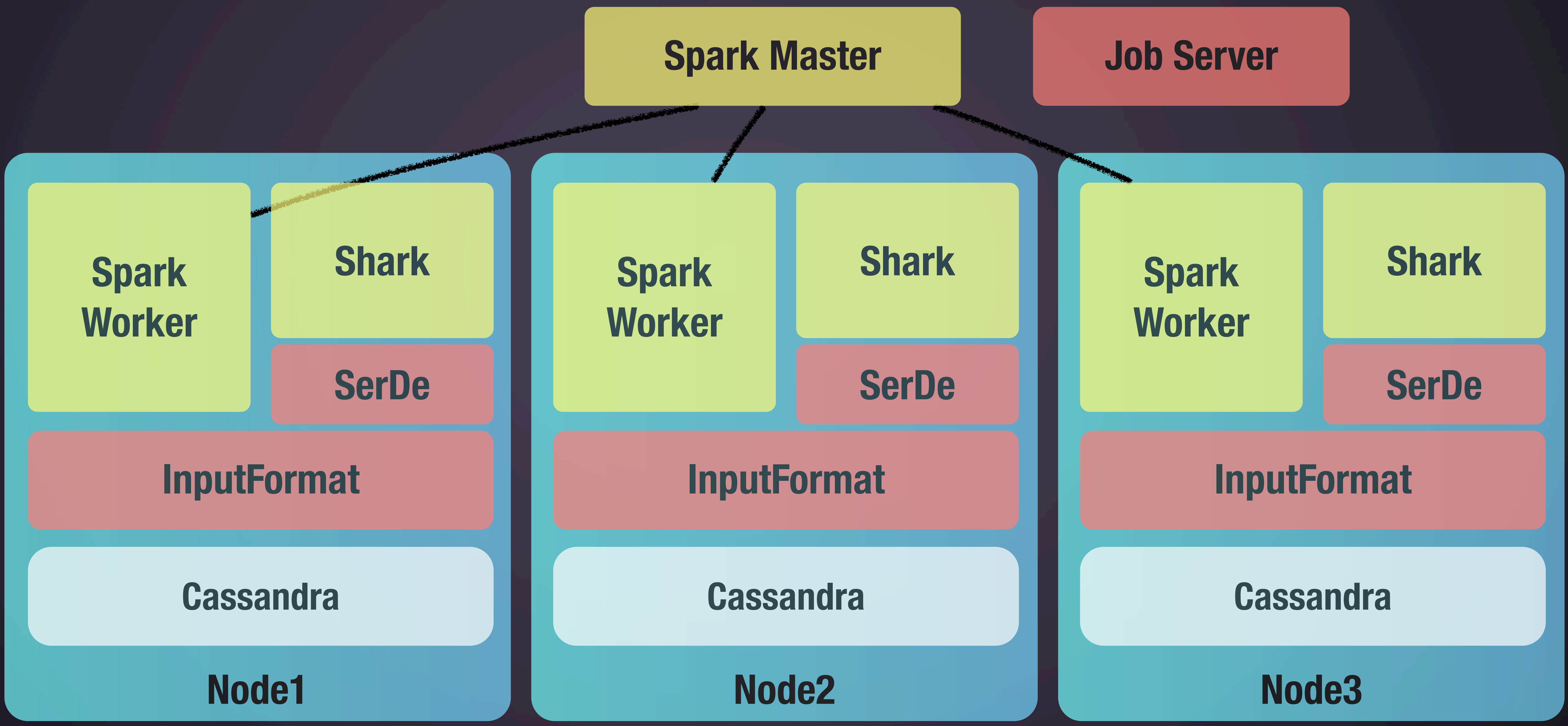
- Fewer platforms == lower TCO
- Much higher code sharing/reuse
- Spark/Shark/Streaming can replace Hadoop, Storm, and Impala
- Integration with Mesos, YARN helps

OUR SPARK ARCHITECTURE

From raw events to fast queries



Our Spark/Shark/Cassandra Stack



CASSANDRA SCHEMA

Event CF

	t0	t1	t2	t3	t4
2013-04-05 T00:00Z#id1	`{event0: a0}`	`{event1: a1}`	`{event2: a2}`	`{event3: a3}`	`{event4: a4}`

EventAttr CF

	ipaddr: 10.20.30.40:t1	videoId:45678:t1	providerId:500:t0
2013-04-05 T00:00Z#id1			

INPUTFORMAT VS RDD

- You can easily use InputFormats in Spark using newAPIHadoopRDD().
- Writing a custom RDD could have saved us **lots** of time.

InputFormat	RDD
Supports Hadoop, HIVE, Spark, Shark	Spark / Shark only
Have to implement multiple classes - InputFormat, RecordReader, Writeable, etc. Clunky API.	One class - simple API.
Two APIs, and often need to implement both (HIVE needs older...)	Just one API.

UNPACKING RAW EVENTS

	t0	t1
2013-04-05T00:00Z#i	{video: 10, type:5}	{video: 11, type:1}
2013-04-05T00:00Z#id	{video: 20, type:5}	{video: 25, type: 9}

UserID	Video	Type
id1	10	5

UNPACKING RAW EVENTS

	t0	t1
2013-04-05 T00:00Z#id	{video: 10, type:5}	{video: 11, type:1}
2013-04-05 T00:00Z#id	{video: 20, type:5}	{video: 25, type: 9}

UserID	Video	Type
id1	10	5

id1	11	1

UNPACKING RAW EVENTS

	t0	t1
2013-04-05T00:00Z#id	{video: 10, type:5}	{video: 11, type: 1}
2013-04-05T00:00Z#i	{video: 20, type:5}	{video: 25, type:9}

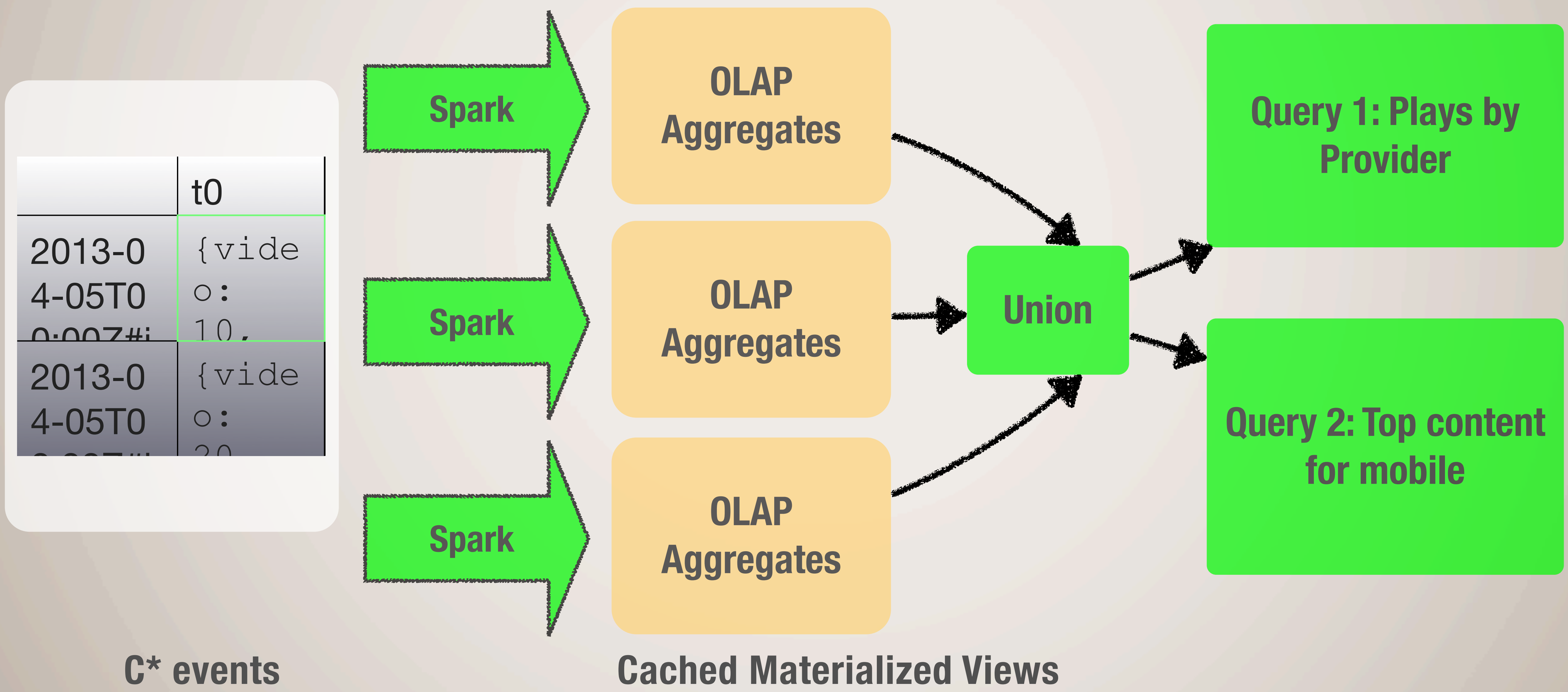
UserID	Video	Type
id1	10	5
id1	11	1
id2	20	5

UNPACKING RAW EVENTS

	t0	t1
2013-04-05 T00:00Z#id	{video: 10, type:5}	{video: 11, type: 1}
2013-04-05 T00:00Z#id	{video: 20, type:5}	{video: 25, type:9}

UserID	Video	Type
id1	10	5
id1	11	1
id2	20	5
id2	25	9

EXAMPLE: OLAP PROCESSING

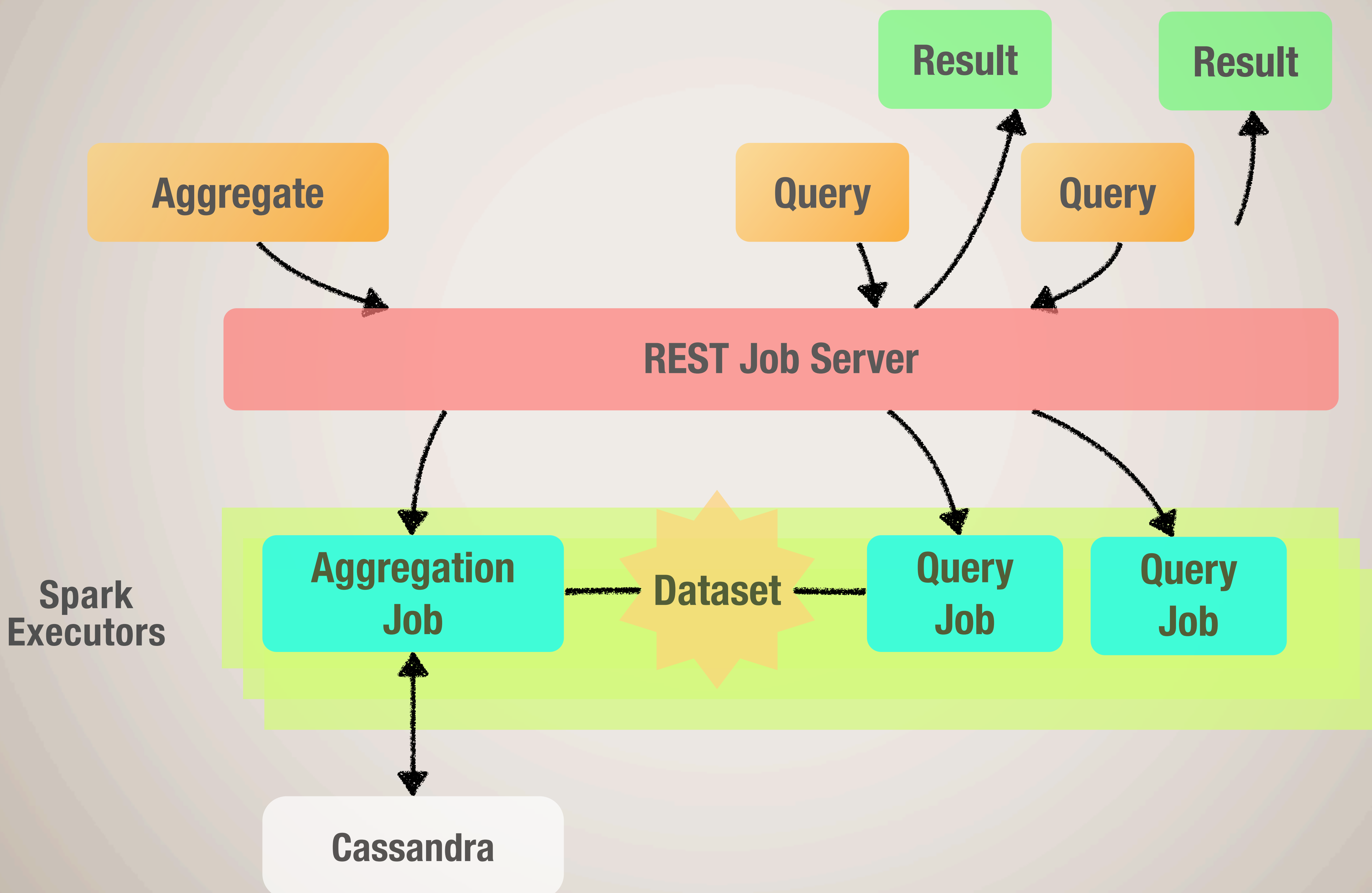


PERFORMANCE #'S

Spark: C* -> OLAP aggregates cold cache, 1.4 million events	130 seconds
C* -> OLAP aggregates warmed cache	20-30 seconds
OLAP aggregate query via Spark (56k records)	60 ms

6-node C*/DSE 1.1.9 cluster, Spark 0.7.0

OLAP WORKFLOW



FAULT TOLERANCE

- Cached dataset lives in Java Heap only - what if process dies?
- Spark lineage - automatic recomputation from source, but this is expensive!
  - Can also replicate cached dataset to survive single node failures
- Persist materialized views back to C*, then load into cache -- now recovery path is much faster

SPARK JOB SERVER

JOB SERVER OVERVIEW

- Spark as a Service - Job, Jar, and Context management
- Run ad-hoc Spark jobs
- Great support for sharing cached RDDs across jobs and low-latency jobs
- Works with Standalone Spark as well as Mesos
- Jars and job history is persisted via pluggable API
- Async and sync API, JSON job results
- Contributing back to Spark community in the near future

EXAMPLE JOB SERVER JOB

```
/**
 * A super-simple Spark job example that implements the SparkJob trait and
 * can be submitted to the job server.
 */
object WordCountExample extends SparkJob {
  override def validate(sc: SparkContext, config: Config): SparkJobValidation = {
    Try(config.getString(“input.string”))
      .map(x => SparkJobValid)
      .getOrElse(SparkJobInvalid(“No input.string”))
  }

  override def runJob(sc: SparkContext, config: Config): Any = {
    val dd = sc.parallelize(config.getString(“input.string”).split(" ").toSeq)
    dd.map((_, 1)).reduceByKey(_ + _).collect().toMap
  }
}
```

SUBMITTING AND RUNNING A JOB

- `curl --data-binary @../target/mydemo.jar localhost:8090/jars/demo`

```
OK[11:32 PM] ~
```

- `curl -d "input.string = A lazy dog jumped mean dog" 'localhost:8090/jobs?appName=demo&classPath=WordCountExample&sync=true'`


```
{
  "status": "OK",
  "RESULT": {
    "lazy": 1,
    "jumped": 1,
    "A": 1,
    "mean": 1,
    "dog": 2
  }
}
```

THANK YOU

And YES, We're HIRING!!

ooyala.com/careers