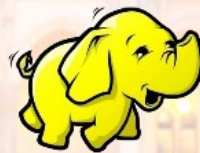




Cassandra



C* rocks on...

- high/fast write throughput
 - schema free design (get rid of JOINS!)
 - time-to-live on data
 - size of data
 - total load
 - uptime
 - tunable consistency vs availability
- } scale out

a few uses...

- #1 Users Search History
- #2 Fraud detection

- schema free design (get rid of JOINS!)
 - time-to-live on data
 - size of data
 - total load
 - uptime
 - tunable consistency vs availability
- } scale out

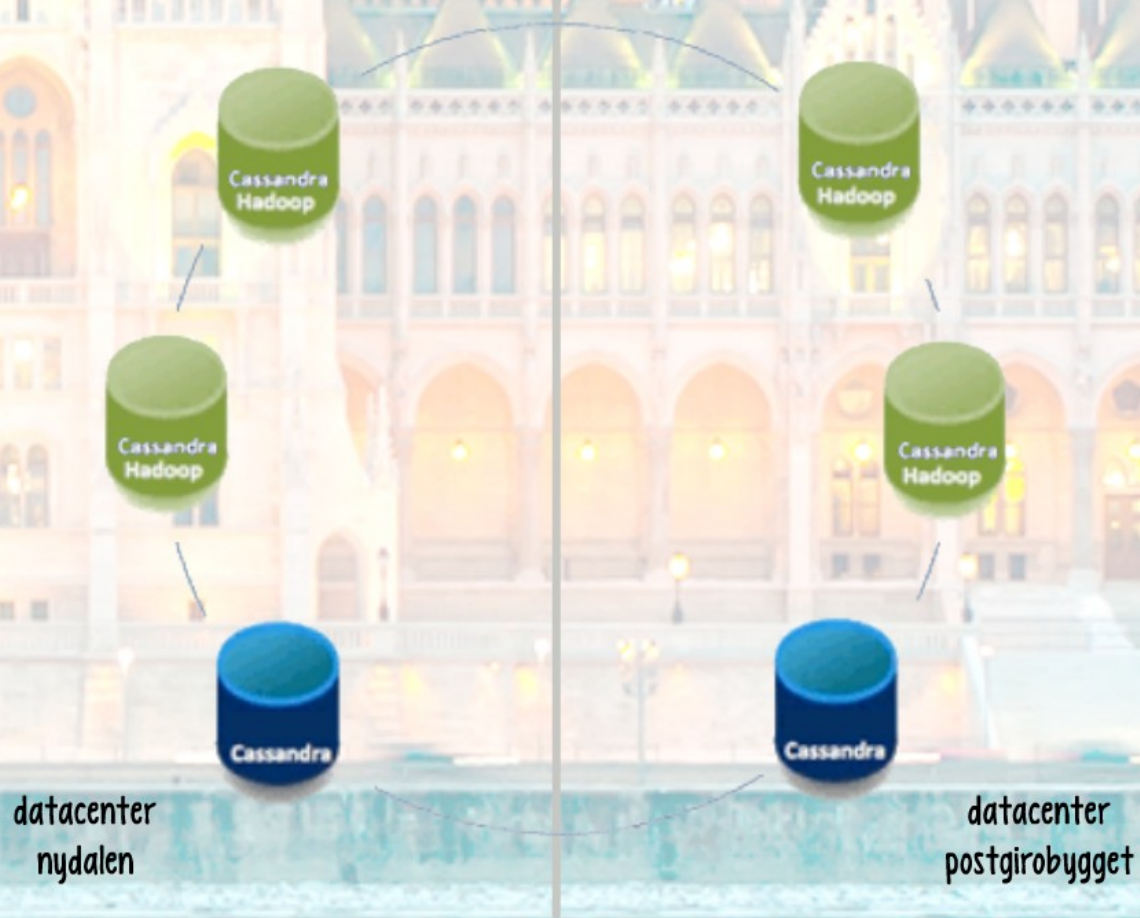
a few uses...

- #1 Users Search History
- #2 Fraud detection
- #3 IP-to-Geography
- #4 Message Inbox
- #5 Microservices metrics
- #6 Event Statistics

MACHINE SPECS

- 24 CPU
- 50 Gb RAM
- 5.5 Tb disks RAID50
- 100Gb SSD (commit logs)
- 100Gb SSD (HDFS)
- NOOP IO kernel scheduler

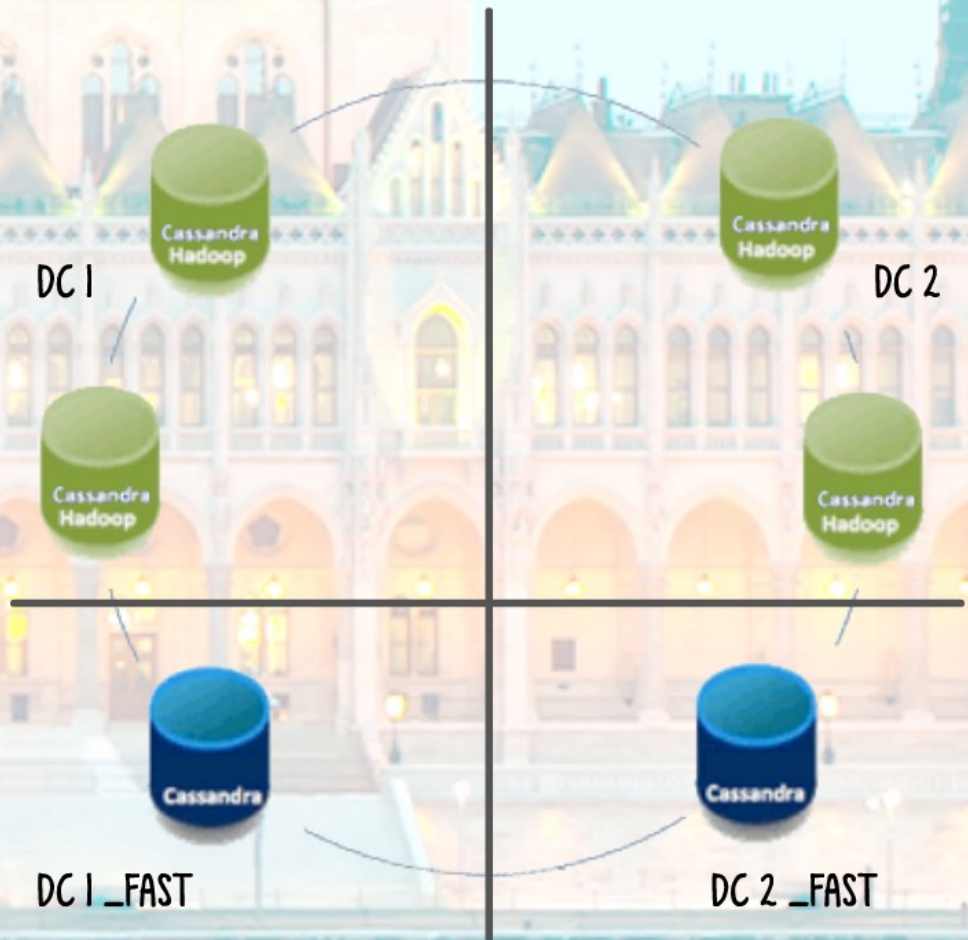
cassandra-2.0.11



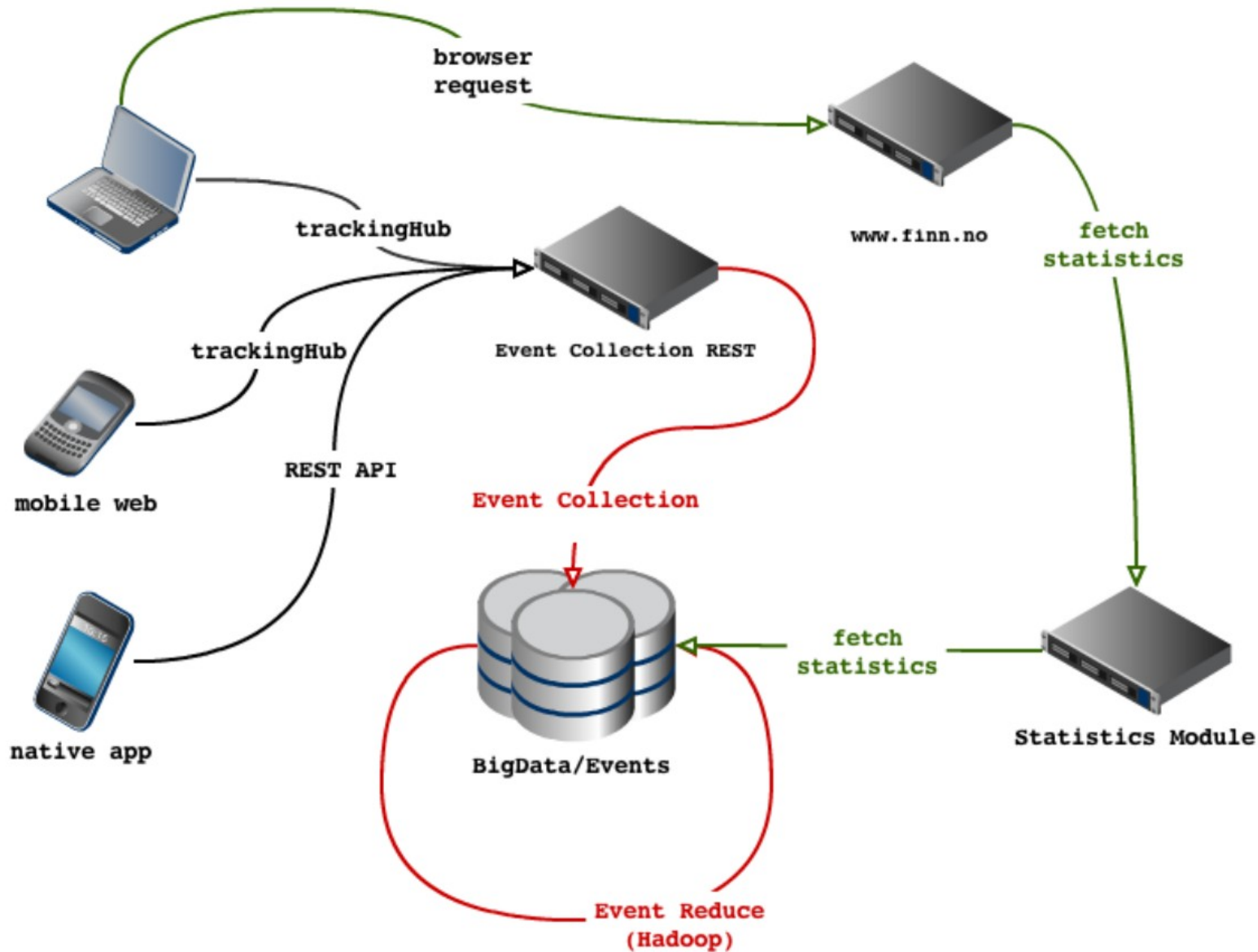
MACHINE SPECS

- 24 CPU
- 50 Gb RAM
- 5.5 Tb disks RAID50
- 100Gb SSD (commit logs)
- 100Gb SSD (HDFS)
- NOOP IO kernel scheduler

cassandra-2.0.11

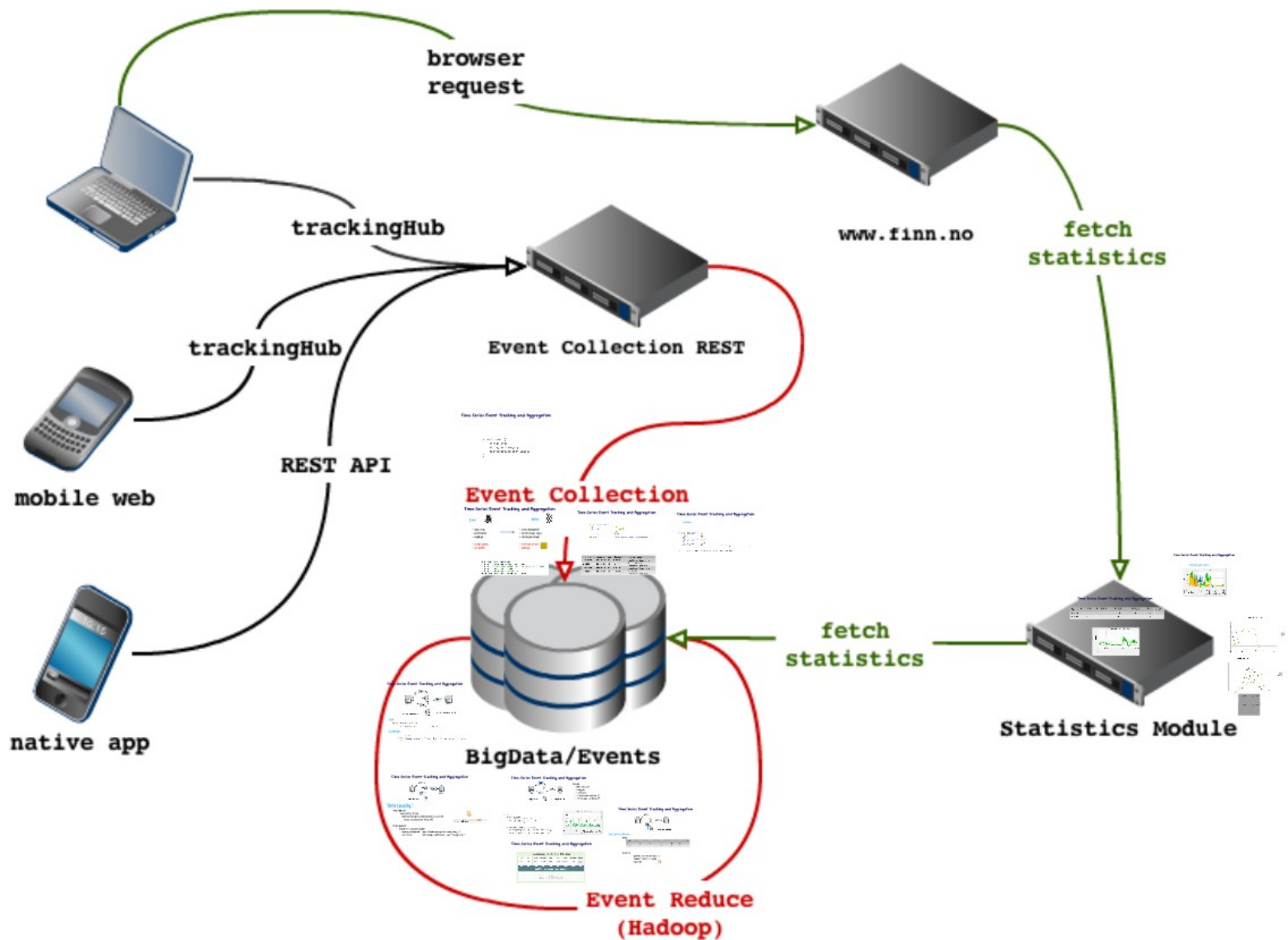


Time-Series Event Tracking and Aggregation



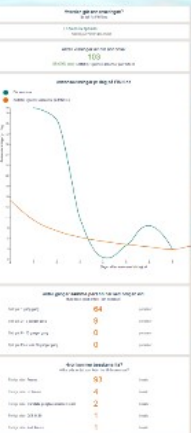
Event and **Statistics**
(Command) (Query separation)

Time-Series Event Tracking and Aggregation



Event and **Statistics**
 (Command) ————— (Query separation)

tion



b

Event Collection REST

API

Time-Series Event Tracking and Aggregation

```

struct Event {
  string type;
  string subCategory;
  map<string, string> values;
}

```

Event Collection

Time-Series Event Tracking and Aggregation

Scribe



- simple logs
- decentralized
- simple ops

- archaic options
- hot buffers

Kafka



- active development
- clustered (logic logs)
- stream processing

- a lot more servers
- no buffers

```

Properties props = new Properties();
props.put("message.serializer", "org.apache.kafka.serializer.StringSerializer");
props.put("key.serializer.class", "kafka.serializer.StringEncoder");
props.put("producer.type", "sync");
props.put("request.timeout.ms", 1000);
return new ProducerBuilder<ProducerConfig>(props);

```

Time-Series Event Tracking and Aggregation

```

CREATE TABLE events (
  timestamp INT,
  type VARCHAR(255),
  subcategory VARCHAR(255),
  values MAP<VARCHAR(255), VARCHAR(255)>
)

```

timestamp	type	subcategory	key_value
13912400	click	pageview	pageview
13912400	click	pageview	pageview
13912400	click	pageview	pageview
13912400	click	pageview	pageview

Time-Series Event Tracking and Aggregation

Final view

```

SELECT timestamp, type, subcategory,
       key_value
FROM events
GROUP BY timestamp, type, subcategory,
         key_value

```



Time-Series Event Tracking and Aggregation



fe
stat

Time-Series Event Tracking and Aggregation

```
struct Event {  
    string type;  
    string subCategory;  
    map<string,string> values;  
}
```

Time-Series Event Tracking and Aggregation

Scribe



- async msgs
- decentralised
- simple ops
- **archaic options**
- **lost buffers**



Kafka



- active development
- clustered (sync msgs)
- stream processing!
- **a lot more servers**
- **zookeeper**



```
Properties props = new Properties();
props.put("metadata.broker.list", BROKER_LIST);
props.put("key.serializer.class", "kafka.serializer.StringEncoder");
props.put("producer.type", "sync");
props.put("request.required.acks", "-1");
return new Producer<>(new ProducerConfig(props));
```

Time-Series Event Tracking and Aggregation

```
CREATE TABLE events (  
  timebucket          text,  
  timestamp           timeuuid,  
  type                text,  
  subcategory         text,  
  keys_and_values     text,  
  PRIMARY KEY        (timebucket, type, timestamp)  
);
```

timebucket_row	timestamp	type	subcategory	keys_and_values
1369164000	1369164000	AD	PageView	{"ad.user":"711130430", "user.uniqueVisitorId":"1799736047",...}
1369164000	1369164001	AD	PageView	{"ad.user":"711130430", "user.uniqueVisitorId":"1799736047",...}
1369164000	1369164002	AD	EmailSent	{"ad.user":"711130430", "user.uniqueVisitorId":"1799736047",...}
1369164000	1369164003	AD	MarkedFavourite	{"ad.user":"711130430", "user.uniqueVisitorId":"1799736047",...}

Time-Series Event Tracking and Aggregation

the real world...

```
CREATE TABLE events (  
  real_timebucket      text,  
  partition            int,  
  type                 text,  
  collected_timestamp  timeuuid,  
  real_timestamp       timeuuid,  
  subcategory          text,  
  keys_and_values      text,  
  PRIMARY KEY         ((real_timebucket, partition), type, collected_timestamp)  
);  
  
CREATE INDEX collected_ts_idx ON events (collected_timestamp);
```



Time-Series Event Tracking and Aggregation

```
CREATE TABLE events (  
  timebucket          text,  
  timestamp           timeuuid,  
  type                text,  
  subcategory         text,  
  keys_and_values     text,  
  PRIMARY KEY        (timebucket, type, timestamp)  
);
```

timebucket_row	timestamp	type	subcategory	keys_and_values
1369164000	1369164000	AD	PageView	{"ad.user":"711130430", "user.uniqueVisitorId":"1799736047",...}
1369164000	1369164001	AD	PageView	{"ad.user":"711130430", "user.uniqueVisitorId":"1799736047",...}
1369164000	1369164002	AD	EmailSent	{"ad.user":"711130430", "user.uniqueVisitorId":"1799736047",...}
1369164000	1369164003	AD	MarkedFavourite	{"ad.user":"711130430", "user.uniqueVisitorId":"1799736047",...}

Event Collection

Time-Series Event Tracking and Aggregation

Scribe  **Kafka** 

- async msgs
- decentralized
- simple ops
- archaic options
- lots buffers

→

- active development
- clustered (sync msgs)
- stream processing
- a lot more servers
- zookeeper

```

    Processor proc = new Processor();
    proc.add("webdata.hadoop.topic", "LOCAL");
    proc.add("log_serializer.class", "kafka.serializer.StringEncoder");
    proc.add("producer.topic", "test");
    proc.add("zookeeper.connect", "127.0.0.1");
    return new ProcessorBuilder(ProcessorConfig.DEFAULT);
    
```

Time-Series Event Tracking and Aggregation

```

    @SuppressWarnings("rawtypes")
    @Table(name = "events")
    @TableType("DELTA")
    @TableIndex(keys = {"timebucket", "type", "timeview"}, values = {"timebucket", "type", "timeview"})
    public class Events {
        // ...
    }
    
```

timebucket_id	timebucket	type	timeview	event_value
100014880	100014880	AD	pageview	USER:1000000
100014880	100014880	AD	pageview	USER:1000000
100014880	100014880	AD	pageview	USER:1000000
100014880	100014880	AD	pageview	USER:1000000
100014880	100014880	AD	pageview	USER:1000000
100014880	100014880	AD	pageview	USER:1000000
100014880	100014880	AD	pageview	USER:1000000
100014880	100014880	AD	pageview	USER:1000000
100014880	100014880	AD	pageview	USER:1000000
100014880	100014880	AD	pageview	USER:1000000

Time-Series Event Tracking and Aggregation

```

    @Table(name = "events")
    @TableType("DELTA")
    @TableIndex(keys = {"timebucket", "type", "timeview"}, values = {"timebucket", "type", "timeview"})
    public class Events {
        // ...
    }
    
```



fetch
statistics

BigData/Events

Time-Series Event Tracking and Aggregation

Splits

```

    @Table(name = "events")
    @TableType("DELTA")
    @TableIndex(keys = {"timebucket", "type", "timeview"}, values = {"timebucket", "type", "timeview"})
    public class Events {
        // ...
    }
    
```

RecordReader

```

    @Table(name = "events")
    @TableType("DELTA")
    @TableIndex(keys = {"timebucket", "type", "timeview"}, values = {"timebucket", "type", "timeview"})
    public class Events {
        // ...
    }
    
```

Time-Series Event Tracking and Aggregation

Data Locality!

Simple Approach

```

    @Table(name = "events")
    @TableType("DELTA")
    @TableIndex(keys = {"timebucket", "type", "timeview"}, values = {"timebucket", "type", "timeview"})
    public class Events {
        // ...
    }
    
```

Proper approach

```

    @Table(name = "events")
    @TableType("DELTA")
    @TableIndex(keys = {"timebucket", "type", "timeview"}, values = {"timebucket", "type", "timeview"})
    public class Events {
        // ...
    }
    
```

Time-Series Event Tracking and Aggregation

Each day

- 300+ minute jobs
- 7 daily jobs
- 7 hourly jobs
- 1 billion records read from C1
- 60M records written to C1

Time-Series Event Tracking and Aggregation

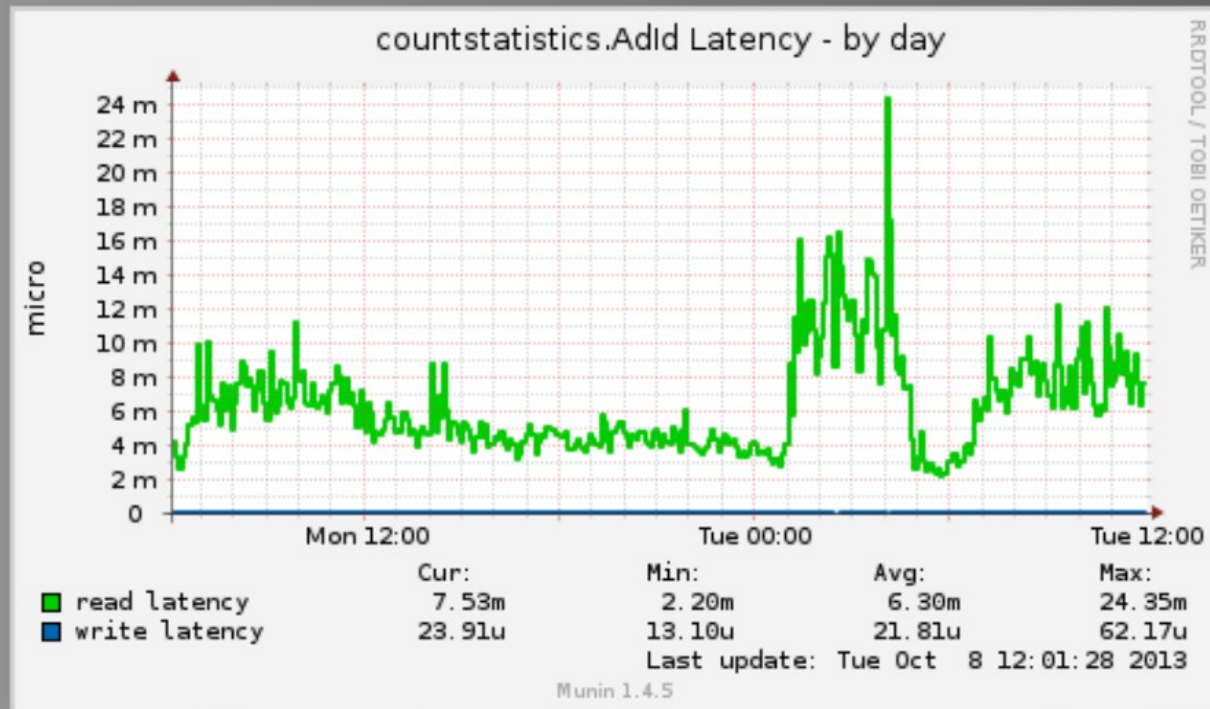
User Centric Statistics

USER	VIEW	CLICK	ADD TO CART	CHECKOUT	REORDER
1000000	1000000	1000000	1000000	1000000	1000000

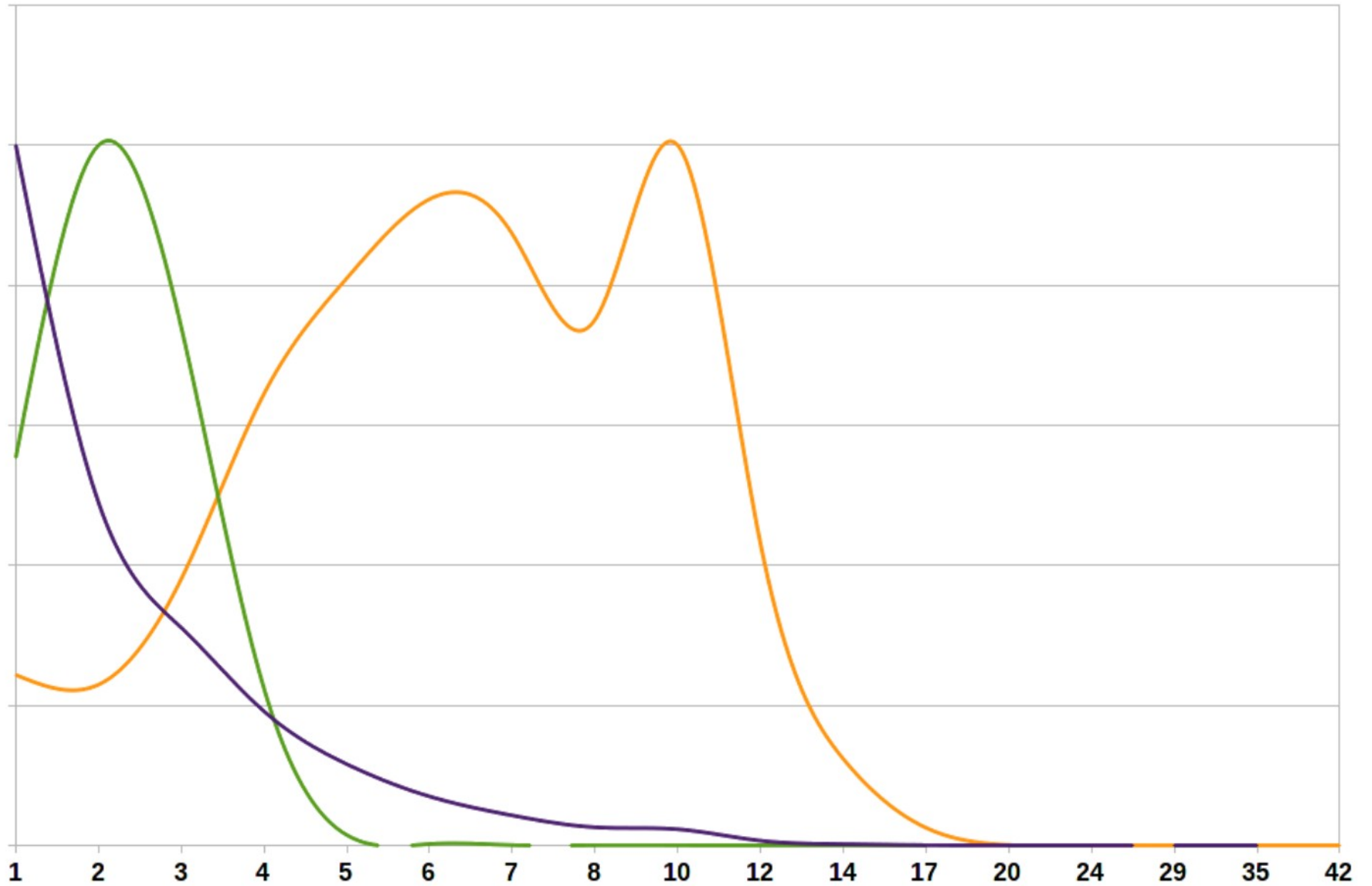
Time-Series Event Tracking and Aggregation

Time-Series Event Tracking and Aggregation

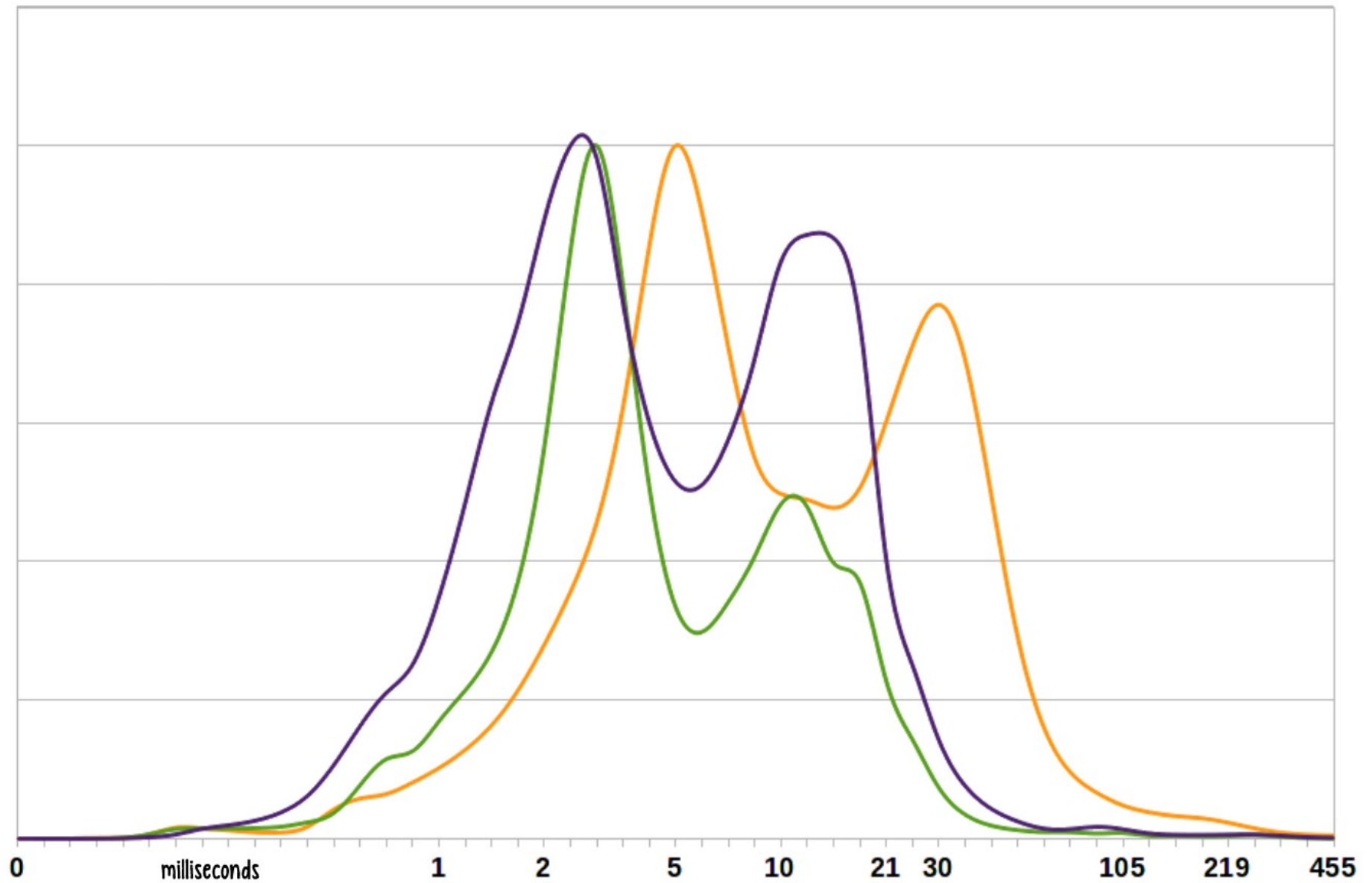
	HOUR_201328111: PageView	HOUR_201328112: EmailSent	DAY_2012338: PageView	DAY_2012339: EmailSent	TOTAL: PageView	TO Email
119	10	2	20	4	50	
052	23	3	102	4	234	

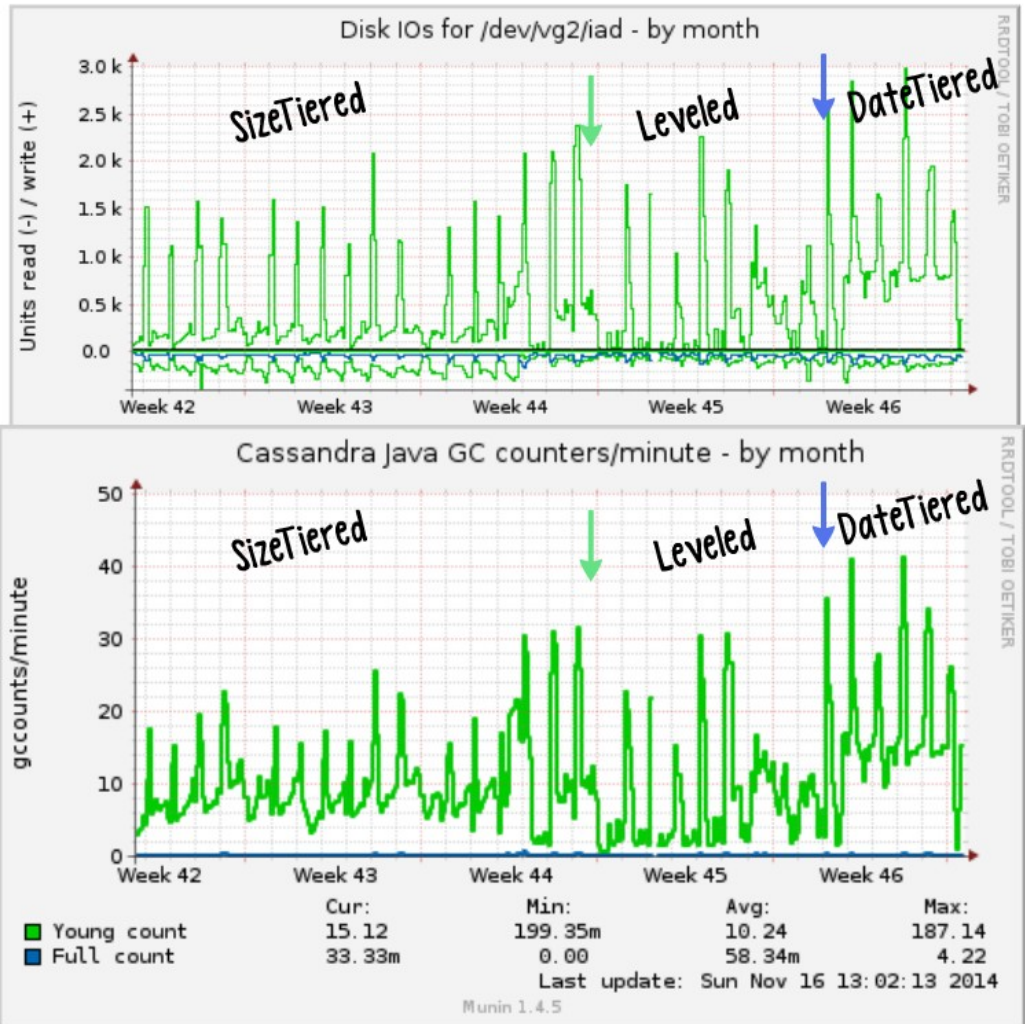
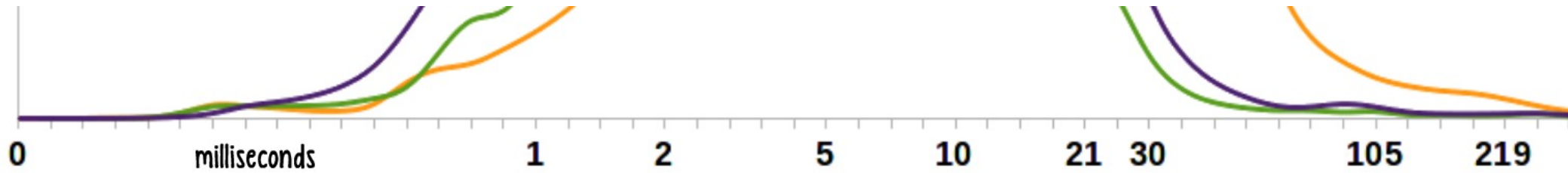


SSTables per Read





Read Latency





Event Collection

Time-Series Event Tracking and Aggregation

Scribe  **Kafka** 

- async msgs
- decentralized
- simple ops
- archaic options
- lots buffers

→

- active development
- clustered (sync msgs)
- stream processing
- a lot more servers
- zookeeper

```

    Processor proc = new Processor();
    proc.add("webdata.hadoop.topic", "LOCAL");
    proc.add("log_serializer.class", "kafka.serializer.StringEncoder");
    proc.add("producer.topic", "test");
    proc.add("zookeeper.connect", "127.0.0.1");
    return new ProcessorBuilder(ProcessorConfig.DEFAULT);
    
```

Time-Series Event Tracking and Aggregation

```

    CREATE TABLE IF NOT EXISTS events (
      timestamp INT,
      bucket INT,
      type INT,
      value INT,
      key_and_value VARCHAR(255)
    );
    
```

timestamp	bucket	type	value	key_and_value
1387148800	1387148800	40	1000000	1000000
1387149200	1387149200	40	1000000	1000000
1387149600	1387149600	40	1000000	1000000
1387150000	1387150000	40	1000000	1000000
1387150400	1387150400	40	1000000	1000000

Time-Series Event Tracking and Aggregation

```

    SELECT time_bucket(
      '1 hour',
      timestamp,
      bucket,
      type,
      value,
      key_and_value
    ) AS bucket,
    SUM(value) AS sum_value
    FROM events
    GROUP BY bucket, sum_value;
    
```



fetch
statistics

Time-Series Event Tracking and Aggregation

Splits

```

    @Override public void run() {
      // ...
    }
    
```

RecordReader

```

    public RecordReader() {
      // ...
    }
    
```

BigData/Events

Time-Series Event Tracking and Aggregation

Data Locality!

Simple Approach

```

    hdfs://localhost:9000/...
    
```

Proper approach

```

    hdfs://localhost:9000/...
    
```

Time-Series Event Tracking and Aggregation

Each day

- 300+ minute jobs
- 7 daily jobs
- 7 hourly jobs
- 1000 records read from C*
- 500 records written to C*

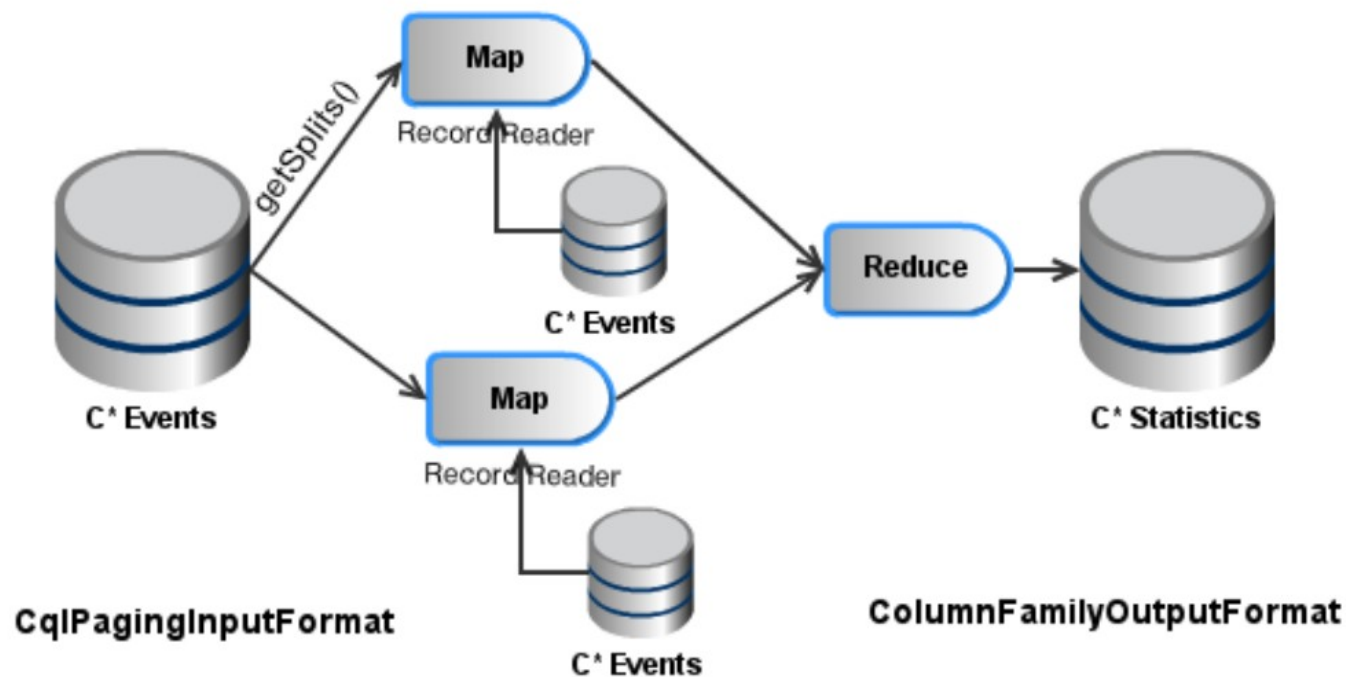
Time-Series Event Tracking and Aggregation

User Centric Statistics

Category	Value	Value	Value	Value	Value
USER	10	20	30	40	50
...

Time-Series Event Tracking and Aggregation

Time-Series Event Tracking and Aggregation



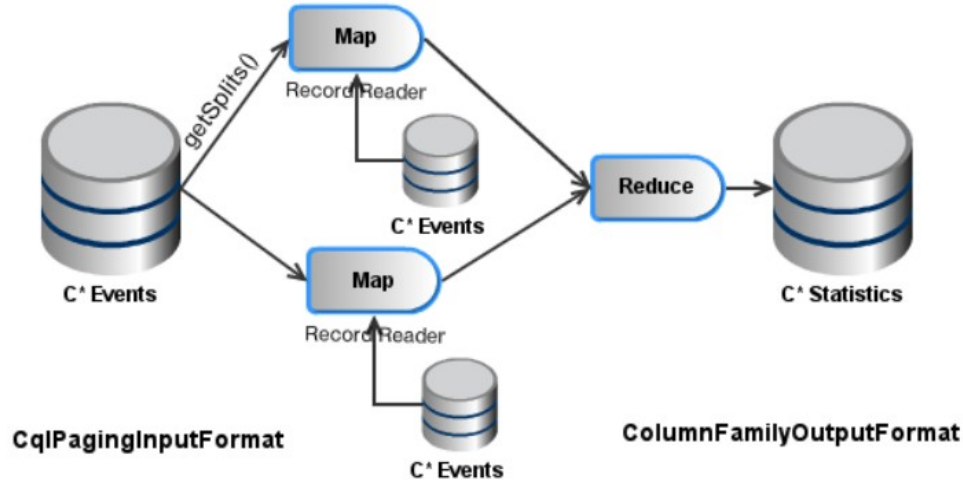
splits

```
SELECT timebucket FROM events
WHERE type = 'AD'
AND timebucket IN ('startMinute..endMinute')
```

RecordReader

```
SELECT * FROM events
WHERE timebucket = 'currentMinute' AND type = 'AD'
AND timestamp >= minTimeuuid('start') AND timestamp < maxTimeuuid('endTime')
LIMIT 100
```

Time-Series Event Tracking and Aggregation



Data Locality !

Simple Approach:

favour localhost location

`TokenAwarePolicy(DcAwareRoundRobinPolicy(local-dc))`

and use consistency level LOCAL_ONE

Proper approach:

hadoop-2.2 & Capacity Scheduler

capacity-scheduler.xml

core-site.xml

`"yarn.scheduler.capacity.node-locality-delay = 3"`

`"net.topology.script.file.name = <your-topology-script>"`



Cluster		Cluster Metrics	
About	Nodes	Apps	AppD
Submitted	16572	Submitted	2
Applications			
NEW	NEW SAVING	Showing 1 to 4 of 4 entries	
SUBMITTED	ACCEPTED	Rack	Node ID
BUILDING	REMOVING	/dc1/rack1	RUNNING
EMERGING	FINISHED	/dc1/rack1	RUNNING
KILLED	FAILED	/dc2/rack1	RUNNING
Scheduler		/dc2/rack3	RUNNING

	Name
Job Counters	Data-local map tasks
	Launched map tasks
	Launched reduce tasks
	Rack-local map tasks



▼ Cluster

- [About](#)
- [Nodes](#)
- [Applications](#)
 - [NEW](#)
 - [NEW_SAVING](#)
 - [SUBMITTED](#)
 - [ACCEPTED](#)
 - [RUNNING](#)
 - [REMOVING](#)
 - [FINISHING](#)
 - [FINISHED](#)
 - [FAILED](#)
 - [KILLED](#)
- [Scheduler](#)

► Tools

Cluster Metrics

Apps Submitted	Apps Pending
16572	2

Show 20 entries

Rack	Node State
/dc1/rack1	RUNNING
/dc1/rack1	RUNNING
/dc2/rack1	RUNNING
/dc2/rack1	RUNNING

Showing 1 to 4 of 4 entries

Name
Data-local map tasks



Cluster

- About
- Nodes
- Applications
 - NEW
 - NEW_SAVING
 - SUBMITTED
 - ACCEPTED
 - RUNNING
 - REMOVING
 - FINISHING
 - FINISHED
 - FAILED
 - KILLED
- Scheduler

Tools

Cluster Metrics

Apps Submitted	Apps Pending
16572	2

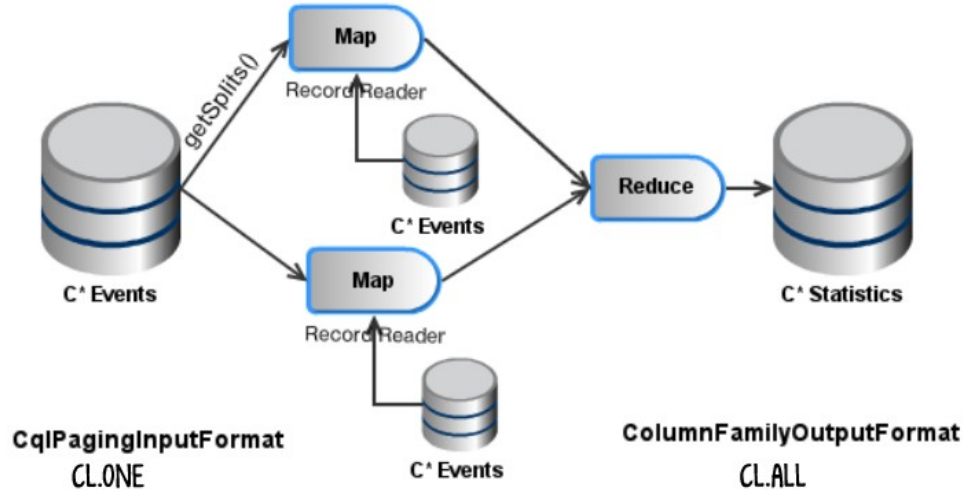
Show 20 entries

Rack	Node State
/dc1/rack1	RUNNING
/dc1/rack1	RUNNING
/dc2/rack1	RUNNING
/dc2/rack1	RUNNING

Showing 1 to 4 of 4 entries

	Name	Total
Job Counters	Data-local map tasks	15139
	Launched map tasks	15140
	Launched reduce tasks	1
	Rack-local map tasks	1

Time-Series Event Tracking and Aggregation



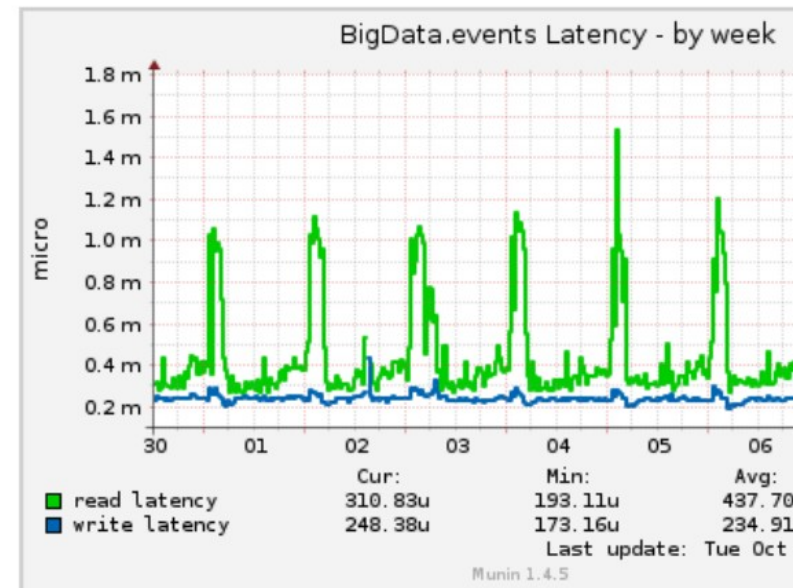
Each day:

- 5000+ minute jobs
- 7 daily jobs
- + ad-hoc jobs
- ~ 1 billion records read
- ~ 150M records written

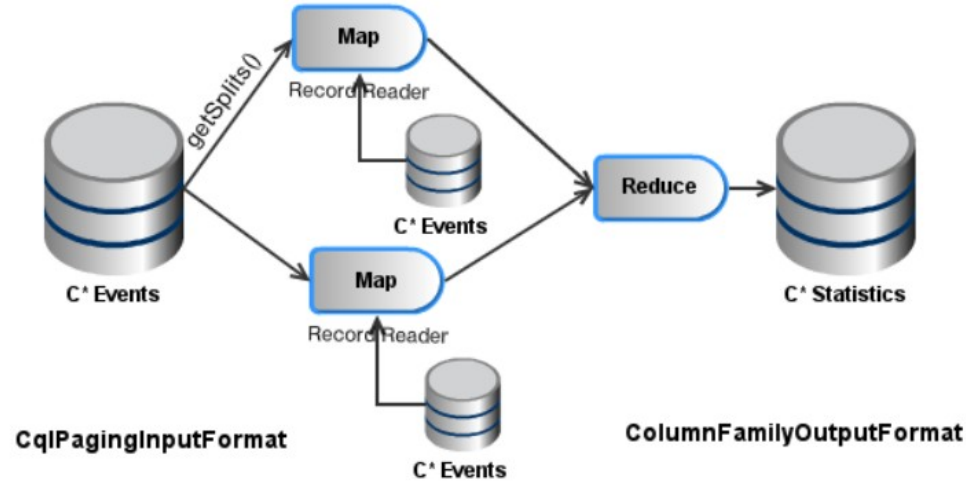
```

s):
event in events:
id = event.getValues()["ad.id"]
subcategory = (event.subcategory, 1)
emit(id, subcategory)

subcategories):
subcategory, count in subcategories:
incrementCassandraCounter(id, subcategory, count, HOUR_XX)
incrementCassandraCounter(id, subcategory, count, DAY_ZZZ)
incrementCassandraCounter(id, subcategory, count, TOTAL)
    
```



Time-Series Event Tracking and Aggregation



Metric Statistics

Simple...

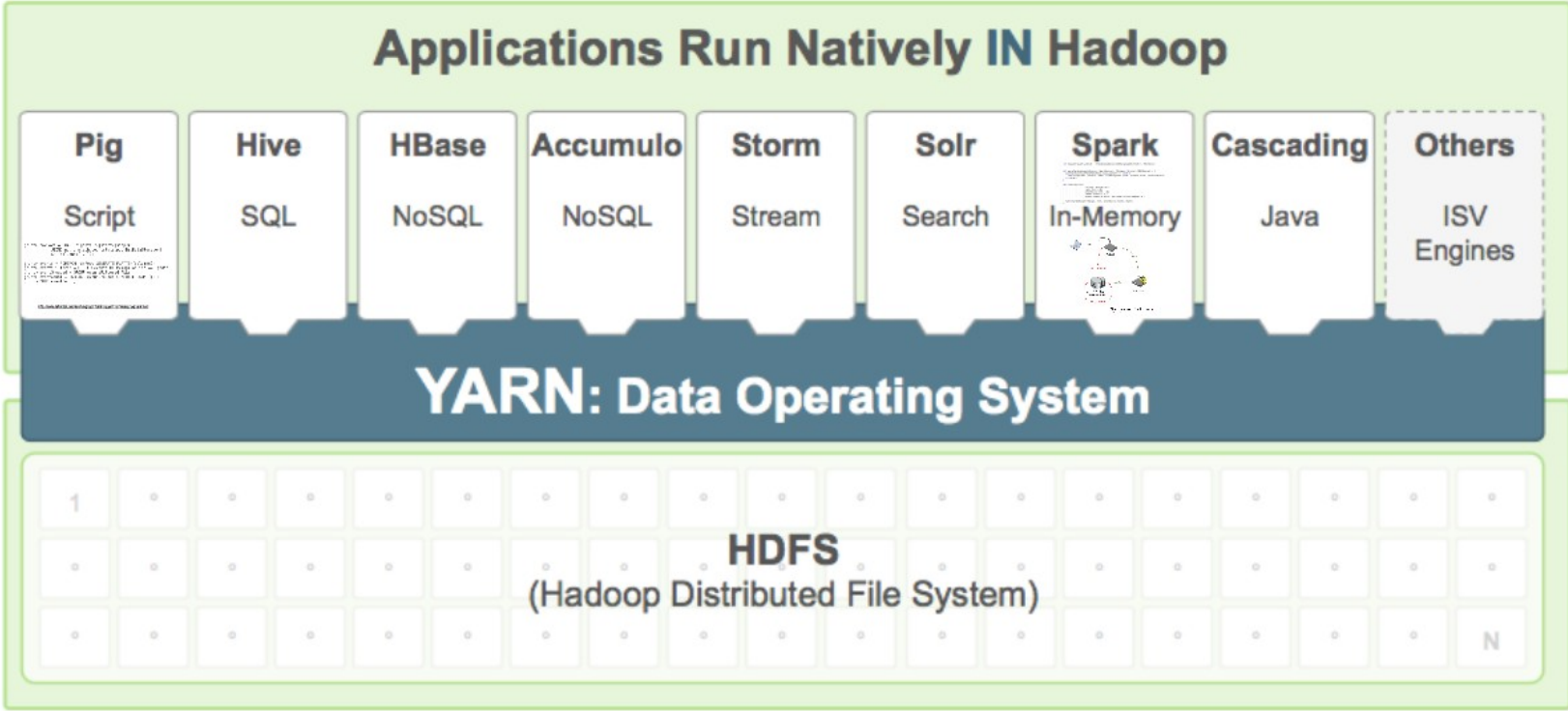
UserId	TOTAL:Ski	TOTAL:Java	TOTAL:Tesla	TOTAL:Bicycle	TOTAL:Oslo	TOTAL:Bergen
706119	10	2	20	4	50	5
706052	23	3	102	4	234	10

Advanced...

- graphing: user->ad, ad->user, etc
- Mahout ("Taste") --> Myrrix
- Spark ALS



Time-Series Event Tracking and Aggregation

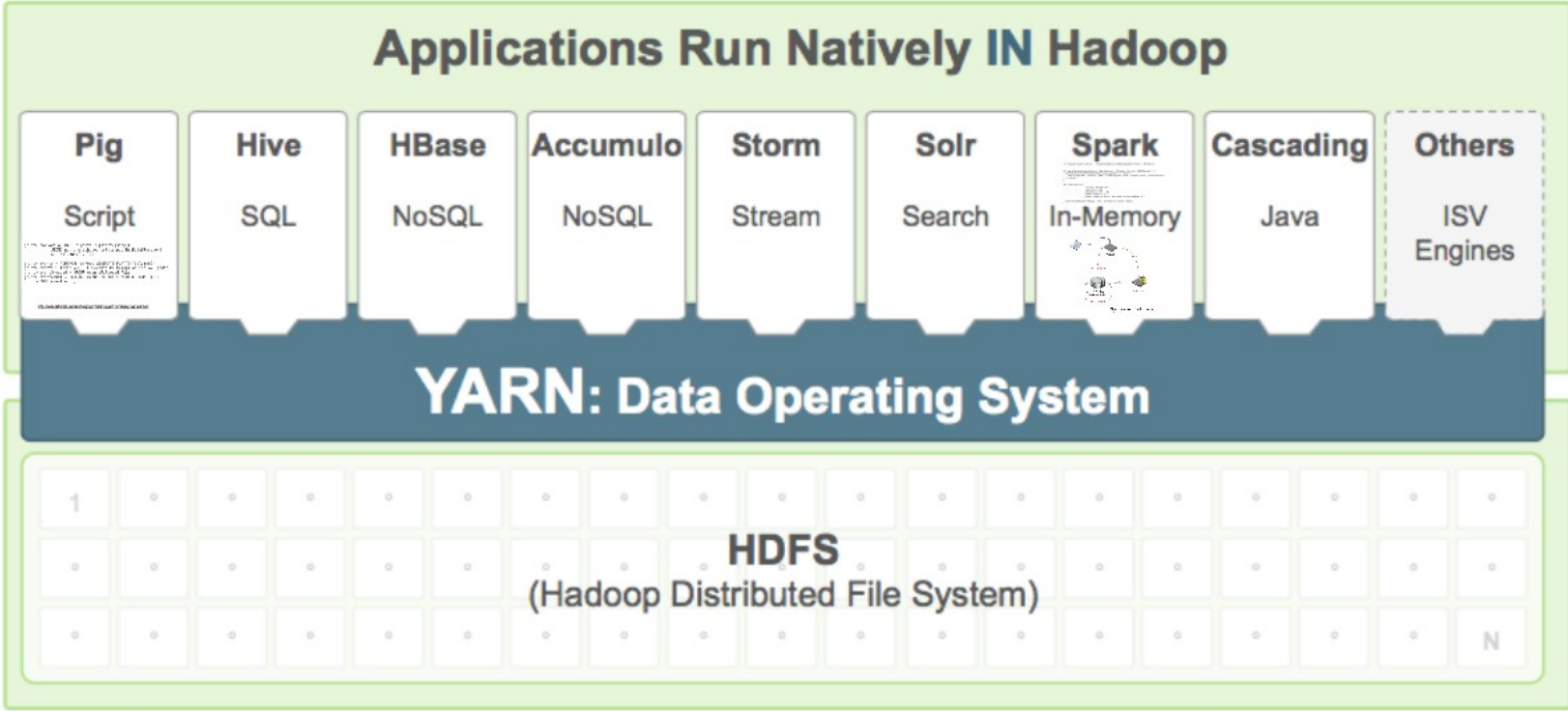


Script

```
grunt> rawRows = LOAD 'bigdata://$start/$stop/AD'  
                USING no.finntech.countstats.pig.BigDataStorage()  
                AS (columns:{...});  
  
grunt> events = FOREACH rawRows GENERATE FLATTEN(columns);  
grunt> eventsFiltered = FILTER events BY kvmap#'ad.id' == '$adid';  
grunt> eventGrouped = GROUP eventsFiltered ALL;  
grunt> eventCount = FOREACH eventGrouped GENERATE COUNT($1);  
grunt> DUMP eventCount;
```

<http://www.datastax.com/dev/blog/cql3-table-support-in-hadoop-pig-and-hive>

Time-Series Event Tracking and Aggregation



Spark

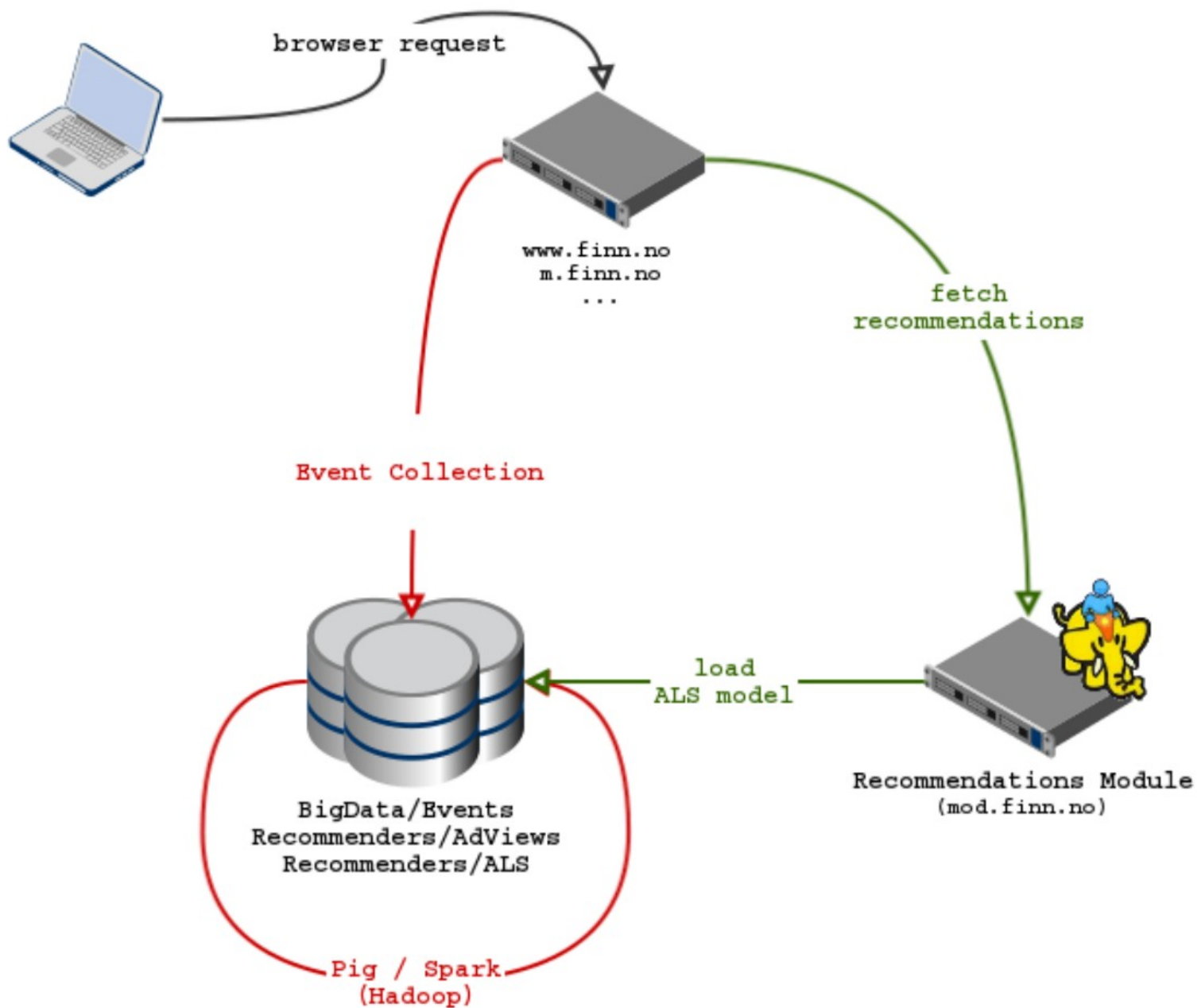
```
val als_matrix_for_mahout = trainImplicit(parseRatings(getContext(), fileName))

def parseRatings(sparkContext: SparkContext, fileName: String): RDD[Rating] = {
  sparkContext.textFile(fileName).map(_._split(",") match {
    case Array(user, product, rate) => Rating(user.toInt, product.toInt, rate.toDouble)
  }).cache()
}

def trainImplicit(
  ratings: RDD[Rating],
  rank: Int = 40,
  iterations: Int = 20,
  lamda: Double = 0.1,
  alpha: Double = 60.0): MatrixFactorizationModel = {
  ALS.trainImplicit(ratings, rank, iterations, lamda, alpha)
}
```

In-Memory





Recommendations

(Command

Query separation)

C* is moving way quick!

- Secondary Indexes
- Compression
- CQL3 (+MapReduce +Spark)
- CQL tracing
- Automatic pagination
- async and unlogged batch statements
- fluent api to cql java driver
- Counters-2 !! (Cassandra-2.1)

lessons learnt

- During C* ops
 - plan carefully, test strategies, test clients,
 - easy to bump CL.ONE -> CL.QUORUM,
 - stop Hadoop
- Heed disk latency + utilisation
 - >20% asking for trouble
 - SSD for commit-log, separate SSD for HDFS,
(Cassandra writes sequential)
- Comprehensive monitoring - detect problems quick
 - C* is robust and will hide problems
 - monitor gc, and pending actions growing,
 - look out for spikes in 95th percentile
- Stay under capacity!
 - easy backup, repair, streaming, etc
- Avoid OrderPreservingPartitioner
- Avoid custom serialised data (transparency is gold)
- Avoid skinny rows on non-commodity machines
- CQL3 rocks (always!)
 - Use json over maps (until you need maps)
- Don't run JobTracker+NameNode on a C* node
- Upgrading to vnodes tedious (~2 months)