Big Data Analytics on Container-Orchestrated Systems

Gerard Casas Saez

University of Colorado Boulder

July 20, 2017

Outline

Introduction

Problem statement

Related work

Approach

Demo

Evaluation & Results
- Scalability
- Maintenance
- Performance

Future work

Introduction

Why?

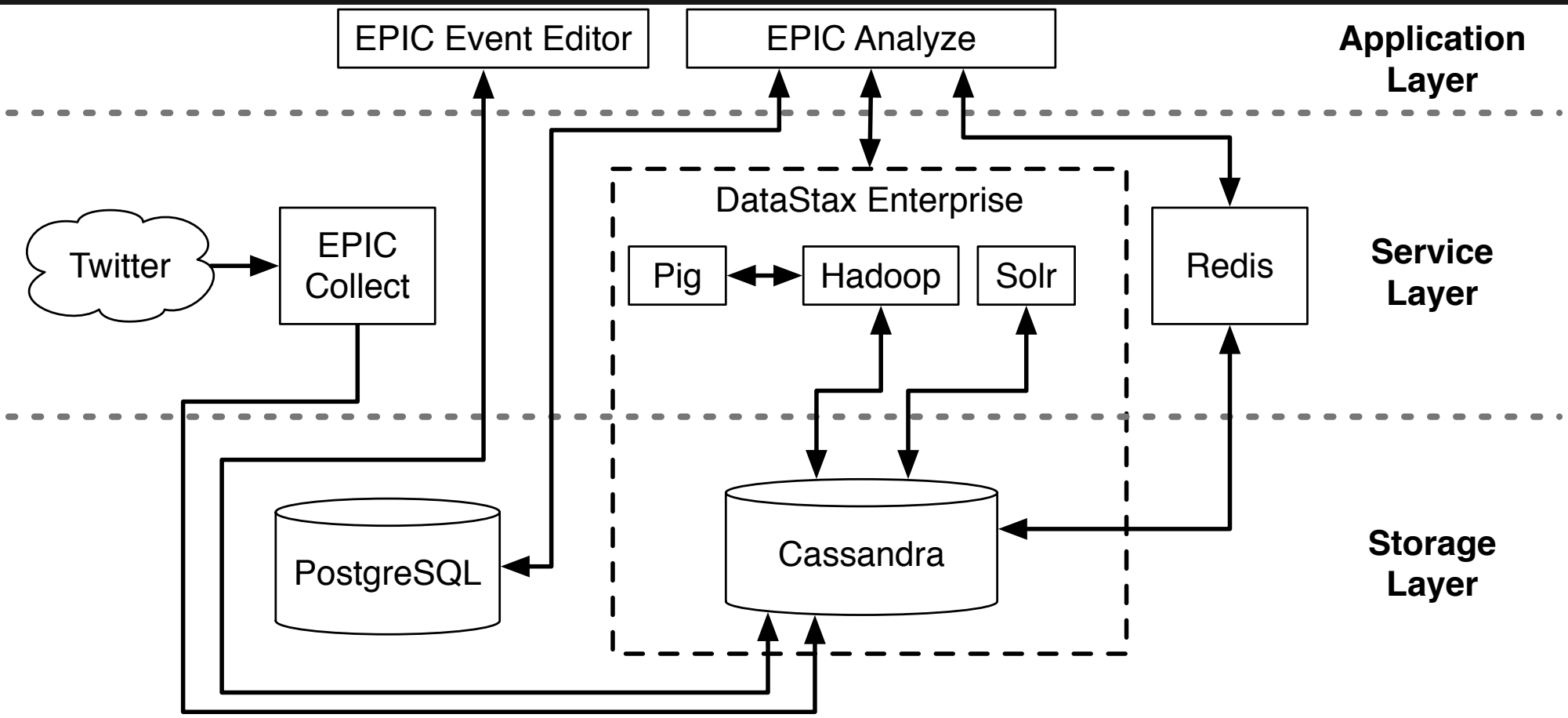
Introduction

- IOT & Social networks
- Increase internet traffic: 3x internet traffic by 2021
- Scale up Big Data Analytics System
- Keeping maintenance at low cost
- Container-orchestrated make infrastructure easier

Keep up with data growth

Background: Project EPIC

- EPIC Collect
- EPIC Analyze


EPIC Event Editor
EPIC Analyze
Application Layer
DataStax Enterprise
Twitter
EPIC Collect
Pig
Hadoop
Solr
Redis
Service Layer
PostgreSQL
Cassandra
Storage Layer

Background: Containerization

- Operating-system-level virtualization
- Use host machine system resources
- **Docker** most used alternative
- Development microservices

Background: Container-orchestration systems

- Container interaction abstraction
- Great to deploy microservices architectures
- Apache Mesos vs **Kubernetes**

Background: Microservices Architecture

- Small & specific
- Better scalability
- Loosely-coupled & highly-cohesive
- Orchestration <> **Coreography**

Background: Coreography microservice architecture

- Easier to extend
- Asyncronous
- PubSub interaction
- Messaging system: **Apache Kafka**

Problem statement

Problem statement

1. Advantages and/or limitations from existing infrastructure
   1.1 More reliable?
   1.2 More scalable?
2. Lower maintenance costs than the existing infrastructure?
   2.1 Easier to deploy?
   2.2 Easier to upgrade?
   2.3 More resilient to failures?

Related work

- SMACK: Spark, Mesos, Akka, Cassandra and Kafka [Raul Estrada et al. 2016]
- Hadoop ecosystem [Han Hu et al. 2014]
- Lambda architecture [Zirije Hasani et al. 2014]

Approach

Features

- Event management
- Real-time collection of streaming Twitter data
- Real-time classification of incoming tweets
- Data Analysis

Non-functional requirements

- Less code
- Easier deployment
- More flexible
- Better scalability

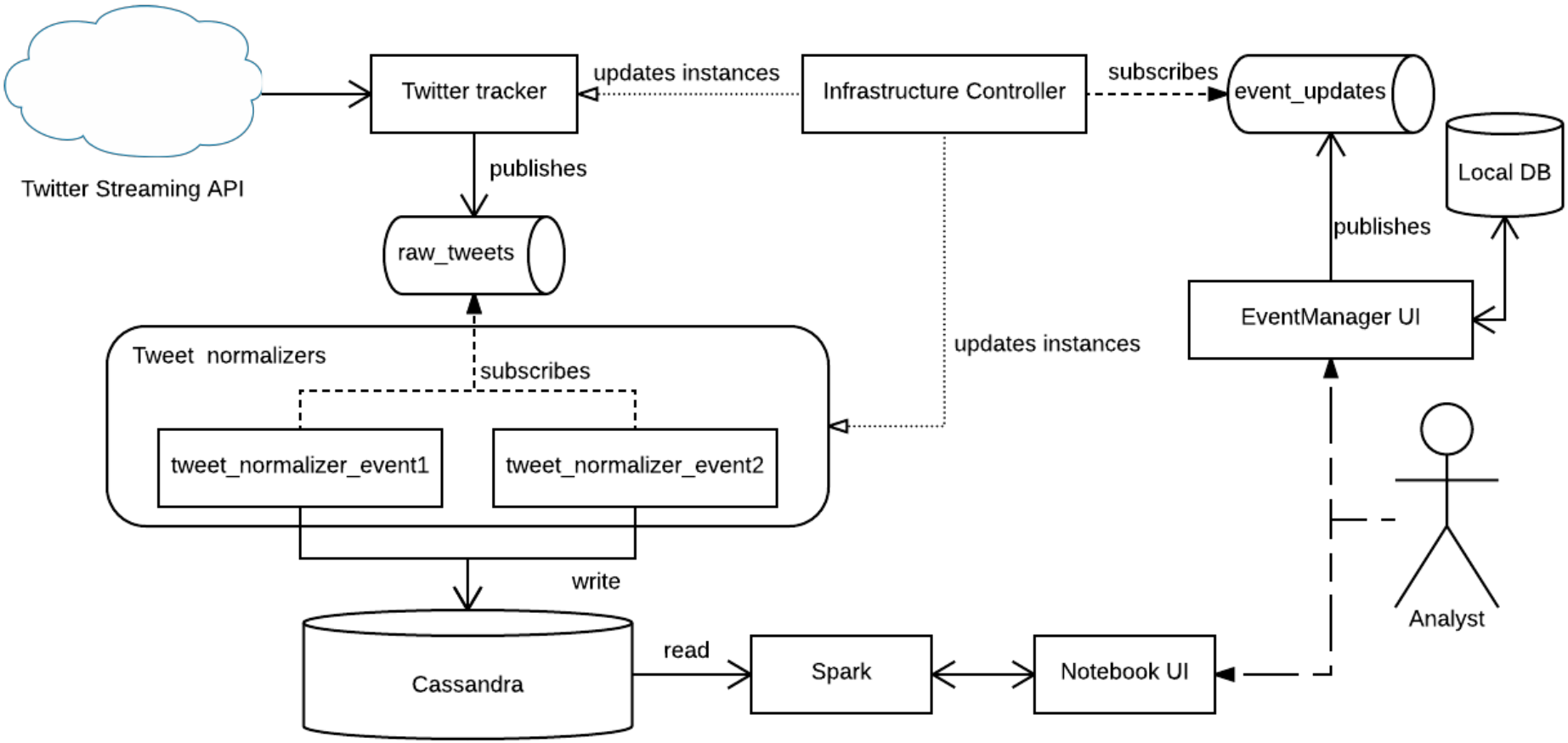
Custom components

- Event Manager
- Infrastructure Controller
- Twitter tracker
- Twitter Normalizer

Cassandra table structure

```
CREATE TABLE twitter_analytics.tweet (
    id uuid, t_id text, event_kw text, event_name text, hashtags list<text>,
    media_url text, t_coordinates text, t_created_at timestamp,
    t_favorite_count int, t_favorited boolean, t_geo text,
    t_is_a_retweet boolean, t_lang text, t_retweet_count int,
    t_retweeted boolean, t_text text, u_created_at timestamp,
    u_description text, u_favourites_count int, u_followers_count int,
    u_friends_count int, u_geo_enabled boolean, u_id text, u_lang text,
    u_listed_count int, u_location text, u_name text, u_screen_name text,
    u_statuses_count int, u_time_zone text, u_url text, u_utc_offset int,
    um_id text, um_name text, um_screen_name text, urls list<text>,
    PRIMARY KEY (id, t_id))
```

Listing 1: Tweets CQL table script


Twitter Streaming API
Twitter tracker
updates instances
Infrastructure Controller
subscribes
event_updates
Local DB
publishes
raw_tweets
publishes
EventManager UI
Tweet normalizers
subscribes
updates instances
tweet_normalizer_event1
tweet_normalizer_event2
write
Analyst
Cassandra
read
Spark
Notebook UI

EPIC EventManager

Refresh architecture | + New event

Smiley

DB code smiley
Tokens :D, :), XD, LOL, WTF, XP, :'(, -.-

Hearth

DB code hearth
Tokens <3, love

nba-draft

DB code nba-draft
Tokens nba, draft, Lonzo, Lavar, basketball, nba draft

NAMESPACE	NAME	READY	STATUS	RESTARTS	AGE
cassandra	cassandra-0	2/2	Running	1	2d
cassandra	cassandra-1	2/2	Running	2	11d
cassandra	cassandra-2	2/2	Running	7	22d
cassandra	spark-master-controller-bkdl7	1/1	Running	0	22d
default	hearth-event-parser-2160998245-m19st	1/1	Running	9	2d
default	k8s-controller-3919038388-75tkk	1/1	Running	0	2d
default	smiley-event-parser-3033807940-c0cwj	1/1	Running	9	2d
default	twitter-tracker-2482383360-s0kz1	1/1	Running	5	2d
frontend	eventmanager-ui-3464180876-h605f	1/1	Running	0	23d
frontend	zeppelin-3633522582-t0kng	1/1	Running	0	2d
kafka	kafka-0	1/1	Running	3	11d
kafka	kafka-1	1/1	Running	0	2d
kafka	zoo-0	1/1	Running	0	2d
kafka	zoo-1	1/1	Running	0	2d
kafka	zoo-2	1/1	Running	0	25d
kube-system	heapster-v1.3.0-4211727876-kv0cl	2/2	Running	0	11d
kube-system	kube-dns-806549836-43lvx	3/3	Running	0	11d
kube-system	kube-dns-autoscaler-2528518105-2wlsr	1/1	Running	1	27d
kube-system	kube-proxy-gke-development-development-dcfa2eb3-2jhj	1/1	Running	0	2d
kube-system	kube-proxy-gke-development-development-dcfa2eb3-l5xb	1/1	Running	1	26d

Zeppelin Notebook Job Search your Notes anonymous

cassandraspark-samples/analytics

Real time twitter Analytics

Press run above to update stats

Columns: id, t_id, event_kw, event_name, hashtags, media_url, t_coordinates, t_created_at, t_favorite_count, t_favorited, t_geo, t_is_a_retweet, t_lang, t_retweet_count, t_retweeted, t_text, u_created_at, u_description, u_favourites_count, u_followers_count, u_friends_count, u_geo_enabled, u_id, u_lang, u_listed_count, u_location, u_name, u_screen_name, u_statuses_count, u_time_zone, u_url, u_utc_offset, um_id, um_name, um_screen_name, urls

```
SQL Context configured
```

Stats on tweets

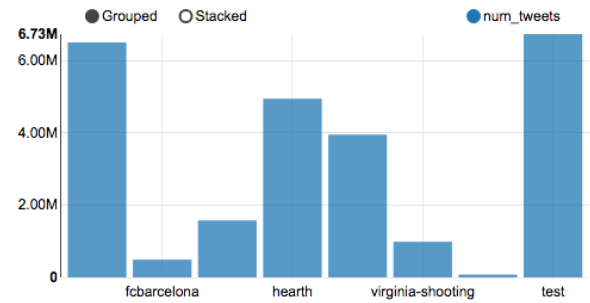
Different stats on all the current dataset available on the database

Total tweet count

```
import org.apache.spark._
import com.datastax.spark.connector._
table: com.datastax.spark.connector.rdd.CassandraTableScanRDD[com.datastax.spark.connector.CassandraRow] = CassandraTableScanRDD[0] at RDD at CassandraRDD.scala:15
res0: IndexedSeq[String] = WrappedArray(id, t_id, event_kw, event_name, hashtags, media_url, t_coordinates, t_created_at, t_favorite_count, t_favorited, t_geo, t_is_a_retweet, t_lang, t_retweet_count, t_retweeted, t_text, u_created_at, u_description, u_favourites_count, u_followers_count, u_friends_count, u_geo_enabled, u_id, u_lang, u_listed_count, u_location, u_name, u_screen_name, u_statuses_count, u_time_zone, u_url, u_utc_offset, um_id, um_name, um_screen_name, urls)
35713760
```

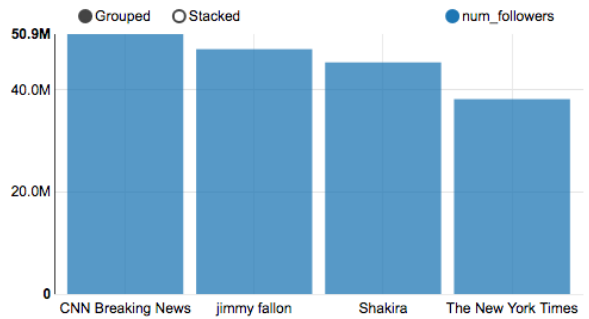
Tweets by event

settings



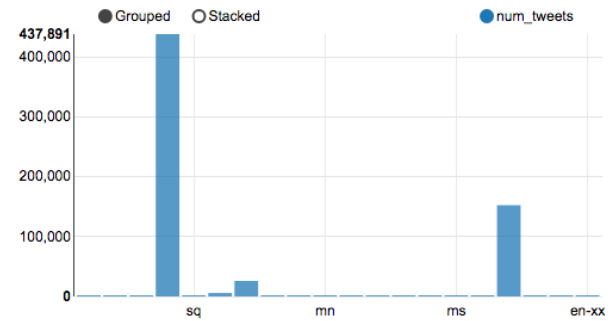
Most influencial accounts (by followers)

settings



Languages used in all tweets

settings



Demo time!

Let’s track an event...

Event Manager UI

...and analyze it!

Zeppelin Notebook

Evaluation & Results

Reliability

Current vs Prototype

- Threads to monitoring
- Every minute script check
- Check log file size

- Kubernetes abstraction
- Auto recovery
- Rolling update
- Node assignation depending on resources usage

Scalability

Current vs Prototype

- Monolyth
- Manual process
- Shared session state
- Load balancing

- Kubernets replica specification
- Stateless microservices
- Independently scalable
- Abstracted infrastructure
- Auto scale

Maintenance

Current infrastructure

- Large & complex
- Strong use of frameworks
- Manual deployment
- Dedicated machines

Prototype

- Easier to mantain: less code, faster development
- Technology flexibility
- Easier to deploy: YAML description files
- Flexibility on cloud provider
- Available built-in tools

Prototype

Twitter tracker 108 lines

Kubernetes controller 145 lines

Event manager 2090 lines

Tweet normalizer 209 lines

Performance

WordCount

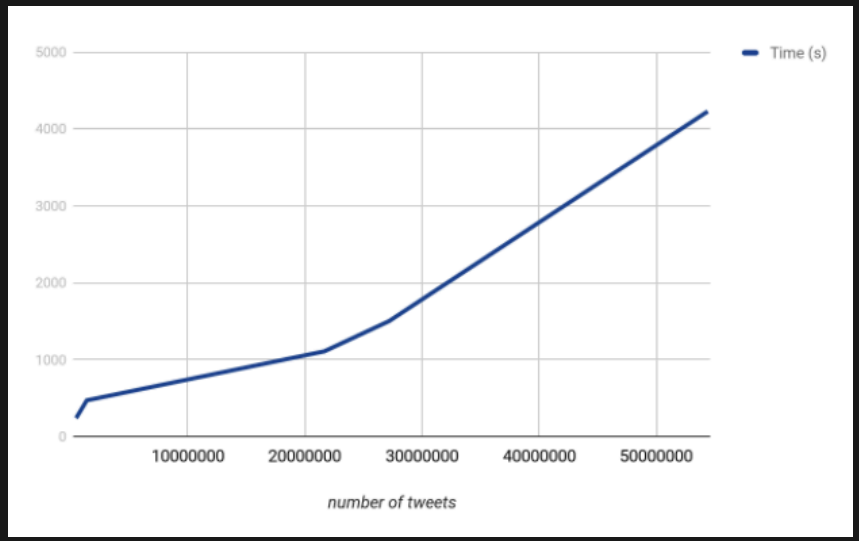
```
val table = sc.cassandraTable("twitter_analytics", "tweet")
val words = table.select("t_text").flatMap(l => l.getString("t_text").split(" ")
    )
     .map(word => (word.toLowerCase,1)).reduceByKey(_ + _).map(_.swap).
          sortByKey(false,1)
```

Listing 2: WordCount Spark script

```
(1) ShuffledRDD[112] at sortByKey at <console>:31 []
+-(176) MapPartitionsRDD[111] at map at <console>:31 []
    |   ShuffledRDD[110] at reduceByKey at <console>:31 []
    +-(176) MapPartitionsRDD[109] at map at <console>:31 []
        |   MapPartitionsRDD[108] at flatMap at <console>:31 []
        |   CassandraTableScanRDD[107] at RDD at CassandraRDD.scala:15 []
```

Listing 3: Debug string rdd

WordCount



Number of tweets	Time (s)	Tweets/sec
490199	238	2060
1400884	469	2987
21680851	1107	19585
27199614	1500	18133
54395957	4228	12866

Future work

- Improve resource specification (CPU, memory)
- Better Cassandra structure
- Unified authentification
- Extend the system with new features

Questions?

Thank you!

Twitter @casassaez

Website gerard.space

Repo github.com/casassg/thesis