Big Data Analytics on Container-Orchestrated Systems

Gerard Casas Saez

University of Colorado Boulder

July 20, 2017

Outline

Why?

Background

Problem statement

Approach

Demo

Evaluation & Results

Related work

Future work

Why?

Keeping up with data growth

Problem

- IOT & Social networks
- Internet traffic
  - Current: 72 petabytes/month
  - Prediction 2021: 232 petabytes/month.

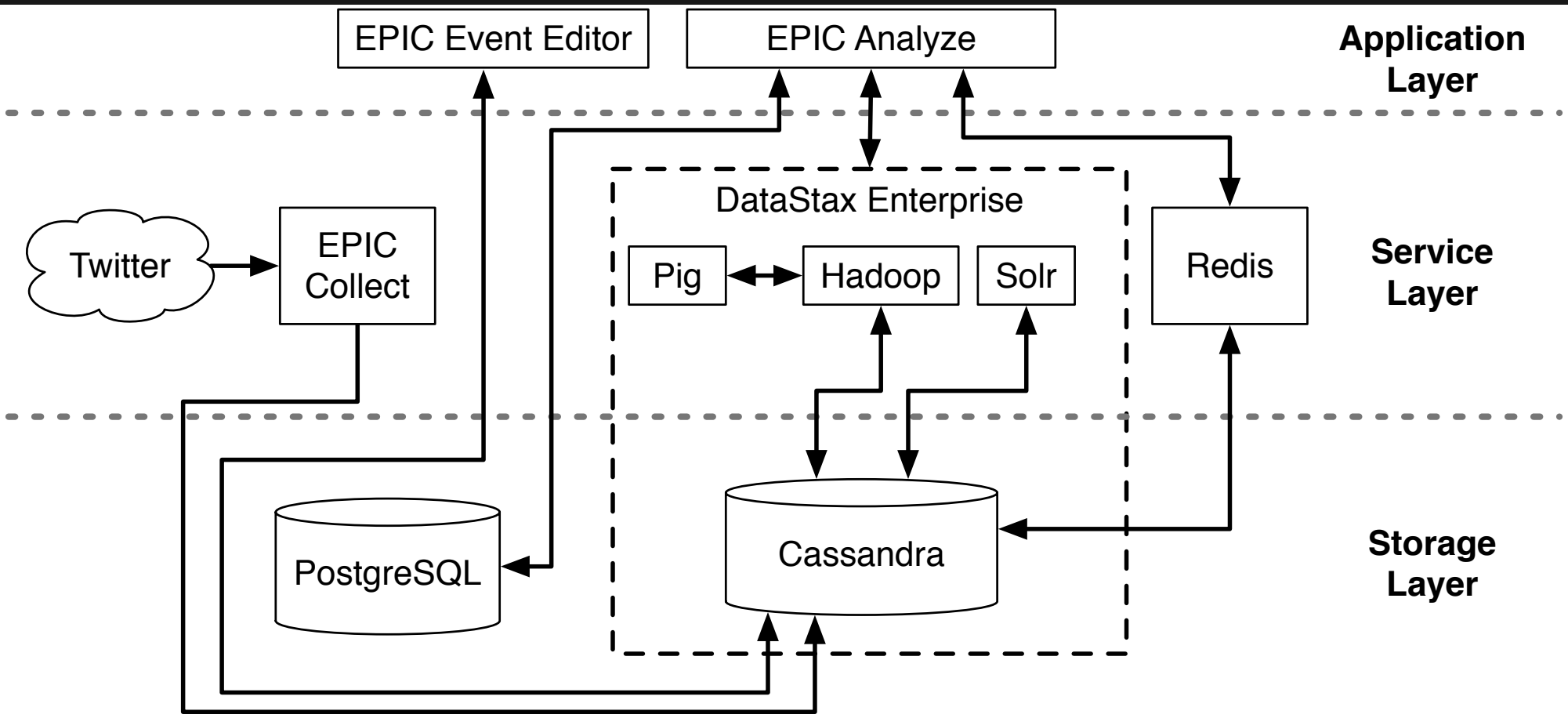
Another problem

- Need to scale Big Data Analytics System
- Keeping mantainance at low cost
- Container-orchestrated make infrastructure easier
- Migrate Project EPIC architecture to container-orchestration system

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

Containerization

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

Container-orchestration systems

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

Microservices Architecture

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

Coreography microservice architecture

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

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?

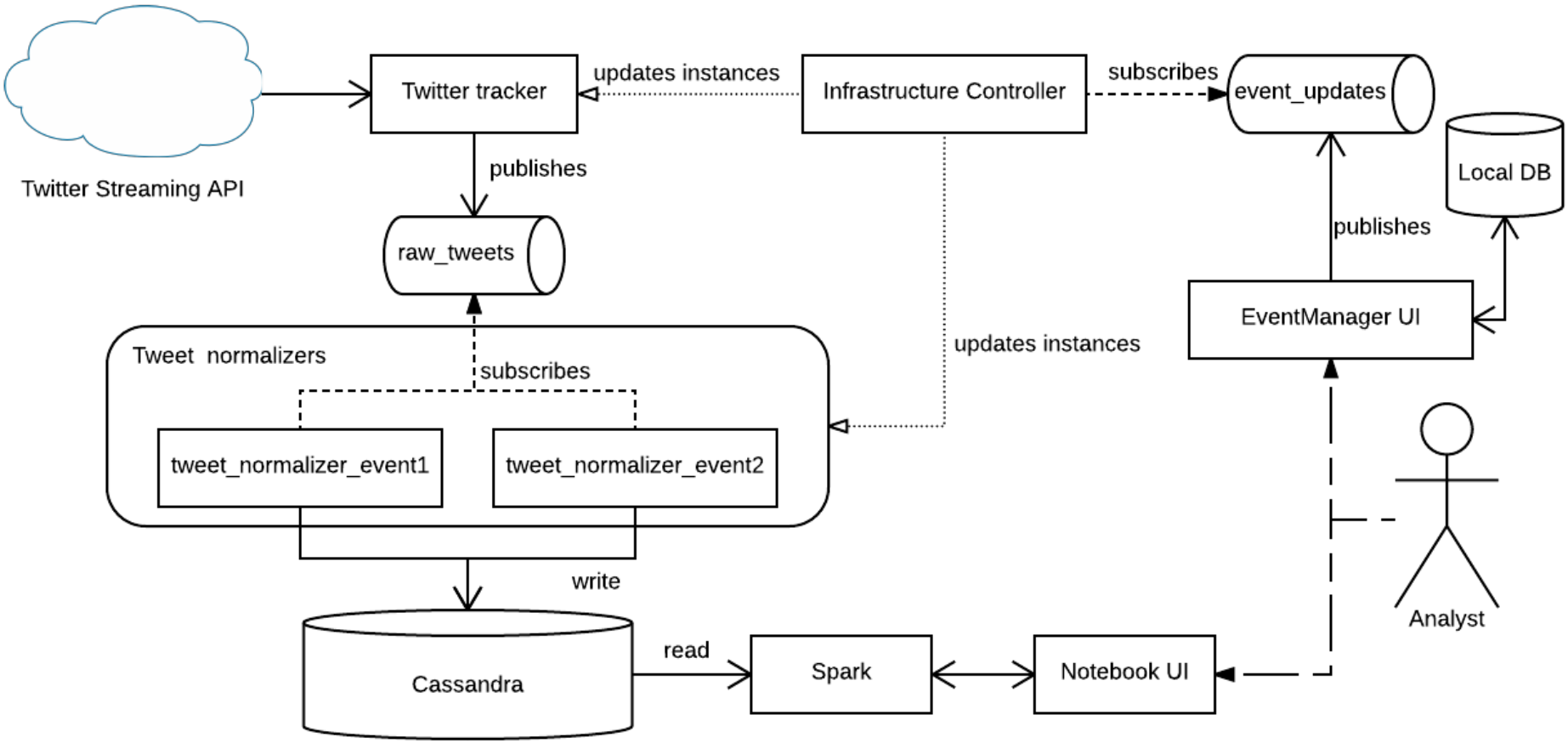
Approach

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


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

Demo time!

Let’s track an event...

Event Manager UI

...and analyze it!

Zeppelin Notebook

Evaluation & Results

Reliability

Current vs Prototype

- EPIC Collect: multi threaded, auto-monitoring
- EPIC Analyze: tight dependency to external resources, manual deployment, no recovery procedure

- Kubernetes abstraction
- Controller manager: Auto-recovery, replica deployment, rolling update
- Scheduler: check memory and cpu and assign intance deployment

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

Scalability

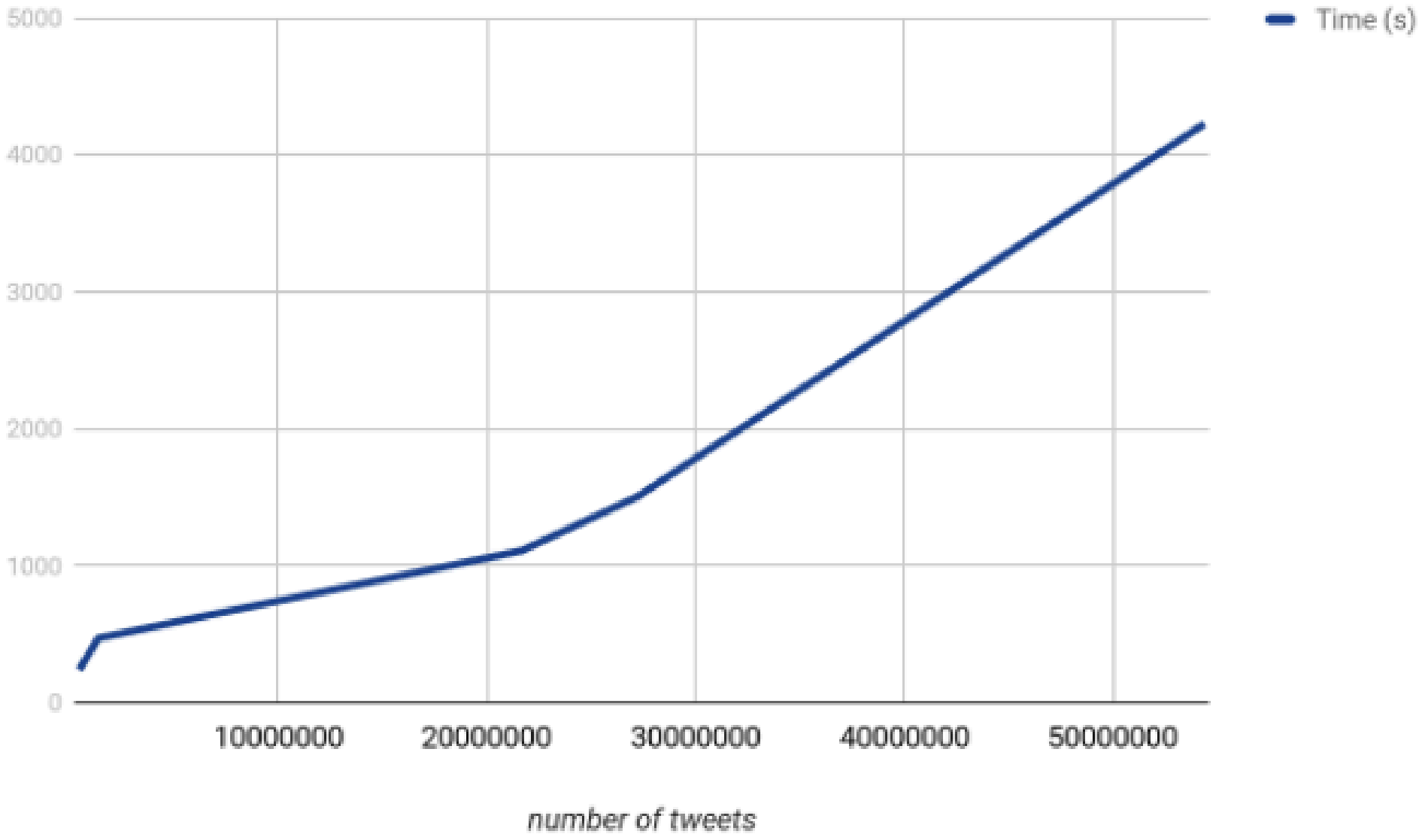
Current vs Prototype

- EPIC Collect: manual, huge load, monolith replication
- EPIC Analyze: monolith, load balancing, shared session state
- Not easy and manual

- Replica specification
- Independently scalable
- Stateless microservices
- Auto scale
- Abstracted management interface

Performance

WordCount execution


Time (s)
5000
4000
3000
2000
1000
0
10000000
20000000
30000000
40000000
50000000
number of tweets

Development and mantenance

Current vs Prototype

- Large and complex
- Strong use of frameworks

- Simple and small code (mostly)
- Independent microservices
- Technology flexibility
- YAML deployment description files

Results

- High scalability with microservices
- Scale components individually
- Abstraction over system management vs manual
- Monitoring built-in

Results

- Easier to mantain: less code, faster development
- Ease of configuration and deployment
- Flexibility on cloud provider deployment
- Resource usage optimizer
- Other features: Rolling-update, auto-scale…

Related work

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]

Future work

Future work

- Improve resource specification (CPU, memory)
- Unified authentification
- Better Cassandra structure
- Extend the system to fit current

Questions?

Thank you!

Twitter @casassaez

Website gerard.space

Repo github.com/casassg/thesis