# Advanced service-based data analytics: Models, Elasticity and APIs

Hong-Linh Truong
Distributed Systems Group, TU Wien

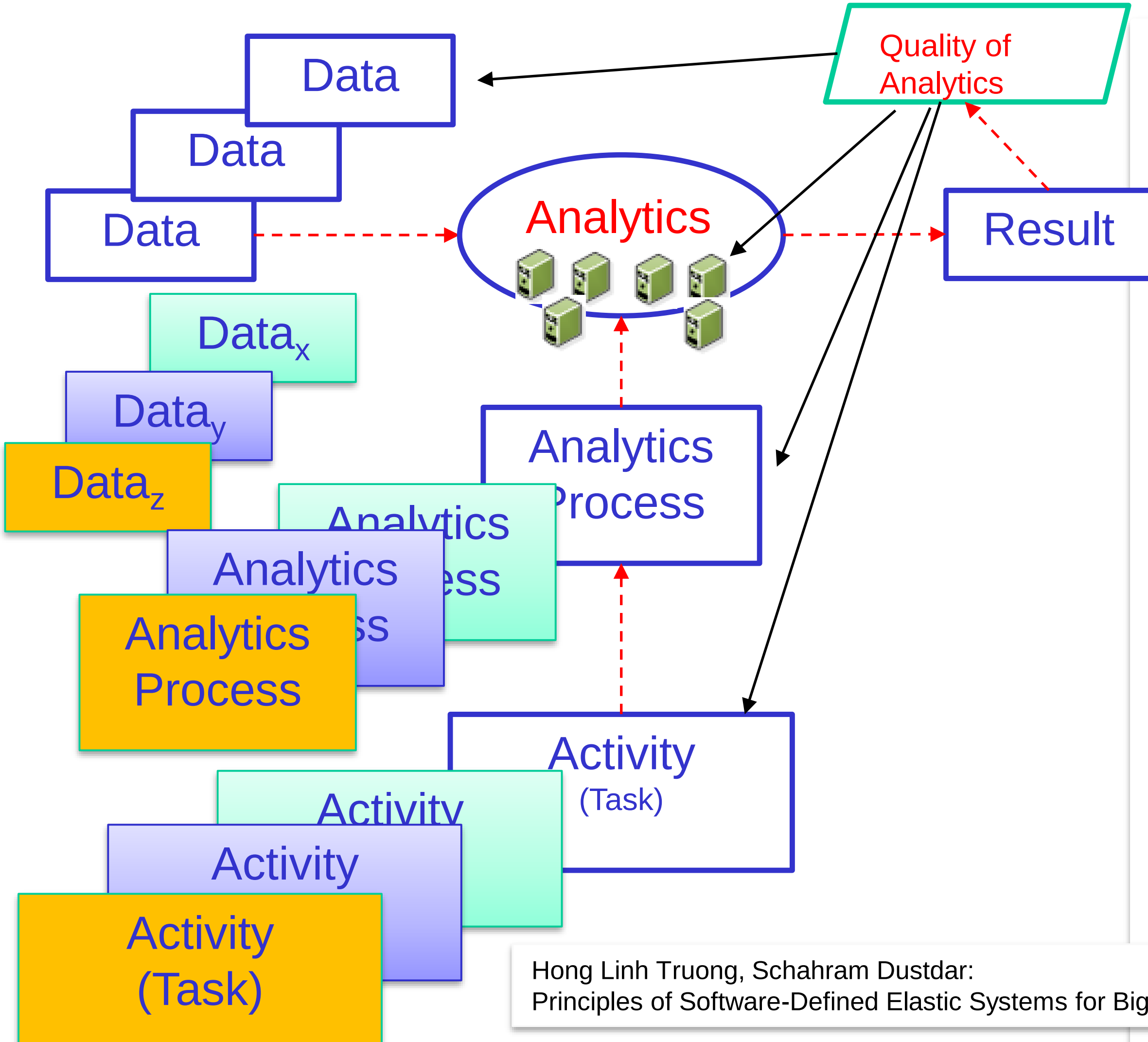
truong@dsg.tuwien.ac.at
dsg.tuwien.ac.at/staff/truong
@linhsolar

# Outline

- Principles of elasticity for advanced service-based data analytics
- Data analytics within a single system
- Data analytics across multiple systems
- APIs management

# PRINCIPLES OF ELASTICITY FOR DATA ANALYTICS

# Elasticity in (big) data analytics

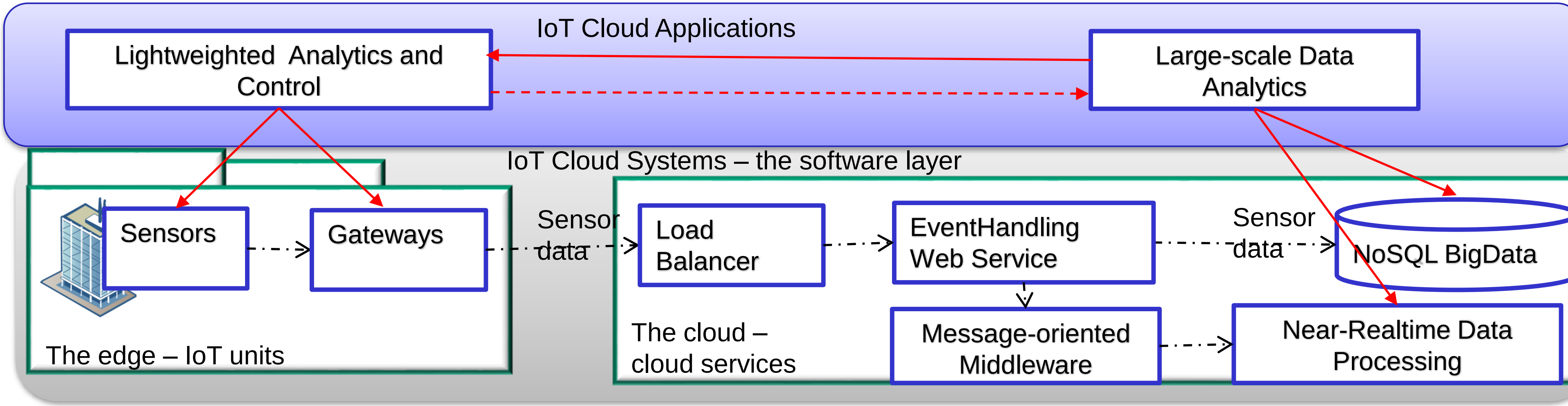



- **More data** → more compute resources (e.g. more VMs)
- **More types of data** → more activities → more analytics processes
- Change **quality of analytics**
  - Change quality of data
  - Change response time
  - Change cost
  - Change types of result (form of the data output, e.g. tree, table, story)

Hong Linh Truong, Schahram Dustdar:
Principles of Software-Defined Elastic Systems for Big Data Analytics. IC2E 2014: 562-567

# Elasticity in slices of IoT, Network functions and cloud resources

## Application example

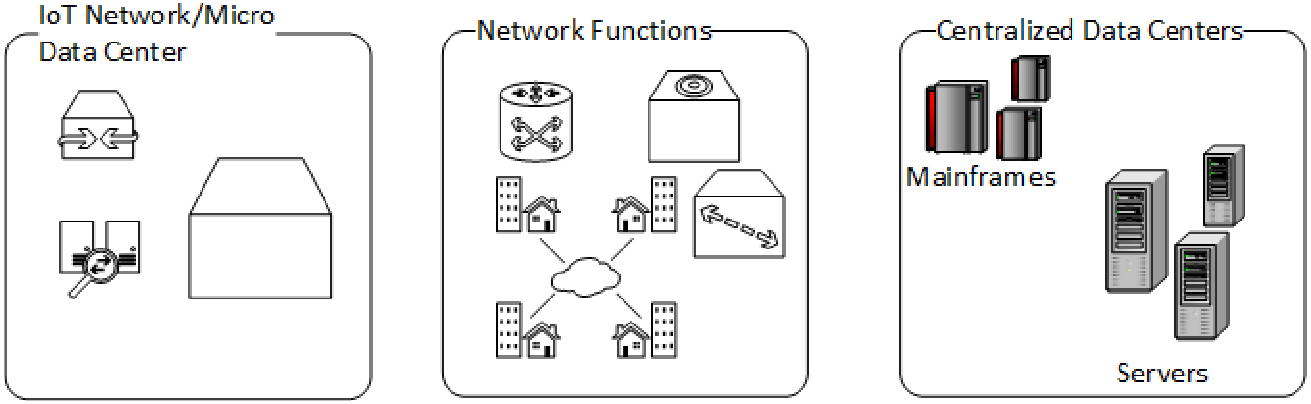



What should we do if suddenly many sensors send a lot of data?

What if you know that “5 minutes from now, 10*n sensors will be started?

# Elasticity in slices of IoT, Network functions and cloud resources

**„IoT + Network functions + Clouds“**




What if in the “network functions” we can create VMs or perform network traffic engineering?

Elasticity principles can be used to support dynamic quality of analytics

# Elasticity Principles: Elasticity of data and analysis processes

- Multiple types of objects from different sources with complex dependencies, relevancies, and quality
- Different data and algorithms models for analyzing the same subject
- New analytics subjects can be defined and analytics goals can be changed
- Decide/select/define/compose not only data but also analysis pipelines based on existing ones

Management and modeling of elasticity of data and processes during the analytics

# Elasticity Principles: Elasticity of data resources

- Data provided, managed and shared by different providers
- Data associated with different concerns (cost, quality of data, privacy, contract, etc.
- Static data, open data, data-as-a-service, opportunistic data (from sensors and human sensing)
- Distributed big data and multiple data owners

Data resources can be taken into account in an elastic manner: similar to VMs, based on their quality, relevancy, pricing, etc.

# Elasticity Principles: Elasticity of humans and software as computing units

- Human in the loop to solve analytics tasks that software cannot do
- Human-based compute units can be scaled up/down with different cost, availability, and performance models
- Human-based compute units + software-based compute units for executing analysis pipelines
- Elasticity controls can be also done by humans

Provisioning hybrid compute units in an elastic way for computing/data/network tasks as well as for monitoring/control tasks in the analytics process

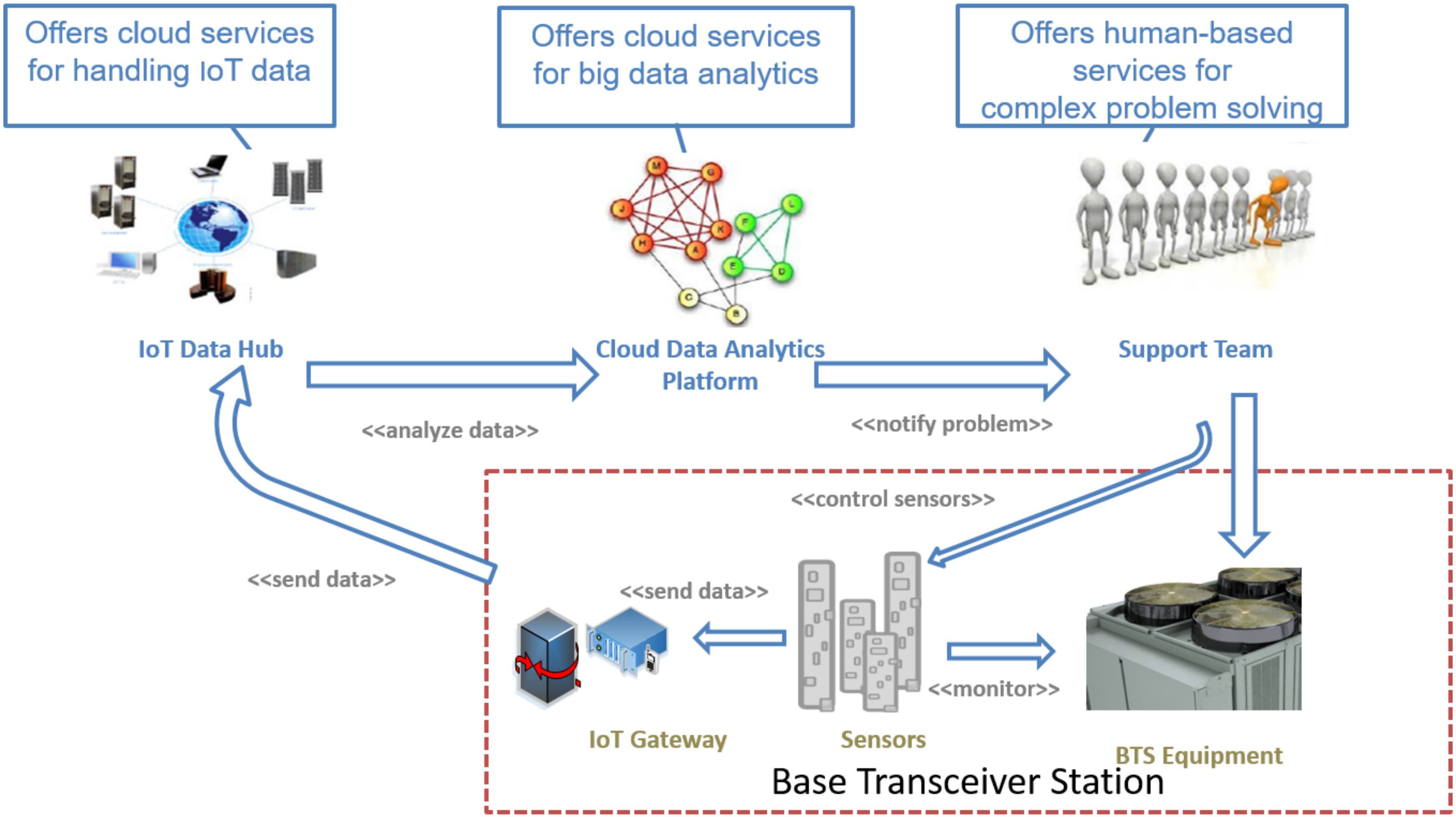
# Elasticity Principles: Elasticity of quality of analytics

- Definition of quality of analytics
  - Trade-offs of time, cost, quality of data, forms of output
- Using quality of analytics to select suitable analysis processs, data resources, computing units
- Multi-level control for the elasticity based on quality of analytics

Able to cope with changes in quality of data, performance, cost and types of results at runtime

# Advanced service-based analytics – which are fundamental engineering questions?

# Recall -- Predictive Maintenance in Telcos



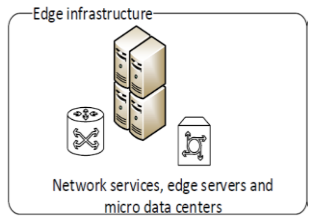
# Advanced service-based data analytics -- fundamental concepts

Domain 1 | Domain 2 | Domain n

Applications: Part A → Part B → ... → Part N

System infrastructures


IoT




Edge servers

Local Cloud

amazon webservices

Windows Azure Platform AppFabric

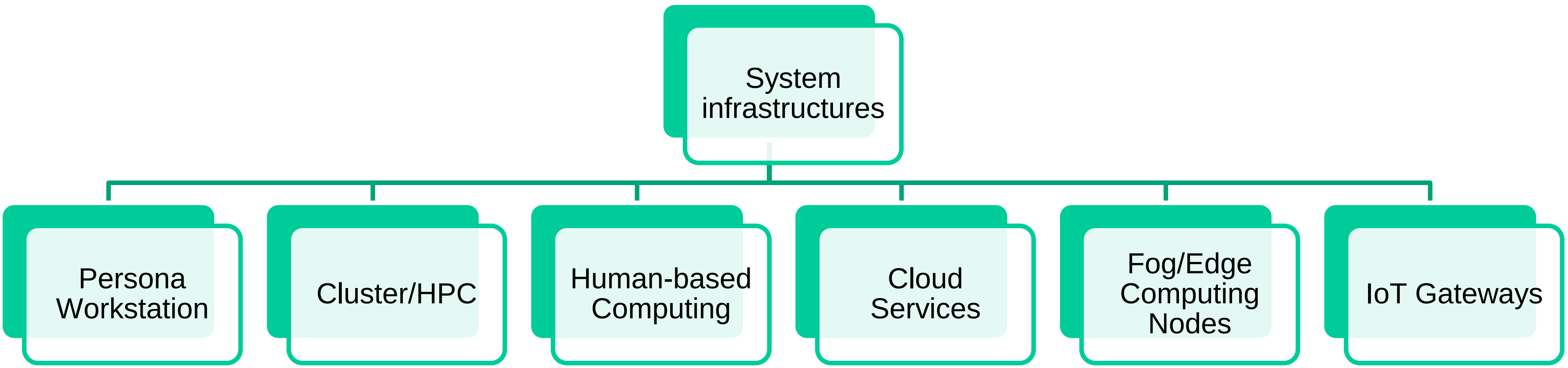
Public cloud

# Design questions

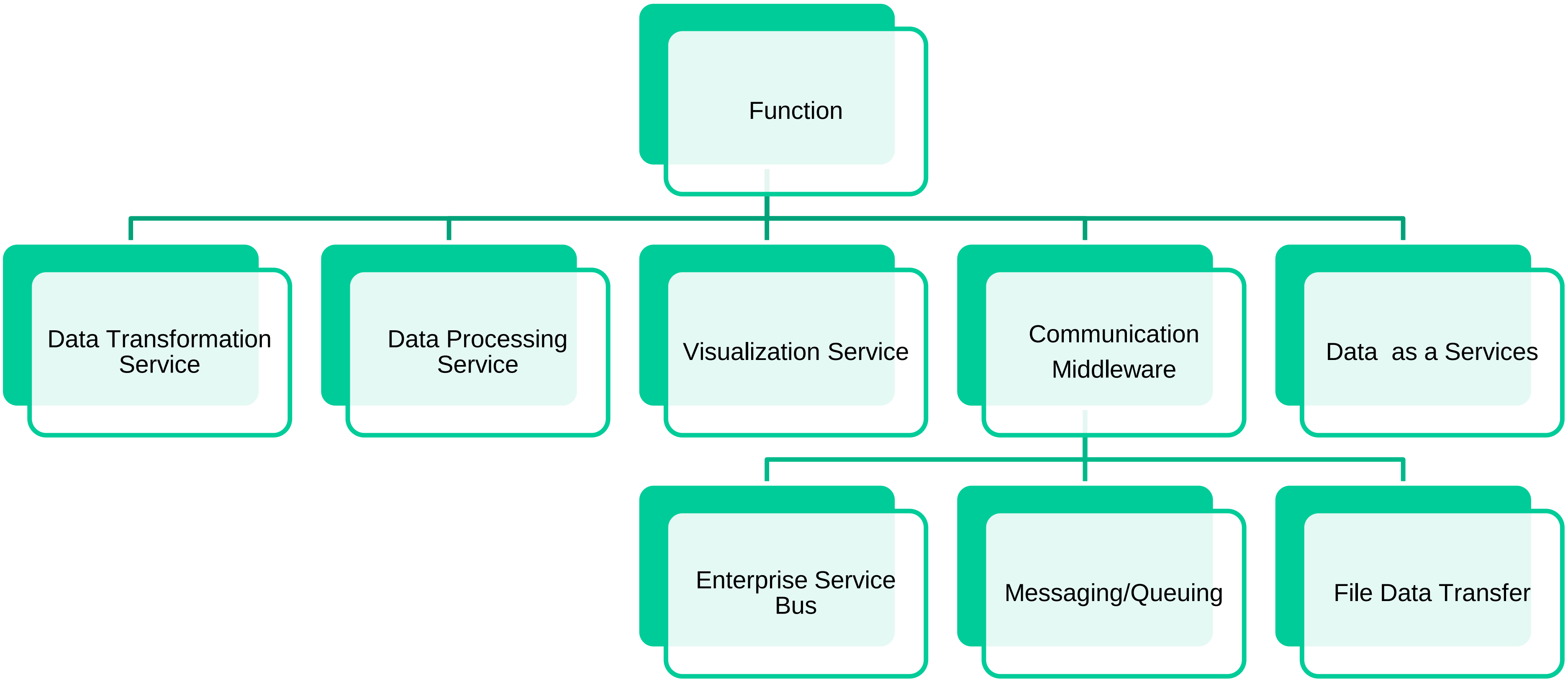
Part = a (composite) services/components

- Which system infrastructures are used?
- Which interfaces/APIs  are suitable for services?
- Which programming models are used within services?
- Which non-functional parameters are important and how to measure them?

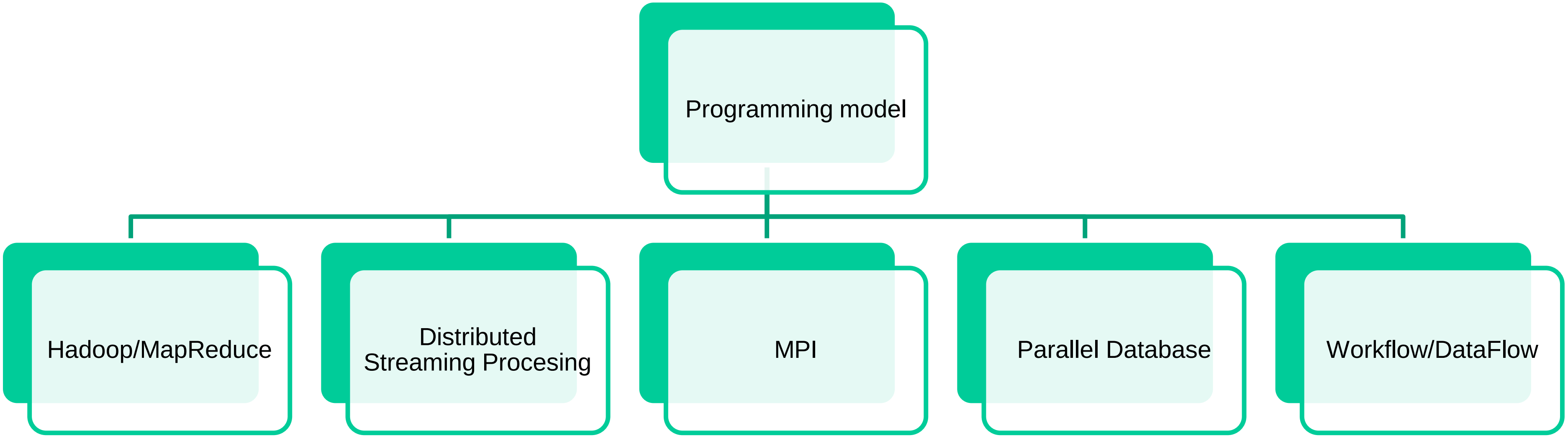
# Fundamental concepts – system infrastructure unit



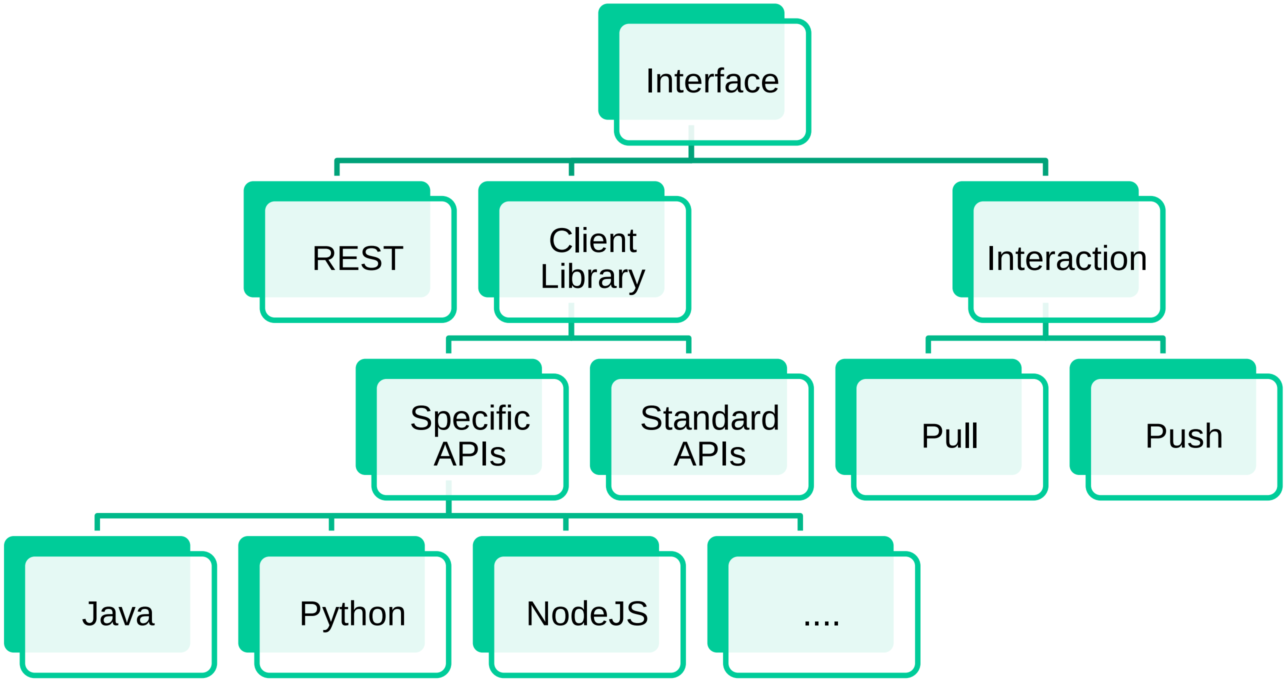
# Fundamental concepts – unit functions



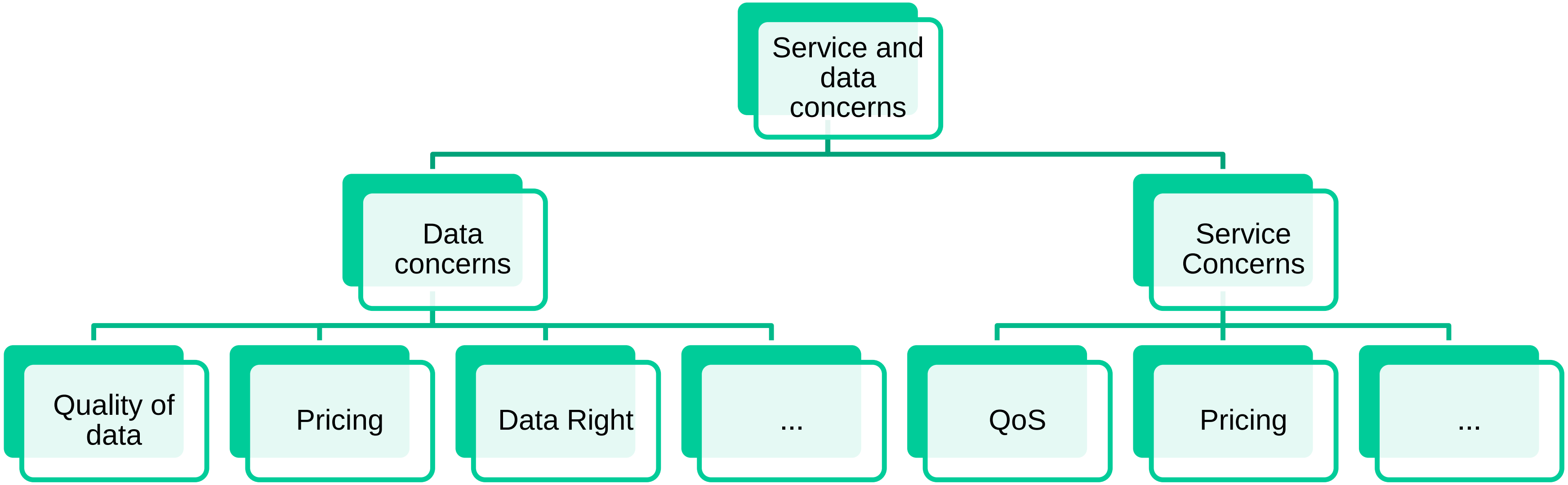
# Fundamental concepts – programming model within units



# Fundamental concepts – interfaces between services



# Fundamental concepts – services and data concerns



# You see we need to deal with many techniques and frameworks

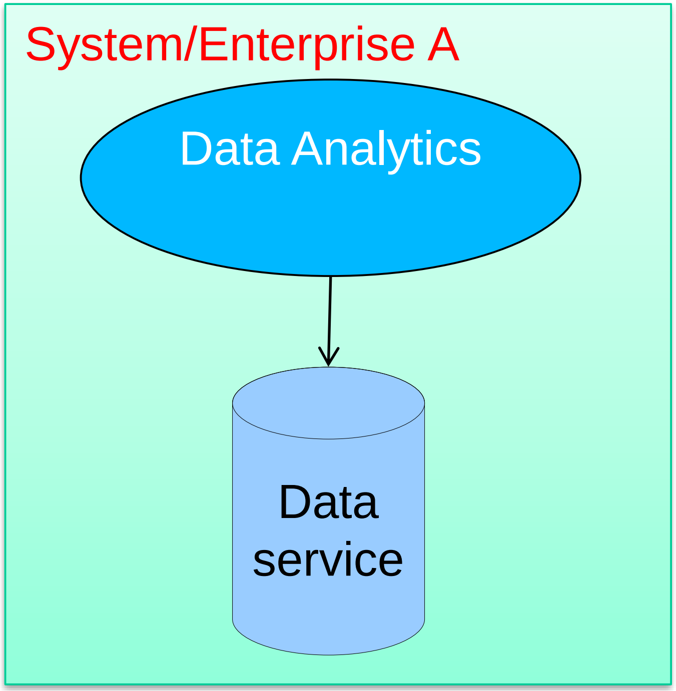
# WE NEED TO START FROM DATA ANALYTICS WITHIN A SINGLE SYSTEM

**What is our understanding about a single system?**

**Location and enterprise boundary?**

**Within a virtual infrastructure owned by a single organization?**

# Data analytics within a single (technical) system



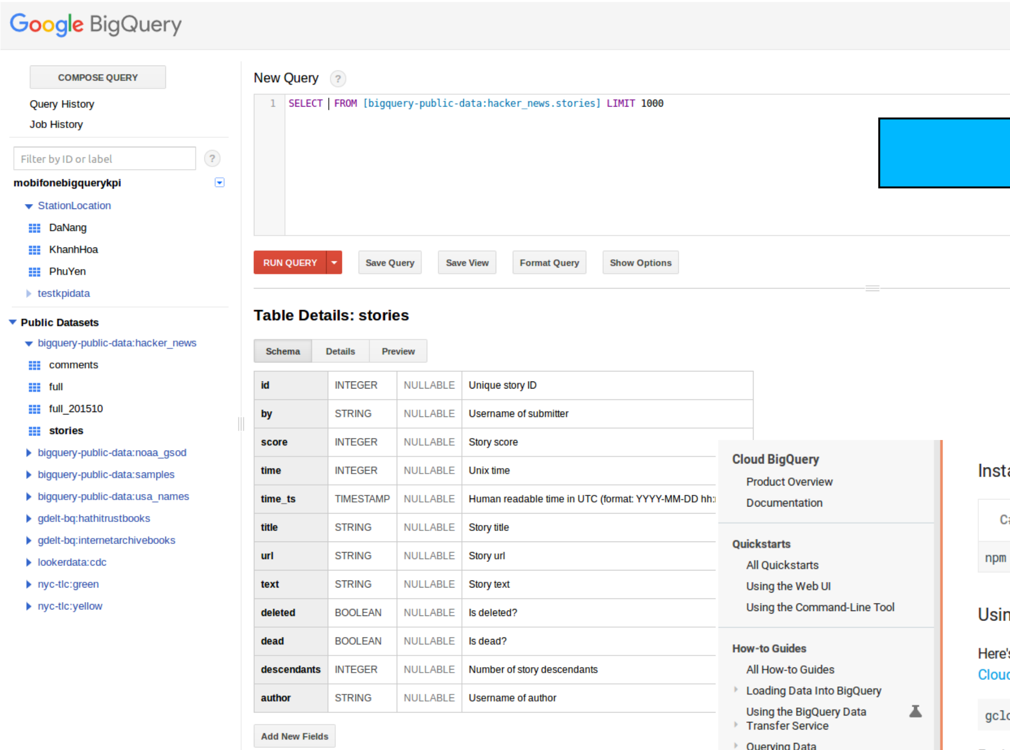

- In a single domain
  - Tightly coupled computing infrastructures
    - E.g., in the same cloud
  - Computation and data are close
  - Several concerns can be by-passed
  - They can be complex

# Data analytics within a single system – some examples

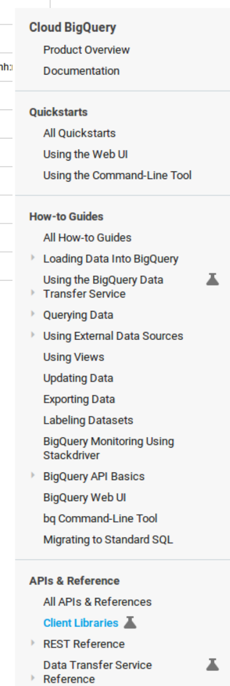
- Message Passing Interface (MPI) + Cluster-based File system
- Parallel Database (SQL/NonSQL)
- MapReduce + Google File System
- Yahoo S4
- Hadoop + HDFS
- Spark
- Dryad+LINQ
- Scientific/Business Workflow

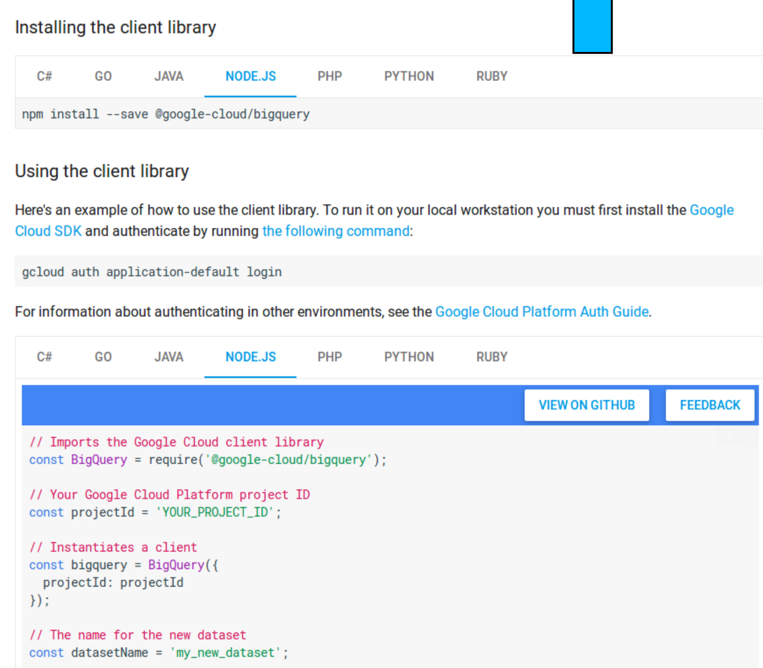
A short, good overview in Chapter 6: Cloud Programming and Software Environments, Book: Distributed and Cloud Computing – from Parallel Processing to the Internet of Things, Kai Hwang, Geoffrey C. Fox and Jack J Dongarra, Morgan Kaufmann, 2012

# Example - BigQuery (1)




Google BigQuery




From https://cloud.google.com/bigquery/docs/reference/libraries

# Example – BigQuery: complexity

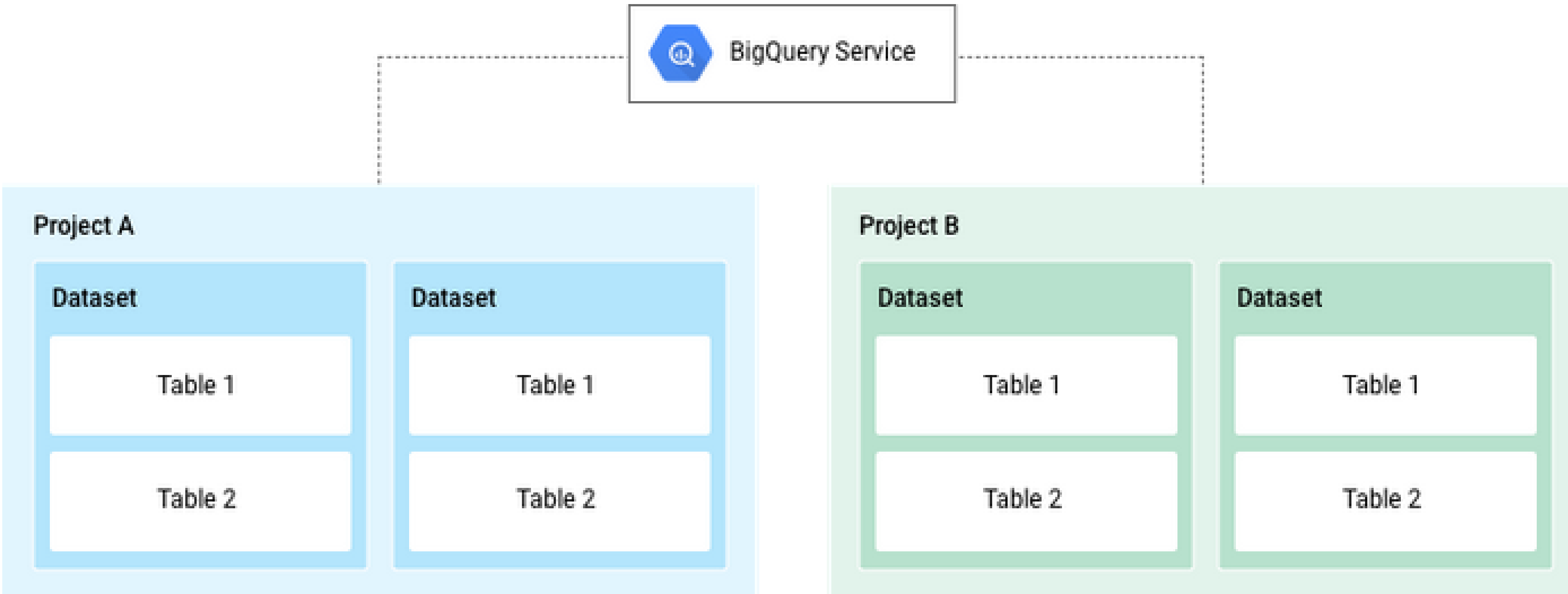
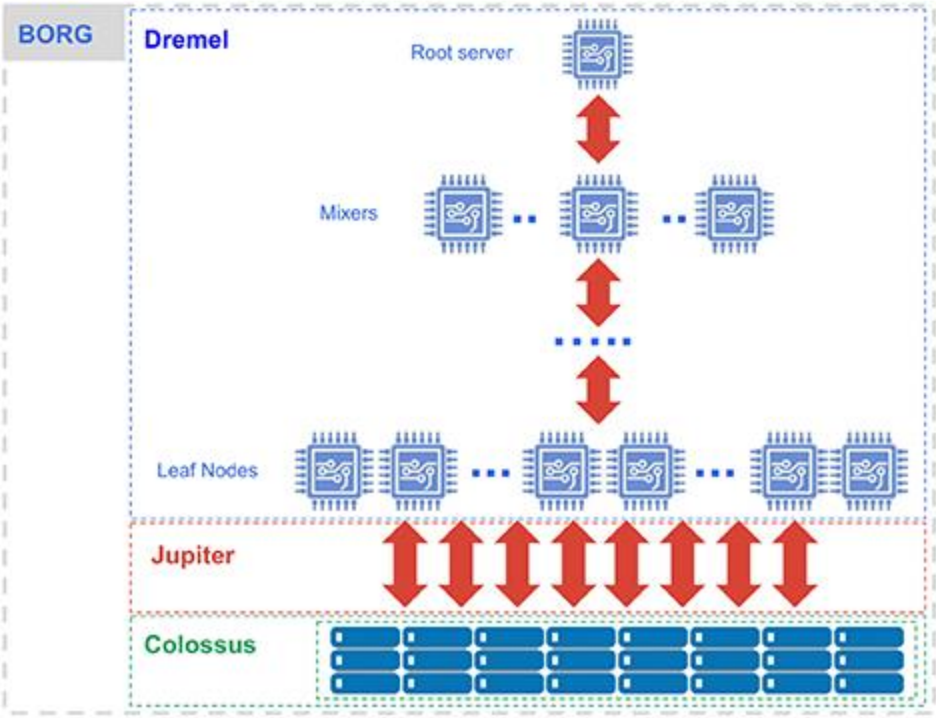



**Figure 1**: BigQuery structural overview

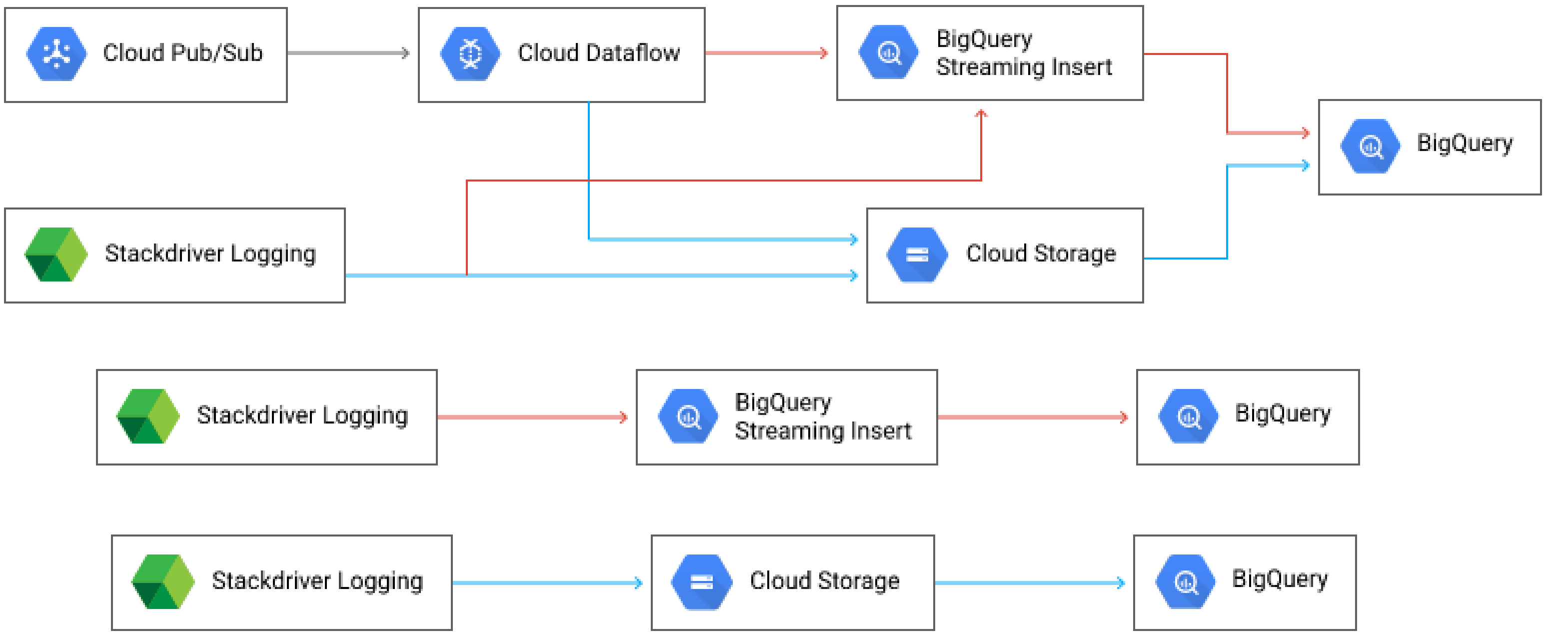
Source https://cloud.google.com/solutions/bigquery-data-warehouse

# Example – BigQuery: complexity




Source: https://cloud.google.com/blog/big-data/2016/01/bigquery-under-the-hood

# Example – BigQuery: complexity




Source: https://cloud.google.com/solutions/architecture/optimized-large-scale-analytics-ingestion

But why it might not be suitable for you? When?

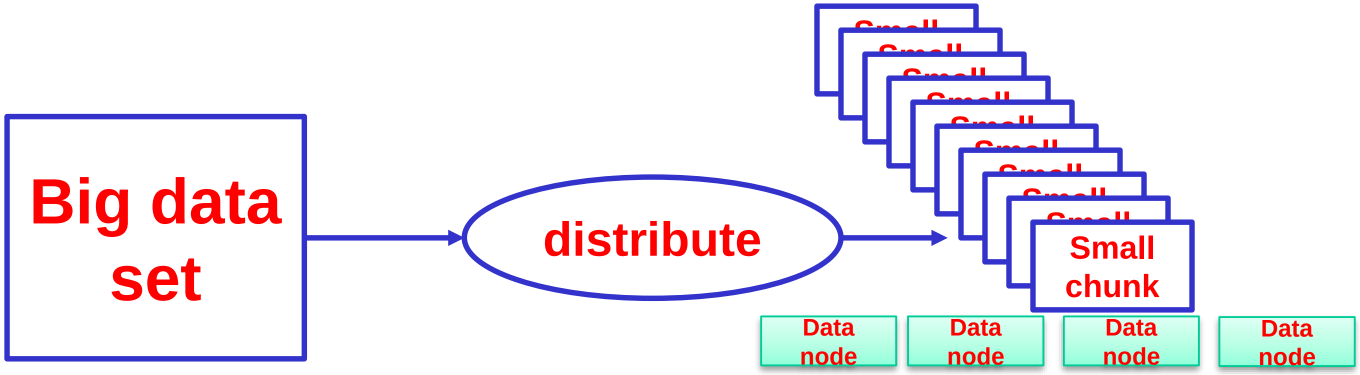
# Example - Hadoop

```
truong@bachphu-spark-m: ~

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
truong@bachphu-spark-m:~$ ls
aa  linh.csv  spark-warehouse  tt.py
truong@bachphu-spark-m:~$ hadoop fs -ls /user/truong
17/05/18 21:03:20 INFO gcs.GoogleHadoopFileSystemBase: GHFS version: 1.6.0-hadoop2
Found 4 items
drwxr-xr-x   - truong hadoop          0 2017-05-17 14:29 /user/truong/.sparkStaging
-rw-r--r--   2 truong hadoop       8945 2017-05-12 07:42 /user/truong/aa
drwxr-xr-x   - truong hadoop          0 2017-05-12 07:40 /user/truong/output
-rw-r--r--   2 truong hadoop       8945 2017-05-12 07:41 /user/truong/part-r-00000-9f88111c-f139-40e5-ac06-53a6e283cd40.csv
truong@bachphu-spark-m:~$ hadoop fs -copyFromLocal aa /user/truong/test.csv
17/05/18 21:04:00 INFO gcs.GoogleHadoopFileSystemBase: GHFS version: 1.6.0-hadoop2
truong@bachphu-spark-m:~$
```

Hadoop File Systems

# Example – Hadoop: complexity

- Distributing data into multiple nodes/machines is the key! Why?
- Hadoop provides a parallel file system – Hadoop File Systems
  - Deal with hardware failures, support data locality,  streaming data access
  - Like traditional file systems with new features for big data
- Key principles:

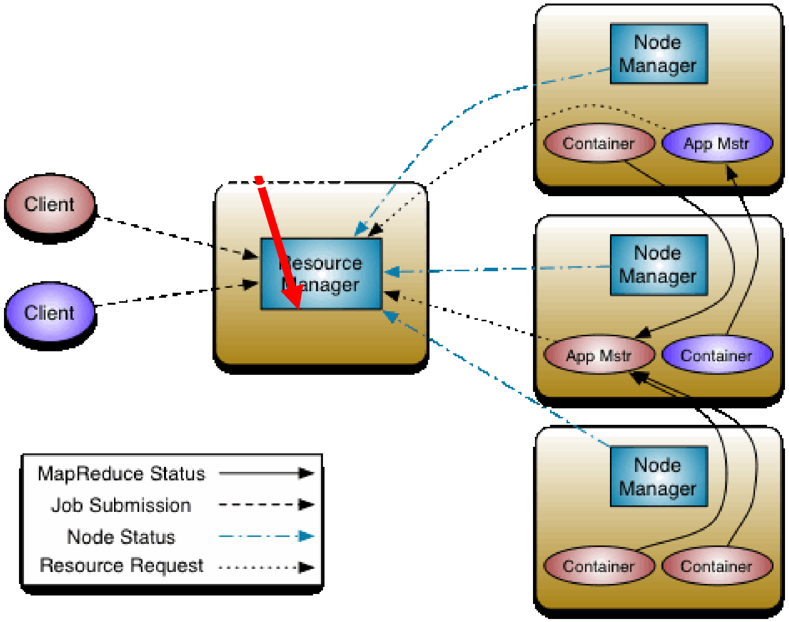




Distributed Systems Group

# Example – Hadoop: complexity

- Several computers are used to setup Resource Manager and Node Manager
- You write the tasks and you submit the tasks




Source: http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html

# Example – Hadoop: simple

```
spark =
SparkSession.builder.appName("sp_AlarmTypePerday").getOrCreate()
df
=spark.read.csv(“hdfs://test/Alarm_nodeB_DN_9_Jan.csv",header=True,
inferSchema=True)
newdf = df.select(['Alarm Number','Started','Canceled'])
newdf.show()
```

Submission command line (your local machine)

gcloud dataproc jobs submit pyspark --cluster cluster-spark Test.py

spark-submit --master spark://master-node:7077 Test.py

Google cloud

cluster-spark
(Google spark installation with n nodes)

master-node

Slave node | Slave node | Slave node | Slave node | Slave node

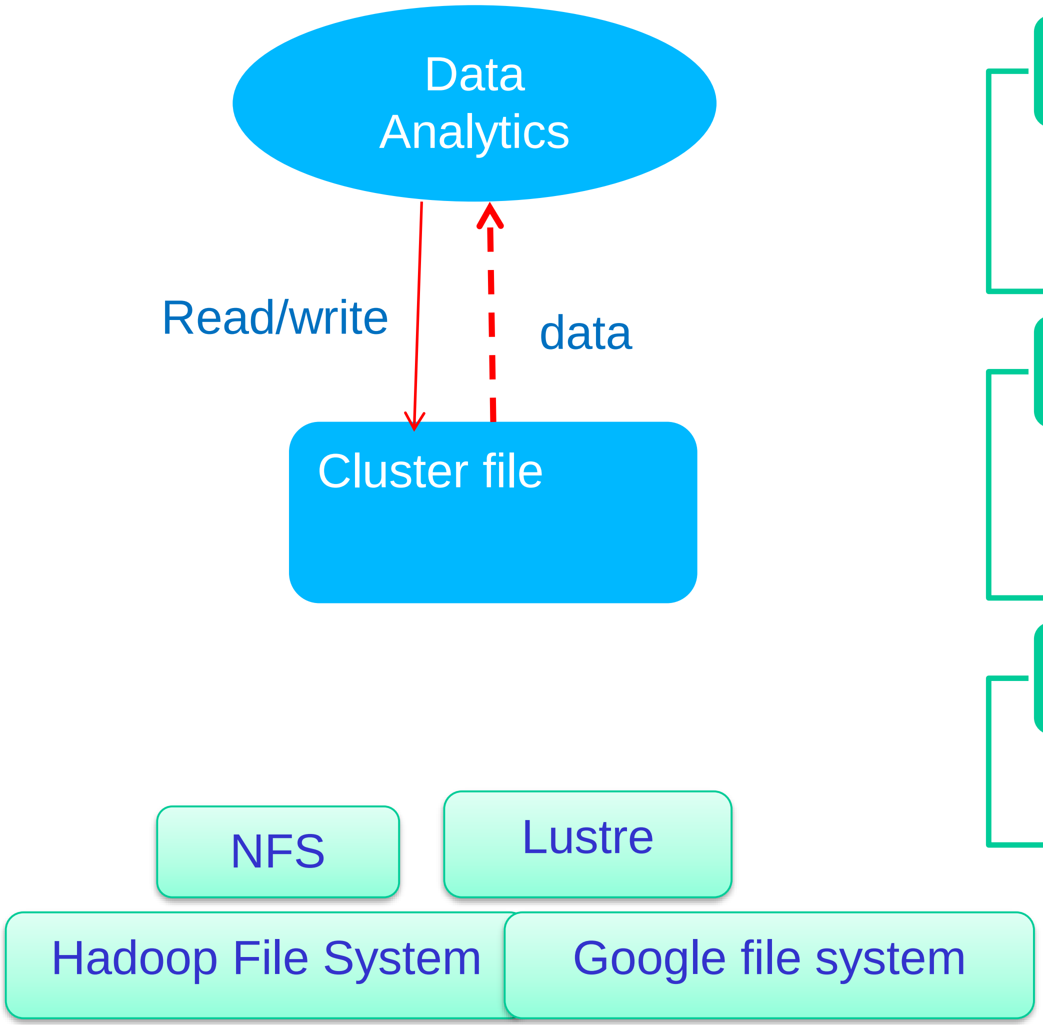
**But why it might not be suitable for you? When?**

# Similar questions

- With ElasticSearch, MongoDB, Canssandra, etc. within a single system → they can be very large and scalable!

- But when are they not enough? When are they not suitable for us?

# Data analytics across multiple systems – data service units




**Interface**

- Read/write data via direct , low-level read/write via IO

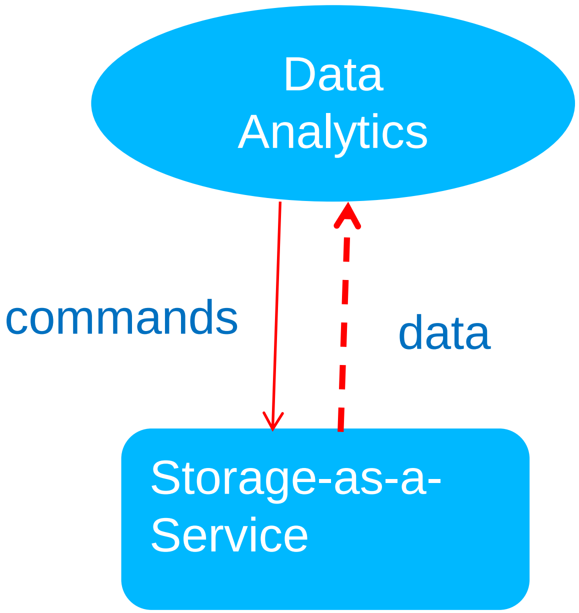
**System**

- Cluster or cluster of clusters
- Can be very large

**Programming model**

- Usually parallel processing

# Data analytics across multiple systems – data service units




**Interface**
- Direct data transfer via REST/SOAP APIs

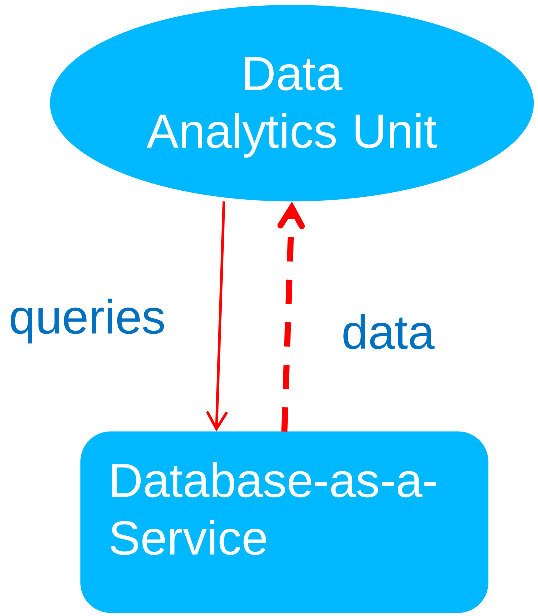
**System**
- Decouple between analytics and storage

**Programming model**
- May require middleware for data transfer
  - Request via SOAP/REST
  - Real data transfer done by external middleware
- A rich set of programming models can be used

Amazon S3 (SOAP/REST API)

Google Storage Service (REST API)

# Data analytics across multiple systems – data service units




**Interface**

- REST/SOAP APIs
- Mainly for commands and results

**System**

- Decouple between analytics unit and database
- Database as a sevice can be very large

**Programming model**

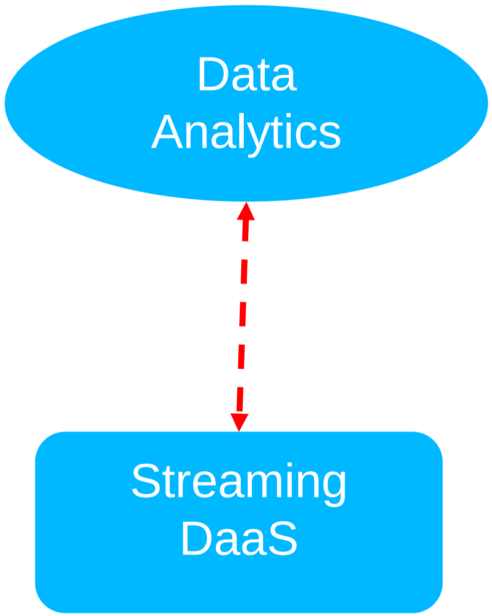
- Analytics can be done at both sides
- Analytic units can use any programming models
- Database-as-a-service can perform  a lot of analytics
  - Parallel database operations

## Technology

MongoDB/MongoLab
Amazon DynamoDB
Amazon SimpleDB
Cloudant Data

SkySQL
Amazon RDS
Microsoft SQL Azure
Clustrix DBaaS

# Data analytics across multiple systems – data service units




Technology

StormMQ, RabbitMQ, CloudMQTT, Google Data Hub, Azure Data Hub, ...

## Interface

- Data transfer can be uni or bi-direction
- Streaming data protocols

## System

- Both systems for DaaS and for analytics units can be very large

## Programming model

- Can be any

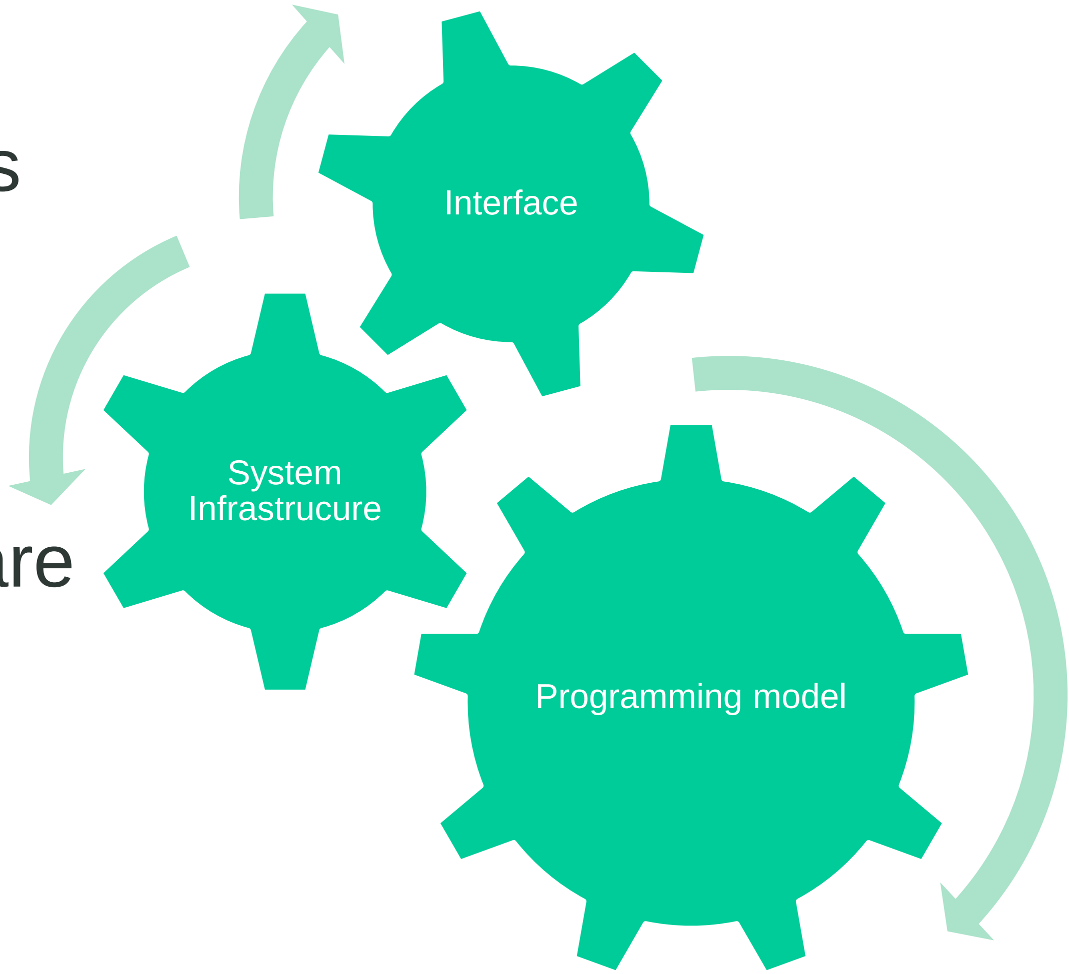
# WHY SHOULD ANALYTICS UNITS BE „CLOSED“ TO DATA UNITS?

# WHICH CONCERNS COULD BE IGNORED IN SINGLE SYSTEM DATA ANALYTICS?

# WHICH ARE THE ISSUES THAT WE NEED TO CONSIDER WHEN OUR DATA UNITS ARE IN DIFFERENT SYSTEMS?

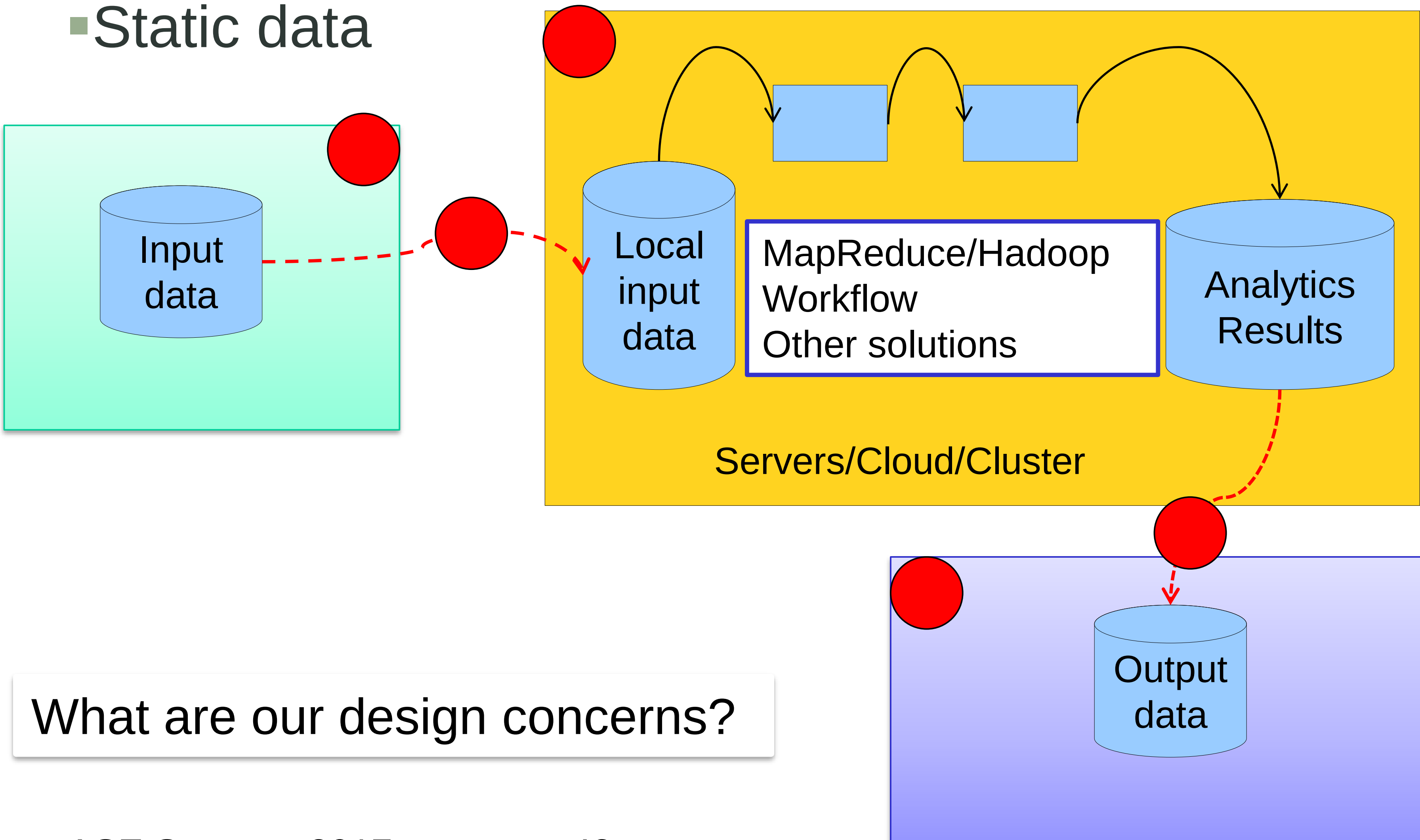
# Data analytics across multiple systems – design choice

- Programming models for data analytics service
- Data service units
- Supporting middleware units



# Data analytics across multiple systems – programming models (1)

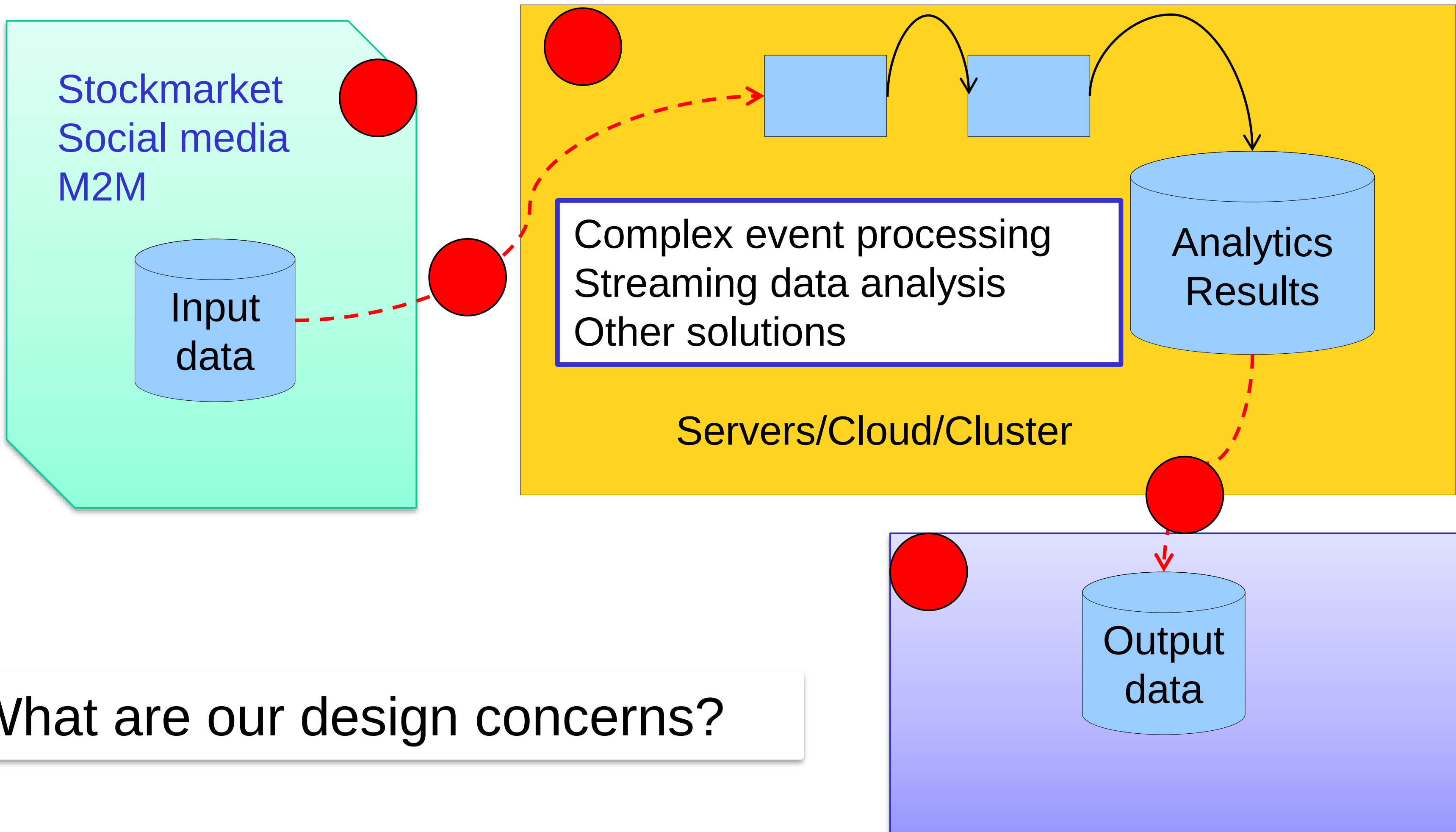
- Static data




What are our design concerns?

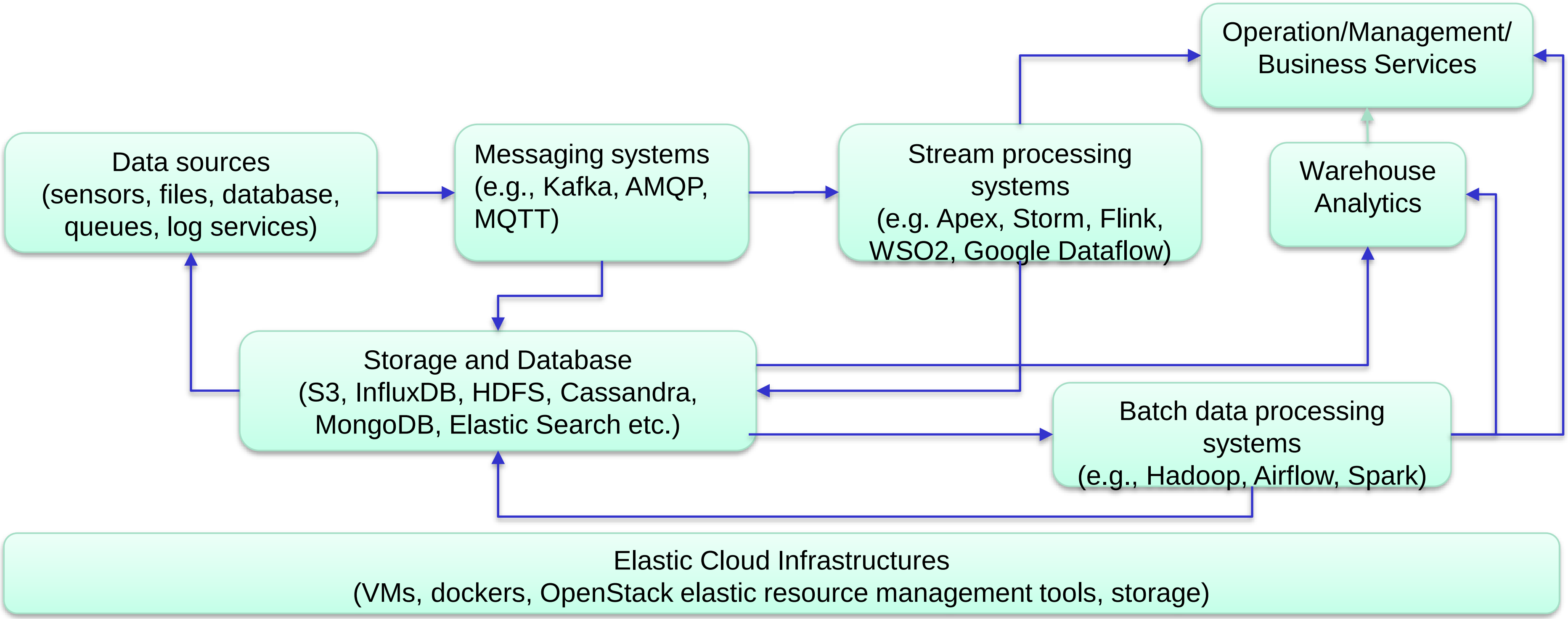
# Data analytics across multiple systems – programming models (2)

- Near-realtime data




What are our design concerns?

# Cloud services and big data analytics




Very complex problems due to software complexity, infrastructures management and service providers

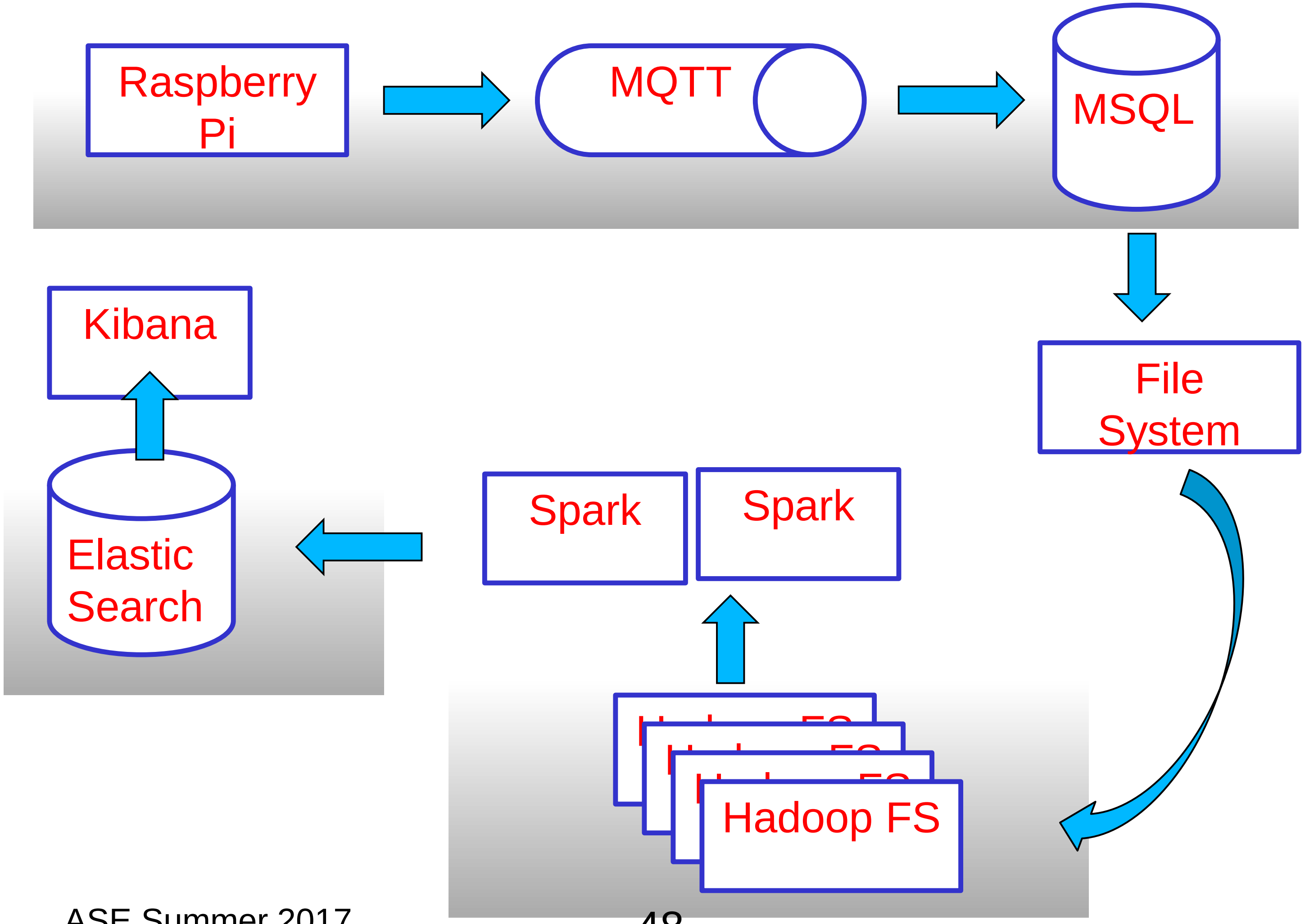
# Case studies

- Monitoring equipment and environments
  - Electricity, temperature, air conditioner breakdown, etc.
- Using MQTT and MySQL


- Requirements:
  - Now would like to do big data analytics (for certain type of problems) – offline per day
  - Do not want to manage the big data analytics system
  - Not worry about data privacy/regulation

What would you recommend for solving the requirements?

# Example – lgacy then how to deal with big data analytics



So many types of services from different providers. Anyway to simplify the management of services for the developer/operator?

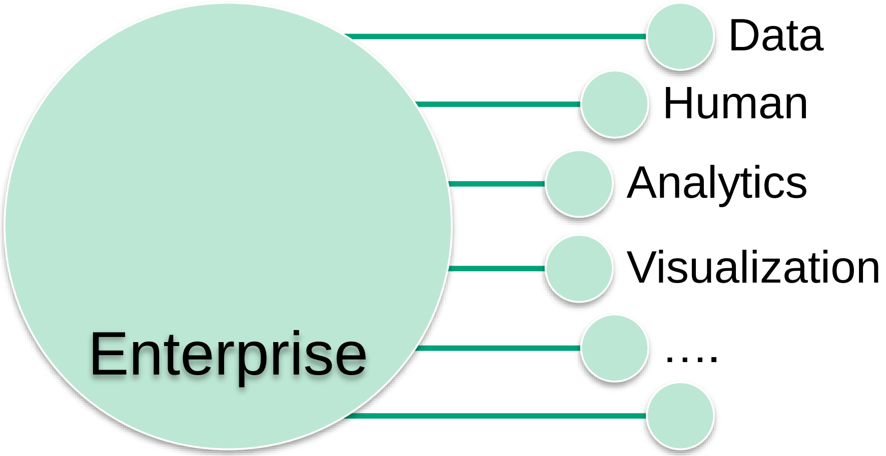
# API MANAGEMENT

# Ecosystem view for advanced service engineering

- Complex data analytics applications → need to understand potential service units from an ecosystem perspective
  - Interdependent systems: Social computing, mobile computing, cloud computing, data management, etc.
  - Different functions (analytics, visualization, communications, etc.)
  - Too many different types of customers (and their interactions)
  - Blending vertical and horizontal analytics

# APIs

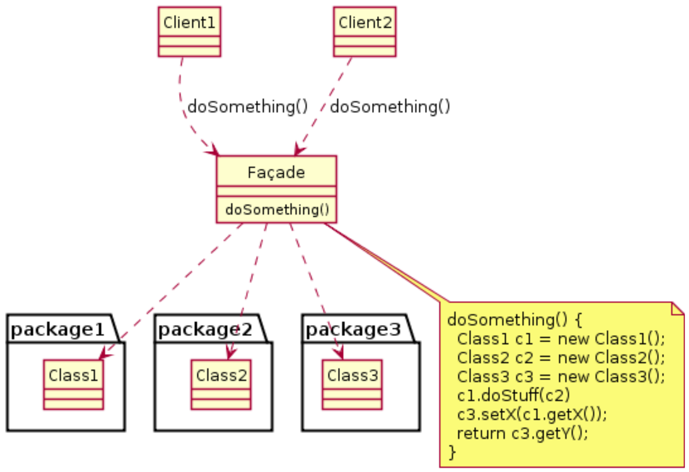
- APIs are key! Why?
  - Enable access to data and function from entities in your ecosystem
  - Virtualization




- An API is an asset
  - We need to have lifecycle, pricing, management, etc.

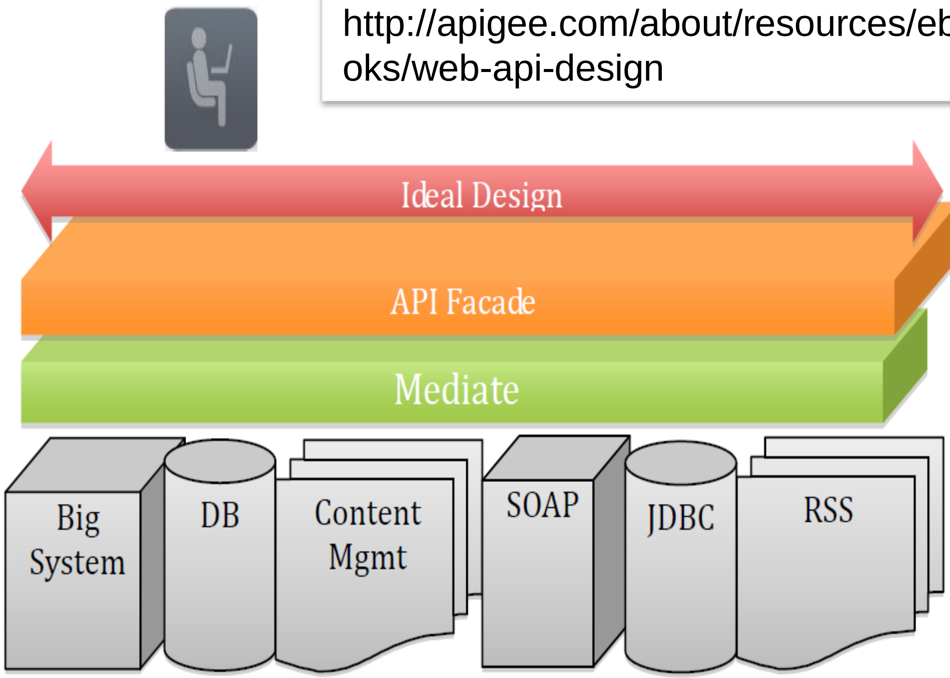
Check http://www.apiacademy.co for some useful tutorials

# API Fasade

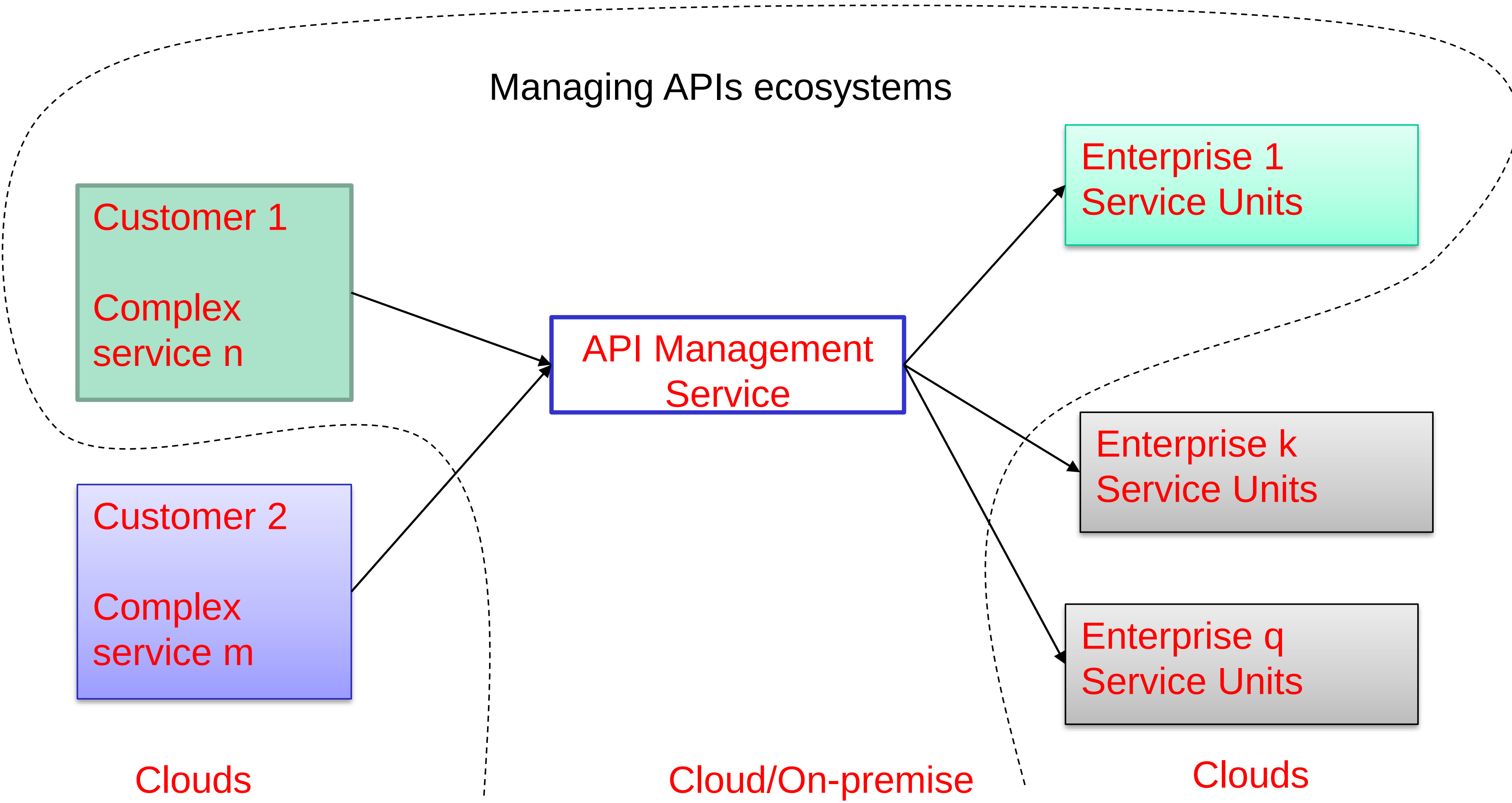



Sourre: https://en.wikipedia.org/wiki/Facade_pattern

Source: Web API Design, Brian Mulloy http://apigee.com/about/resources/ebooks/web-api-design



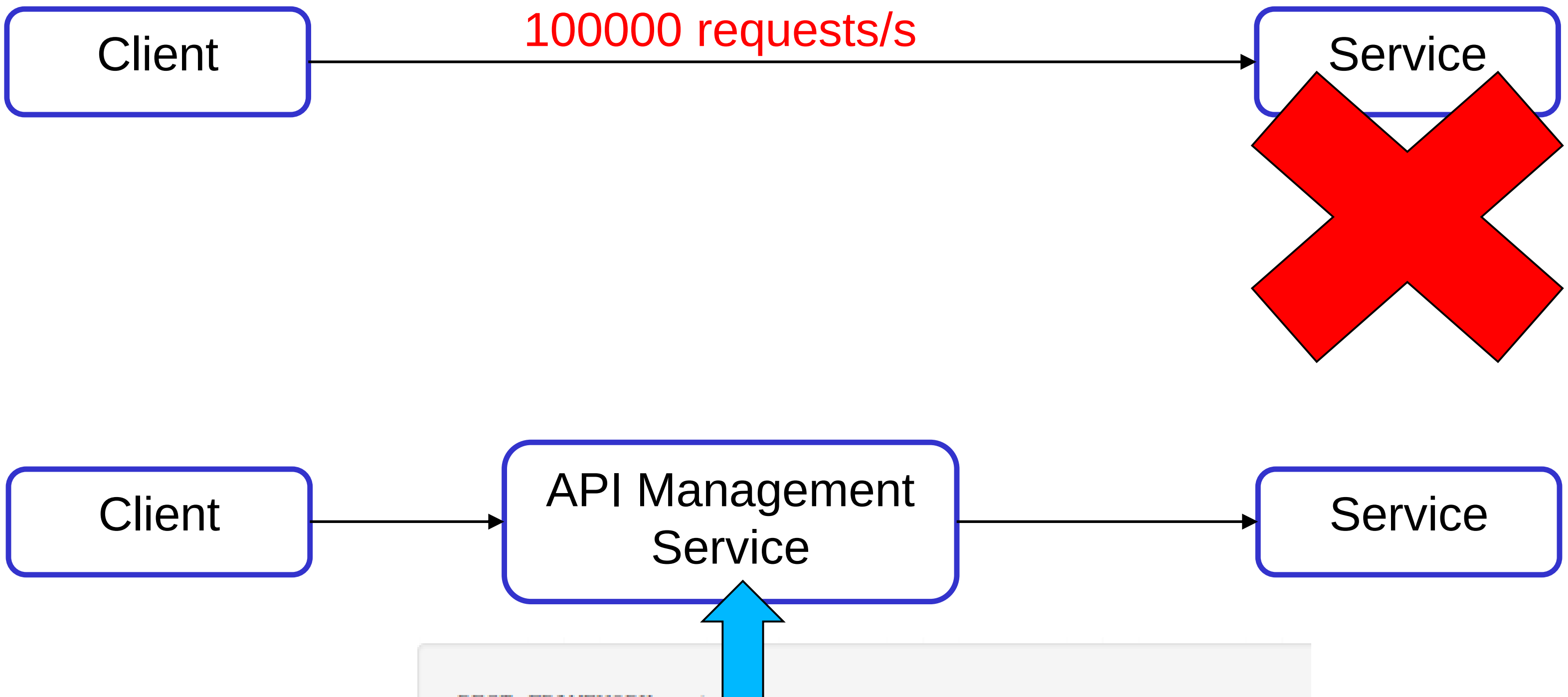
# API management & APIs as a service



# Development of APIs

- Not just the functions behind the APIs
  - This we have learned since a long time
- Emerging (business/service) management aspects
  - Usage control and security
  - Any where from any device for any customer
    - Interfaces (communications, inputs/output formats)
  - APIs as a service:
    - Availability and reliability of APIs are important – think APIs are similar to a service that your client will consume

# Prevent too many accesses?




```
REST_FRAMEWORK = {
    'DEFAULT_THROTTLE_CLASSES': (
        'rest_framework.throttling.AnonRateThrottle',
        'rest_framework.throttling.UserRateThrottle'
    ),
    'DEFAULT_THROTTLE_RATES': {
        'anon': '100/day',
        'user': '1000/day'
    }
}
```

Code: http://www.django-rest-framework.org/api-guide/throttling/#how-throttling-is-determined

## How can we use API management for data/service contracts?

# Issues on APIs management

- Publish
  - Business and operation planning
    - API usage schemes (e.g., pricing, data concerns)
    - API payload transform policies
    - API throttling
  - API publish and discovery  (like service discovery?)
- Management
  - Management roles in enterprises, versions, etc.
- Monitoring and analytics
  - monitoring and analytics information (availability, types of customers, usage frequencies, etc.)

# Some well-known frameworks

- http://apigee.com
- Oracle API management: http://www.oracle.com/us/products/middleware/soa/api-management/overview/index.html
- http://wso2.com/api-management/
- http://www.ca.com/us/lpg/layer-7-redirects.aspx
- https://www.mashape.com/
- http://apiaxle.com/

# Build your own APIs ecosystem

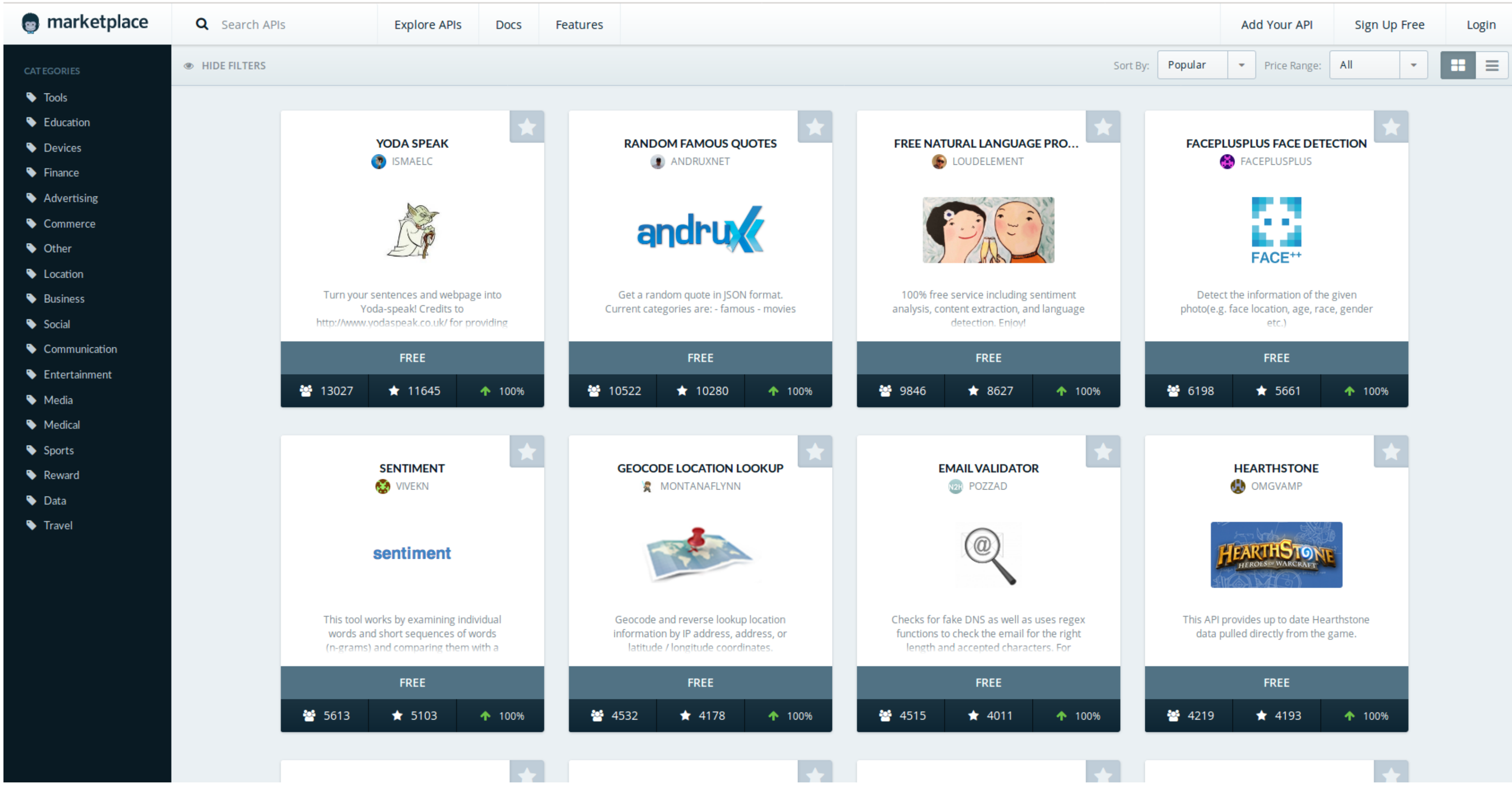
- Which APIs you need? Which ones are crucial for you to build complex services?
  - Data APIs
    - Data collection, Visualization, Analytics APIs
  - Communication
  - Coordination of tasks

→ API management for IoT?

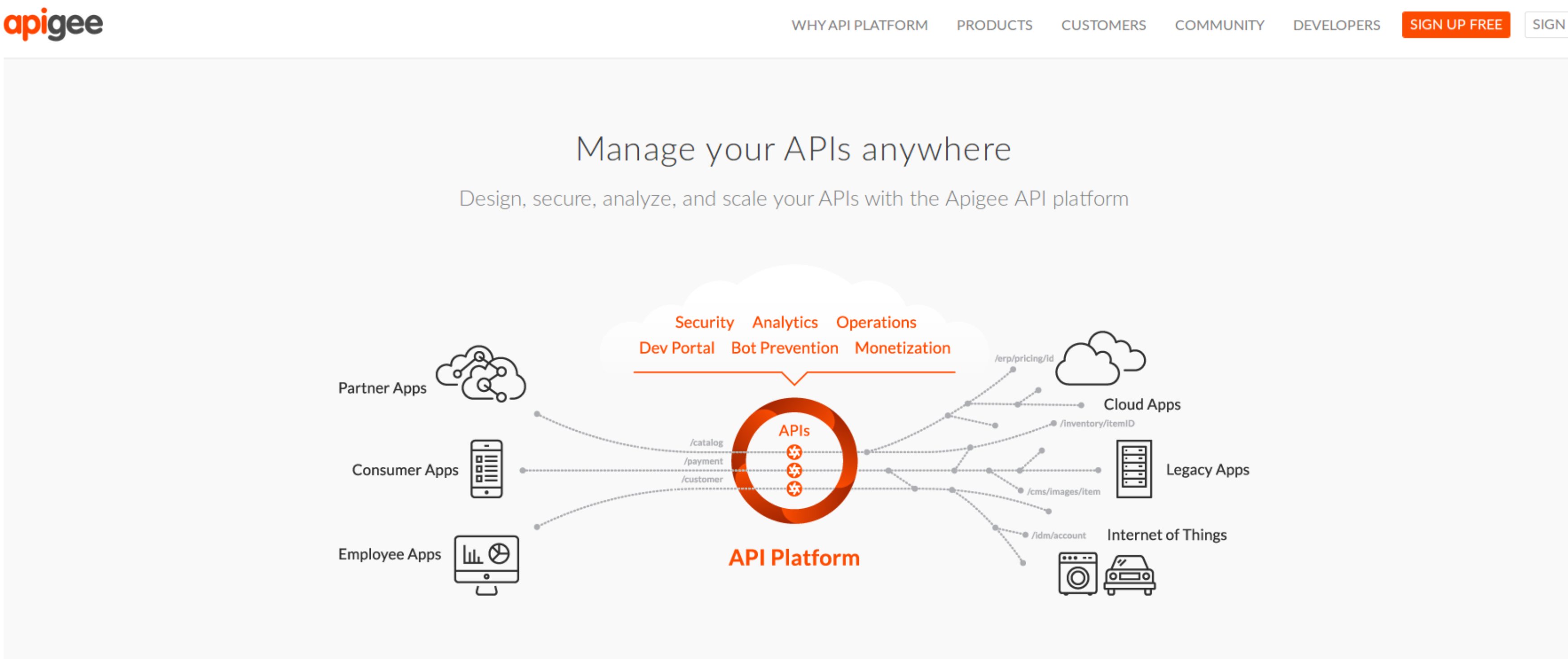
(http://ubiquity.acm.org/article.cfm?id=2822873)

- API marketplaces → your APIs
- Using existing API platforms to manage your APIs

# Examples of an API marketplace



# Use API Management for your mini project?




From https://apigee.com

# Exercises

- Read mentioned papers
- Analyze the relationships between programming models and system infrastructures for data analytics across multiple domains
- Examine http://cloudcomputingpatterns.org and see how it supports data analytics patterns
- Develop some patterns for data analytics across multiple systems
- Setup an API management platform for your work

# Data analytics within a single system

## Some papers


1. Andrew Pavlo, Erik Paulson, Alexander Rasin, Daniel J. Abadi, David J. DeWitt, Samuel Madden, and Michael Stonebraker. 2009. A comparison of approaches to large-scale data analysis. In Proceedings of the 2009 ACM SIGMOD International Conference on Management of data (SIGMOD '09), Carsten Binnig and Benoit Dageville (Eds.). ACM, New York, NY, USA, 165-178. DOI=10.1145/1559845.1559865 http://doi.acm.org/10.1145/1559845.1559865

2. Leonardo Neumeyer, Bruce Robbins, Anish Nair, Anand Kesari: S4: Distributed Stream Computing Platform. ICDM Workshops 2010: 170-177

3. Jerry Chou, Mark Howison, Brian Austin, Kesheng Wu, Ji Qiang, E. Wes Bethel, Arie Shoshani, Oliver Rübel, Prabhat, and Rob D. Ryne. 2011. Parallel index and query for large scale data analysis. In Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis (SC '11). ACM, New York, NY, USA, , Article 30 , 11 pages. DOI=10.1145/2063384.2063424 http://doi.acm.org/10.1145/2063384.2063424

4. Boduo Li, Edward Mazur, Yanlei Diao, Andrew McGregor, Prashant J. Shenoy: A platform for scalable one-pass analytics using MapReduce. SIGMOD Conference 2011: 985-996

5. Fabrizio Marozzo, Domenico Talia, Paolo Trunfio: A Cloud Framework for Parameter Sweeping Data Mining Applications. CloudCom 2011: 367-374

6. Yingyi Bu, Bill Howe, Magdalena Balazinska, Michael D. Ernst: HaLoop: Efficient Iterative Data Processing on Large Clusters. PVLDB 3(1): 285-296 (2010)

# Thanks for your attention

Hong-Linh Truong
Distributed Systems Group, TU Wien
truong@dsg.tuwien.ac.at
dsg.tuwien.ac.at/staff/truong
@linhsolar