Big data service systems: Models, Elasticity, and Platforms

Hong-Linh Truong
Faculty of Informatics, TU Wien

hong-linh.truong@tuwien.ac.at
www.infosys.tuwien.ac.at/staff/truong
@linhsolar
Outline

- Data analytics within a single system
- Data analytics across multiple systems
- APIs management and big data systems
- Principles of elasticity for advanced service-based data analytics
Advanced service-based analytics – which are fundamental engineering questions?
Predictive Maintenance in Telcos

- Complex types of data
- Various services
- Complex analytics/data processing algorithms
Advanced service-based data analytics -- fundamental concepts

Part A

Part B

... 

Part N

System infrastructures

IoT

Edge servers

Local Cloud

Public cloud

Domain 1

Domain 2

Domain n

Applications
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

- System infrastructures
  - Persona Workstation
  - Cluster/HPC
  - Human-based Computing
  - Cloud Services
  - Fog/Edge Computing Nodes
  - IoT Gateways
Fundamental concepts – unit functions

- Function
  - Data Transformation Service
  - Data Processing Service
  - Visualization Service
  - Communication Middleware
  - Data as a Services
    - Enterprise Service Bus
    - Messaging/Queuing
    - File Data Transfer
Fundamental concepts – programming model within units

- Hadoop/MapReduce
- Distributed Streaming Processing
- MPI
- Parallel Database
- Workflow/DataFlow
Fundamental concepts – interfaces between services

- Interface
  - REST
  - Client Library
  - Specific APIs
  - Standard APIs
  - Java
  - Python
  - NodeJS
  - Push
  - Pull
  - ....
Fundamental concepts – services and data concerns

- Data concerns
  - Quality of data
  - Pricing
  - Data Right
  - ...

- Service Concerns
  - QoS
  - Pricing
  - ...

ASE Summer 2018
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
- Big Query
- Hadoop + HDFS
- Amazon RedShift
- Parallel Database (SQL/NonSQL)
- Azure HDInsight
- Apache Spark
- Scientific/Business Workflow

Example - BigQuery (1)

From https://cloud.google.com/bigquery/docs/reference/libraries
Example – BigQuery: complexity

Figure 1: BigQuery structural overview

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

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:
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()
```

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

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

cluster-spark (Google spark installation with n nodes)

Submission command line (your local machine)

Google cloud

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

- With ElasticSearch, MongoDB, Cansassandra, 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 Unit: Characteristics

- Can be simple or complex
  - E.g., a python program based on scikit-learn or a pySpark program or a workflow
- Can be written in different program languages
- Can be deployed and run “as a service”
  - Clear input & output
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 Unit

Read/write data

Cluster file

- NFS
- Lustre

Hadoop File System

Google file system
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

**Data Analytics Unit**

- **Interface**
  - REST/SOAP APIs
  - Mainly for commands and results

- **System**
  - Decouple between analytics unit and database
  - Database as a service 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

**Database-as-a-Service**

**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

**Data Analytics Unit**

**Streaming DaaS**

**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

**Technology**
- StormMQ, RabbitMQ, CloudMQTT, Google Data Hub, Azure Data Hub, ...
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 - example

How many systems?
Programming languages?
Type of data?
Data analytics across multiple systems – programming models (1)

Static data

Input data

Local input data

Hadoop/Spark
Airflow
Etc.

Analytics Results

Servers/Cloud/Cluster

Output data

What are our design concerns?
Data analytics across multiple systems – programming models (2)

Near-realtime data

- E.g., equipment monitoring

- Complex event processing
  - Streaming data processing (e.g. Flink, Kafka, Apex)
  - Other solutions

Input data → Analytics Results → Servers/Cloud/Cluster → Output data

What are our design concerns?
Cloud services and big data analytics

Data sources (sensors, files, database, queues, log services)

Messaging systems (e.g., Kafka, AMQP, MQTT)

Storage and Database (S3, InfluxDB, HDFS, Cassandra, MongoDB, Elastic Search etc.)

Stream processing systems (e.g. Apex, Storm, Flink, WSO2, Google Dataflow)

Batch data processing systems (e.g., Hadoop, Airflow, Spark)

Elastic Cloud Infrastructures (VMs, dockers, OpenStack elastic resource management tools, storage)

Operation/Management/Business Services

Warehouse 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 – legacy 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 AND BIG DATA
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](http://www.apiacademy.co) for some useful tutorials
API Fasade

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

Source: https://en.wikipedia.org/wiki/Facade_pattern
API management & APIs as a service

Managing APIs ecosystems

Customer 1
Complex service n

API Management Service

Enterprise 1
Service Units

Enterprise k
Service Units

Enterprise q
Service Units

Customer 2
Complex service m

Clouds

Cloud/On-premise

Clouds
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
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://wso2.com/api-management/
- 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?

Manage your APIs anywhere
Design, secure, analyze, and scale your APIs with the Apigee API platform

From https://apigee.com
What would be the relationship between API management and big data?
Aspects:
- Data access and contract
- New source of data
- Data analytics
Changes in Application, Analytics and data

All are changing internally. Can we keep the API remains and new APIs are added?
Example of Architecture Design from Amazon

Figure source: https://aws.amazon.com/answers/big-data/data-lake-solution/
PRINCIPLES OF ELASTICITY FOR BIG DATA SYSTEMS
Elasticity in (big) data analytics

- More data $\rightarrow$ more compute resources (e.g. more VMs)
- More types of data $\rightarrow$ more activities $\rightarrow$ 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)

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 processes, 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
General software design concept: Lifecycle of applications and elasticity

- Elasticity specification
- Control processes
- Orchestrate concrete operations

Operate Time

Monitoring information

Check: https://doi.org/10.1016/j.procs.2016.08.276
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


Thanks for your attention

Hong-Linh Truong
Faculty of Informatics, TU Wien
hong-linh.truong@tuwien.ac.at
www.infosys.tuwien.ac.at/staff/truong
@linhsolar