Aalto University
School of Science

Architecting Big Data Platforms

Hong-Linh Truong
Department of Computer Science
linh.truong@aalto.fi, https://rdsea.github.io

Learning objectives

- Understand key issues in designing a big data platform
- Learn key architecture design issues
- Be familiar with the course' selected big data platform technologies

Your big data platform story - an evolving scenario

“Your team has to build a big data platform for X types of data. Data will be generated/collected from N data sources. We expect to have 10+ GBs/day of data to be ingested into our platform. We will have to serve K thousands of requests for different types of analytics – to be determined. Our response time for an analytics request should be in t milliseconds. Our services should not be …”

You may have several questions?

- Do we have to support multiple types of data?
- How do data pipelines and data load look like?
- How to enable different data processing models?
- Which runtime parameters must be monitored? Which service level and data metrics must be guaranteed?
- To where we should distribute/deploy our components?
- Which part of the platform we must manage by ourselves and which part will be fully managed by other providers?
- How to design elastic big data infrastructures?
- Etc.

Your Big Data Platform story starts with Big Data Platform architectures!

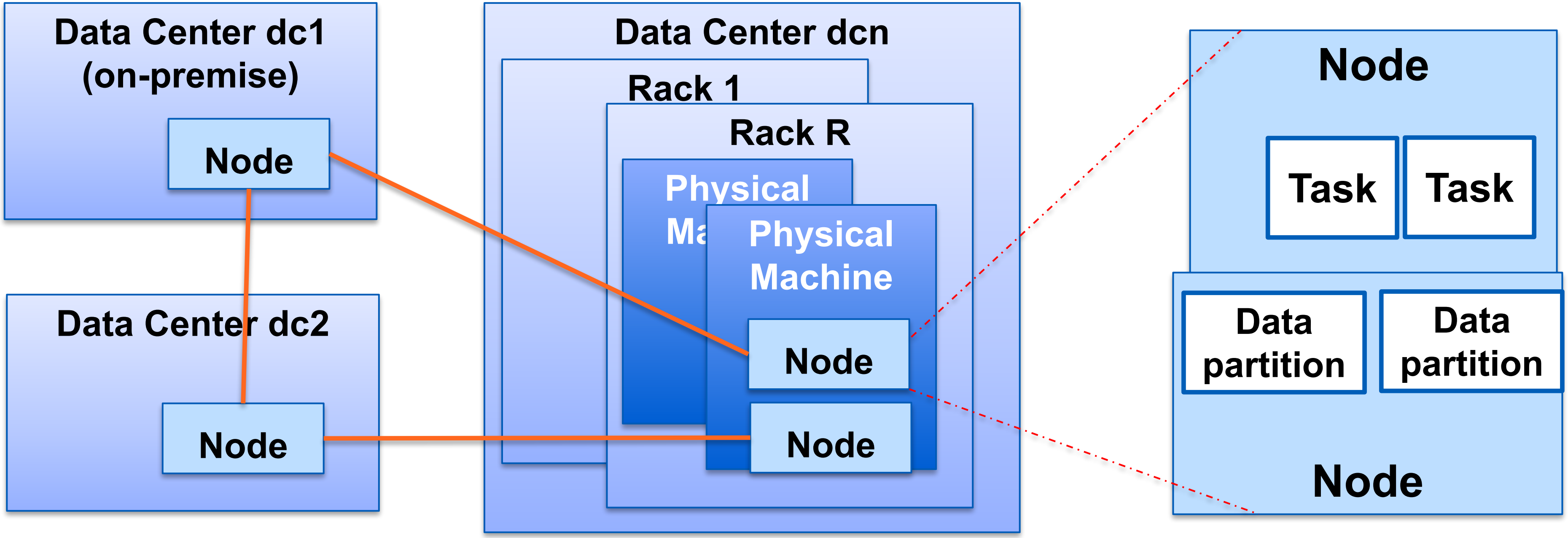
To architect the platform centered around big data and data intensive activities!

Underlying computing infrastructures is for data intensive tasks

- **Computing resources and services**
  - many machines, virtual infrastructures, different types of services
- **Distributed infrastructures from different administrative domains**
  - in multiple data centers, locations and countries
  - with different security and network policies
- **Diverse service level objectives (SLO) and service level agreements (SLAs)**
  - performance, service failure, cost, privacy/security …
  - data and data governance

Understanding the underlying infrastructures for big data platforms

large-scale distributed infrastructures ⇒ hybrid and or multi cloud/edge



several data intensive tasks are optimized with rack/topology awareness

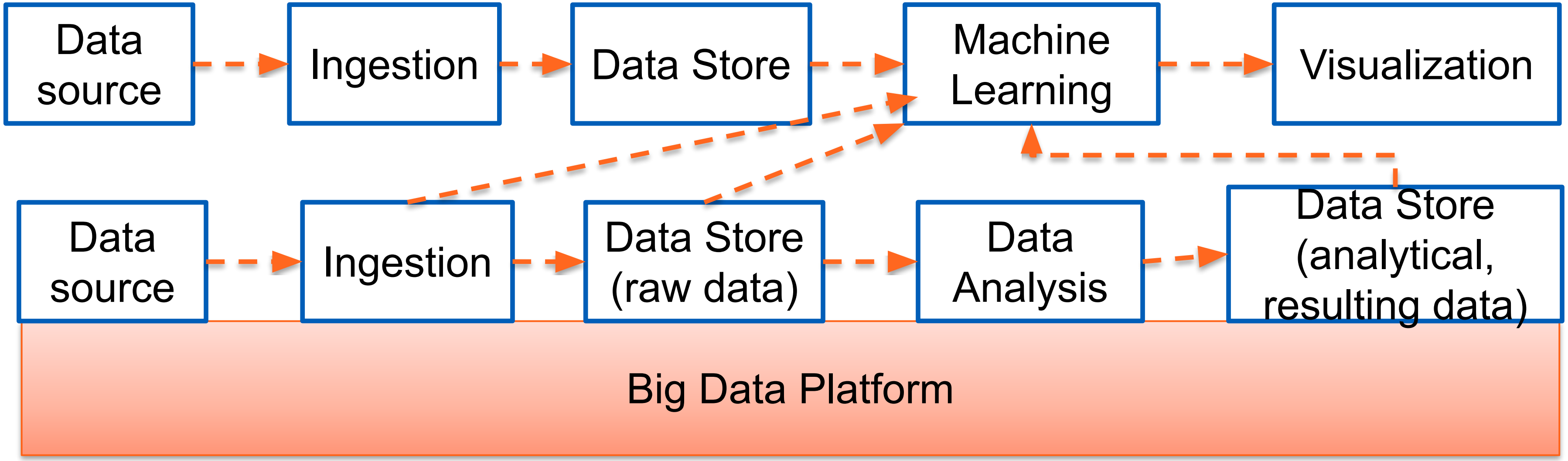
Data-centric development & operations

- **Data ingestion and ETL (Extract, Transform, Load)**
  - from various data sources we move data into the platform
- **Data storing and management**
  - ingested data stored and managed using different types of storages and databases
- **Data analysis and (Machine) learning**
  - data processed, analyzed and learned, finding insights and creating ML models
  - data **at rest** vs data **in motion**
- **Reporting and visualization**
  - pa or making de

Data Governance at very large-scale

Big Data Pipelines

Multiple big data pipelines can be constructed atop a big data platform (and across distributed infrastructures)



Handling multiple types of data

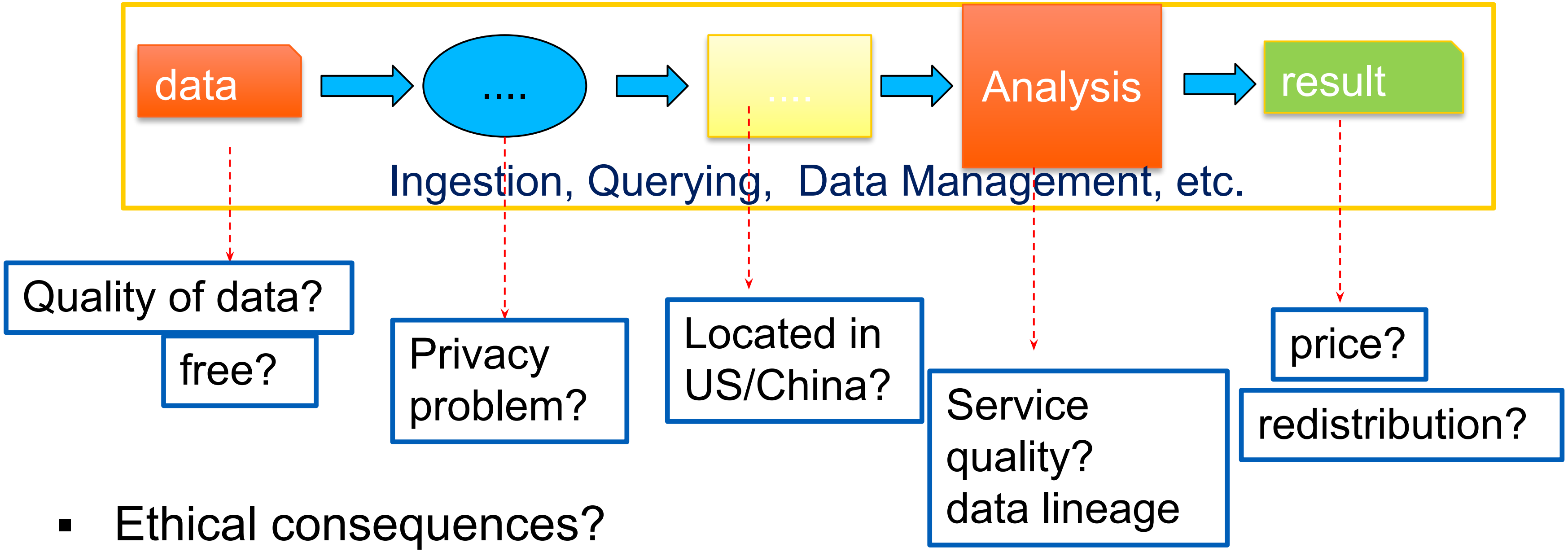
- **Hardly to avoid the support for multiple types of data**
  - tenant requirements and application needs, given data as “products”/”assets”
- **Multiple types of data**
  - different characteristics and values
- **Any elastic solution that ensures minimum changes to support generalization and extensibility**
  - multi-model databases, microservices of multiple of databases or data lake
  - new workloads, new types of customers

Data governance: concerns & SLAs

- **Key common problems:**
  - data quality, data lineage,
- **Ingesting data:** ingestion of data under V*
  - mapping and transforming data, data validation/quality control
- **Storing data**
  - data sharding and consistency, data backup, retention, etc.
  - the impact of the rights to remove data
- **SLA multitenancy versus single tenancy**
  - security, privacy, performance, reliability and maintenance?

The volume of data is **increasing** but its **usefulness may not** because of the bad data quality

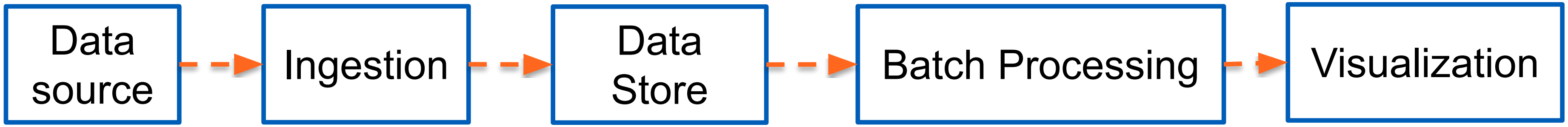
Data concerns: example



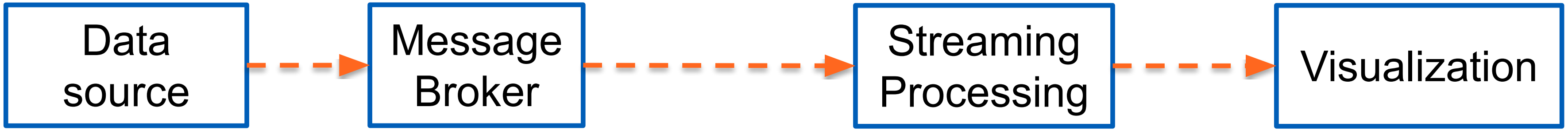
- Ethical consequences?
- Regulation-compliant platforms: e.g., GDPR

Fast/slow reliable processing

big data but not near real-time, e.g., take customer transaction files from companies and move to data centers for analytics



fast, small IoT data in near real-time flows, e.g. position of cars



Software services design goals

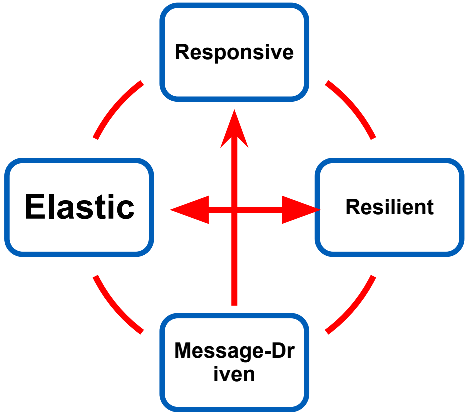
For dealing with V*

- Responsive: guarantee quality of services
- Resilient: deal within failures
- Elastic: deal with different workload and quality of analytics
- Loosely coupling: support reusability, composition, and extensibility

Designs must address various aspects for big data

- **Responsive:**
  - distributed computing, multi layer optimization
- **Resilient:**
  - replication, containment, isolation
- **Elastic:**
  - sharding, replication, load balancing, scale up/out
- **Message-driven:**
  - loosely coupling with messages, non-blocking protocols, location-independent

Reactive systems



Source: https://www.reactivemanifesto.org/

Efficiency and sustainability

- **Highly efficient might not be good for sustainability?**
- **Sustainability within big data platforms**
  - energy consumption, reusability, extensibility

Design and implementation

- Reliability
- Resilience
- Elasticity

→ ▪ *Sustainability* →

Sus. goal | Platform design | Concrete techniques | Verification

How to have it for

Aalto University
School of Science

Distributed systems of components are used to manage, ingest data and process data

Partitioning: splitting functionality & data

- **Breakdown the complexity**
  - easy to implement, replace and compose
  - deal with performance, scalability, security, etc.
  - support teams in DevOps
  - cope with technology changes
- **Many things are related to the current trend of microservices and domain-driven designs**
- **But tradeoffs with management w.r.t data resources**

Example of functional and data partitioning

Service-oriented components

Microservices and domain-oriented microservices for data

Serverless functions/function as-a service

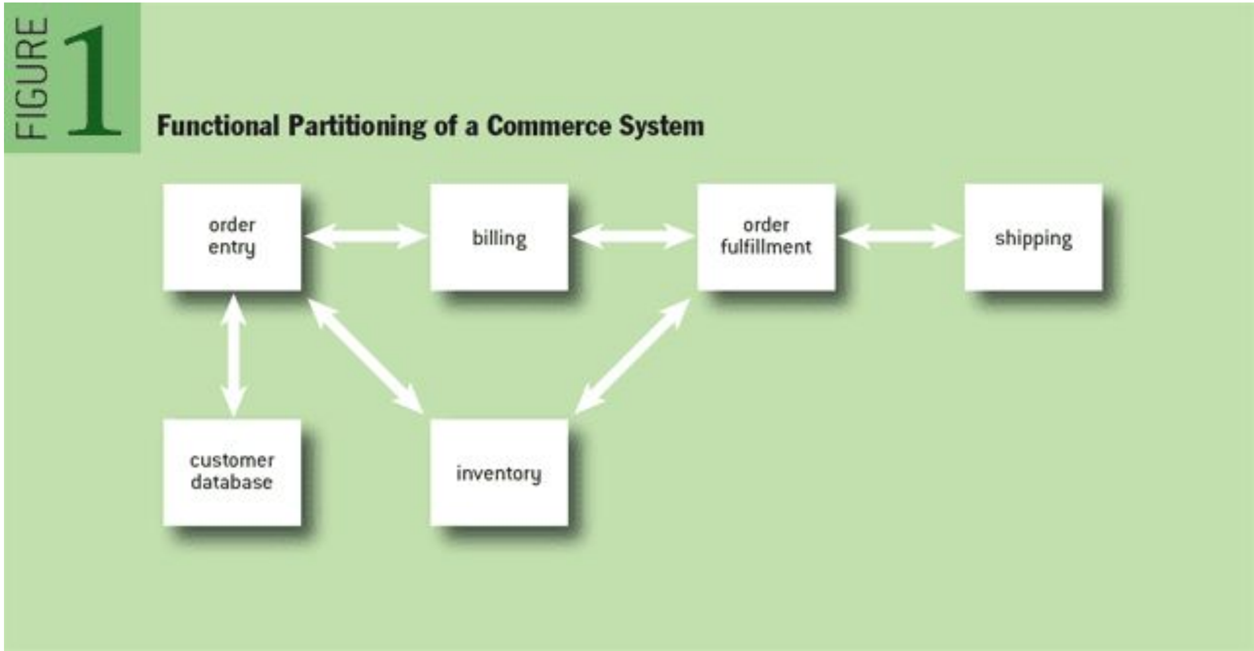


Figure source: http://queue.acm.org/detail.cfm?id=1971597

Example of functional and data partitioning

Data sharding

Multi data spaces

Multi data services

Multiple data infrastructures

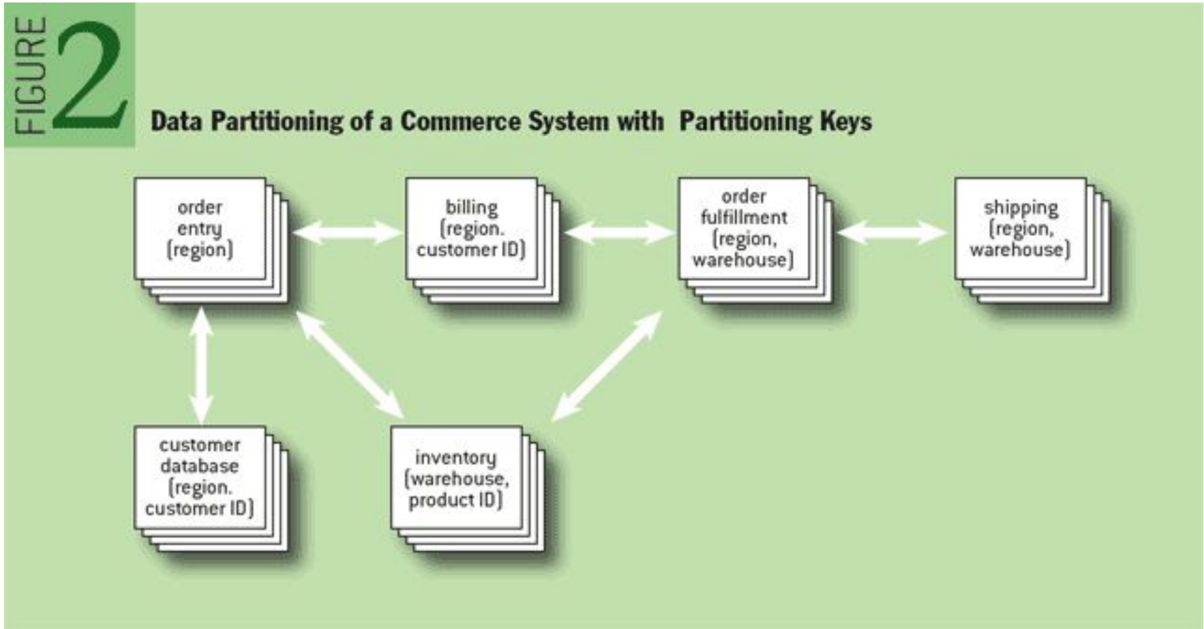
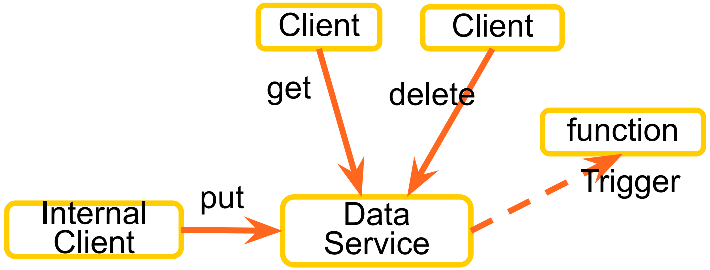


FIGURE 2
Data Partitioning of a Commerce System with Partitioning Keys
order entry (region)
billing (region. customer ID)
order fulfillment (region, warehouse)
shipping (region, warehouse)
customer database (region. customer ID)
inventory (warehouse, product ID)

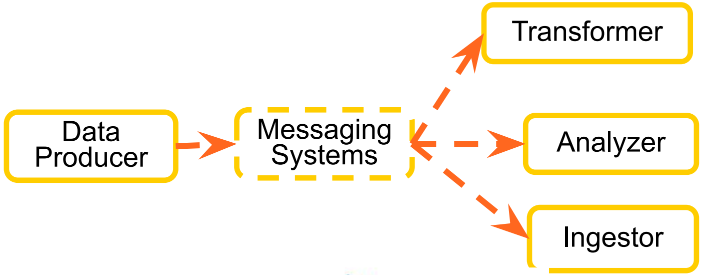
Figure source: http://queue.acm.org/detail.cfm?id=1971597

Interaction: multiple models

- **Protocols**
  - REST, gRPC, Message Passing, Stream-oriented Communication
- **Models**
  - One-to-many, many-to-one, many-to-many
  - Synchronous/asynchronous calls
  - Internal data exchange versus open/external exchange

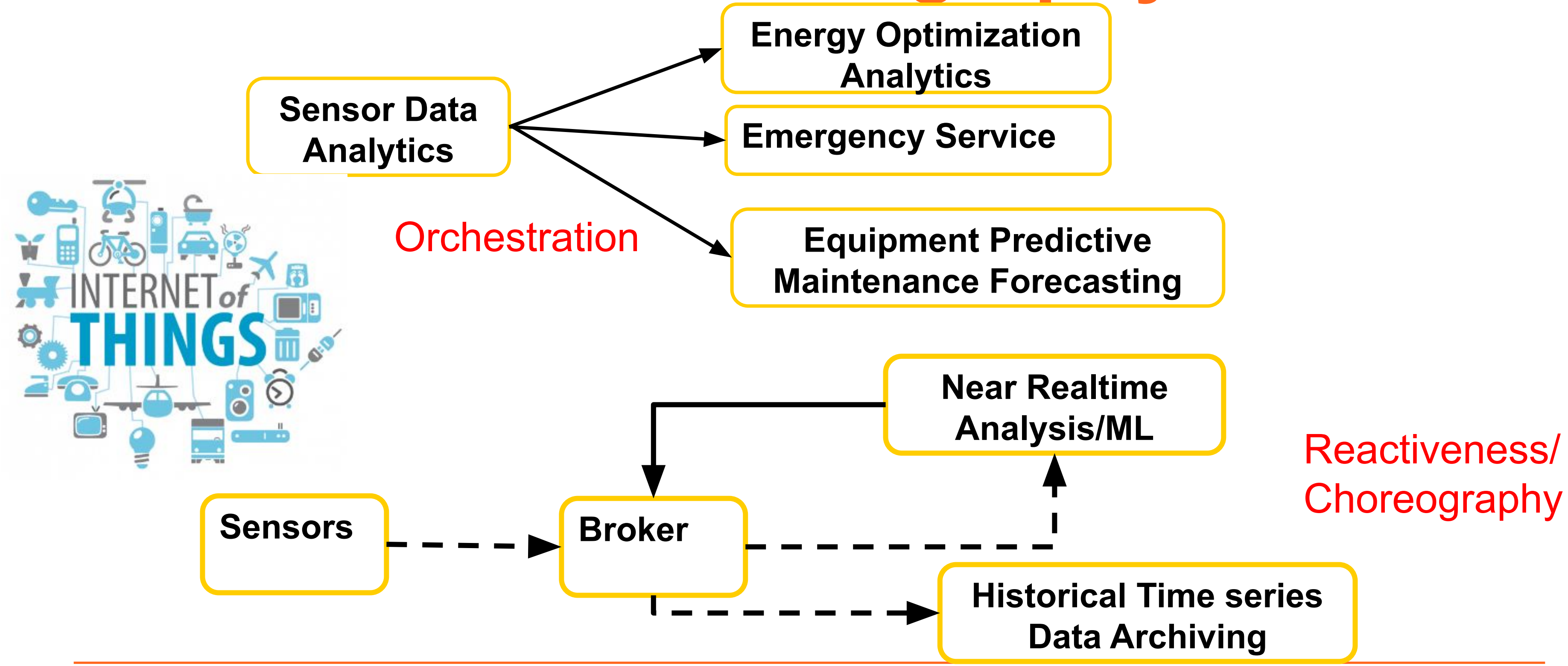


Amazon S3/MongoDB



druid Apache hudi

Coordination: Orchestration and Reactiveness/Choreography



Sensor Data Analytics

Energy Optimization Analytics

Emergency Service

Equipment Predictive Maintenance Forecasting

Orchestration

Near Realtime Analysis/ML

Sensors

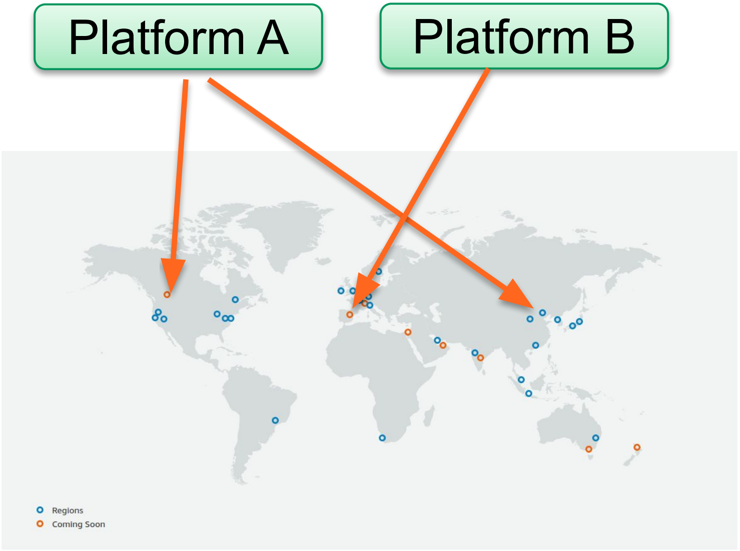
Broker

Historical Time series Data Archiving

Reactiveness/ Choreography

Distribution: Edge or Data Centers?

- **Big data & components components can be distributed in different places!**
  - *performance, security, regulation, energy efficiency*
- **Global deployment or not?**
- **Move analytics/work or move data?**



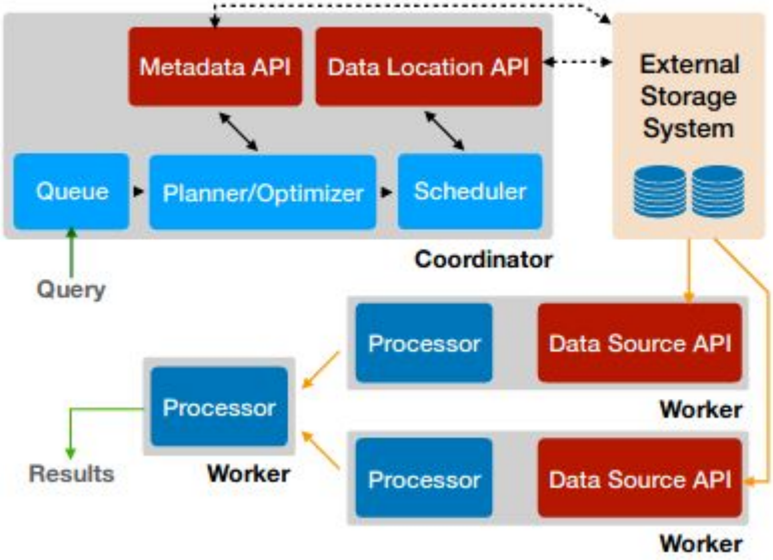
Map of AWS infrastructure (08.01.2022)
Source: https://aws.amazon.com/about-aws/global-infrastructure/

An outage can lead to a huge problem. Example: https://www.thousandeyes.com/blog/aws-outage-analysis-dec-7-2021

Scalability & elasticity

Lyft Presto Gateway

Presto: https://prestodb.io/



"As of today we have 60 PB of query-able event data stored in an S3 based data lake and about 10 PB of raw data is being scanned every day using Presto"

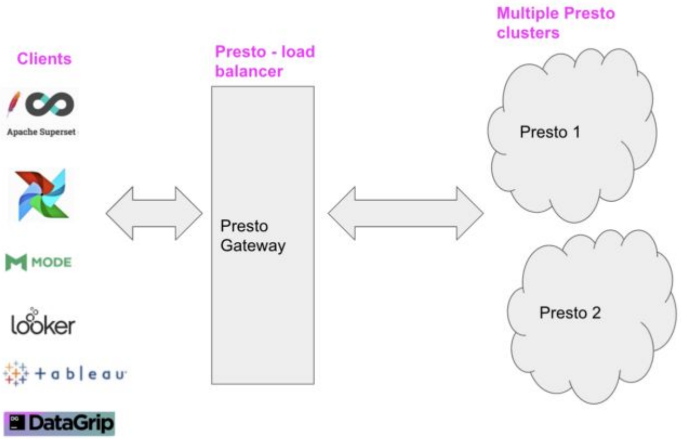
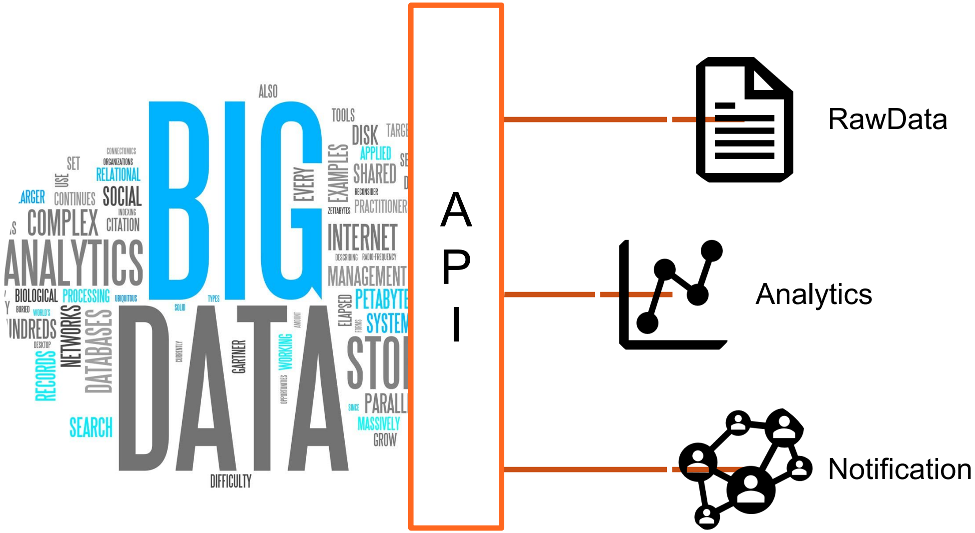


Figure source: *Presto: SQL on Everything https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8731547&tag=1*

Figure and text source: *https://eng.lyft.com/presto-infrastructure-at-lyft-b10adb9db01*

API for Platform as a Service

- **APIs are key! Why?**
  - enable customers access to data and analysis functions from your big data platforms without worrying about changes within your platforms
  - enable virtualization and management (hide internal, control access, throttling)
  - establish protocols for data exchanges

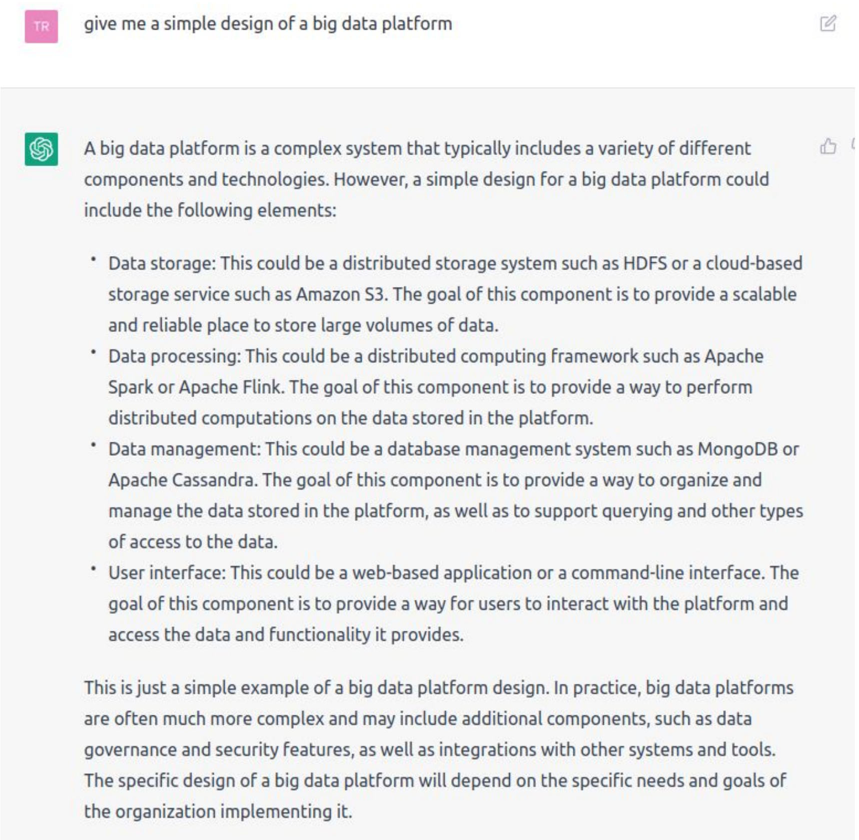


Delivery data as “asset”/”product”

- **FAIR principles (https://www.nature.com/articles/sdata201618)**
    - Findable, Accessible, Interoperable and Reusable
- **Product/asset characteristics**
- **Consequences for the platform**
    - Metadata and governance
    - Service-level objectives and quality controls for data
    - Search
    - Access control policies

Common, high-level architecture view with popular state-of-the art technologies for our study

Your big platform suggested by ChatGPT



TR give me a simple design of a big data platform

A big data platform is a complex system that typically includes a variety of different components and technologies. However, a simple design for a big data platform could include the following elements:

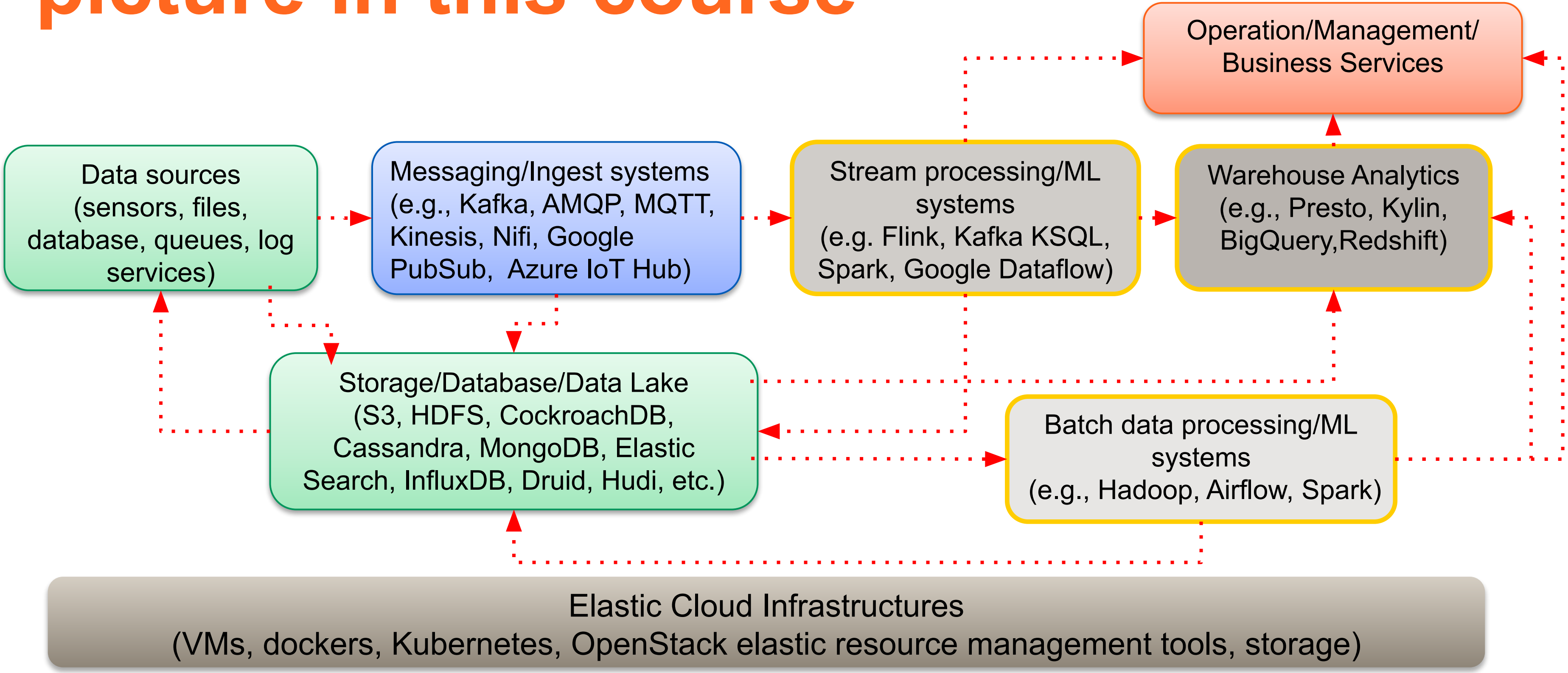
- Data storage: This could be a distributed storage system such as HDFS or a cloud-based storage service such as Amazon S3. The goal of this component is to provide a scalable and reliable place to store large volumes of data.
- Data processing: This could be a distributed computing framework such as Apache Spark or Apache Flink. The goal of this component is to provide a way to perform distributed computations on the data stored in the platform.
- Data management: This could be a database management system such as MongoDB or Apache Cassandra. The goal of this component is to provide a way to organize and manage the data stored in the platform, as well as to support querying and other types of access to the data.
- User interface: This could be a web-based application or a command-line interface. The goal of this component is to provide a way for users to interact with the platform and access the data and functionality it provides.

This is just a simple example of a big data platform design. In practice, big data platforms are often much more complex and may include additional components, such as data governance and security features, as well as integrations with other systems and tools. The specific design of a big data platform will depend on the specific needs and goals of the organization implementing it.

dated: 03.12.2022

Aalto University
School of Science

Our big data at large-scale: the big picture in this course


Operation/Management/ Business Services
Data sources (sensors, files, database, queues, log services)
Messaging/Ingest systems (e.g., Kafka, AMQP, MQTT, Kinesis, Nifi, Google PubSub, Azure IoT Hub)
Stream processing/ML systems (e.g. Flink, Kafka KSQL, Spark, Google Dataflow)
Warehouse Analytics (e.g., Presto, Kylin, BigQuery,Redshift)
Storage/Database/Data Lake (S3, HDFS, CockroachDB, Cassandra, MongoDB, Elastic Search, InfluxDB, Druid, Hudi, etc.)
Batch data processing/ML systems (e.g., Hadoop, Airflow, Spark)
Elastic Cloud Infrastructures (VMs, dockers, Kubernetes, OpenStack elastic resource management tools, storage)

Thanks!

Hong-Linh Truong
Department of Computer Science

rdsea.github.io