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 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 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 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.

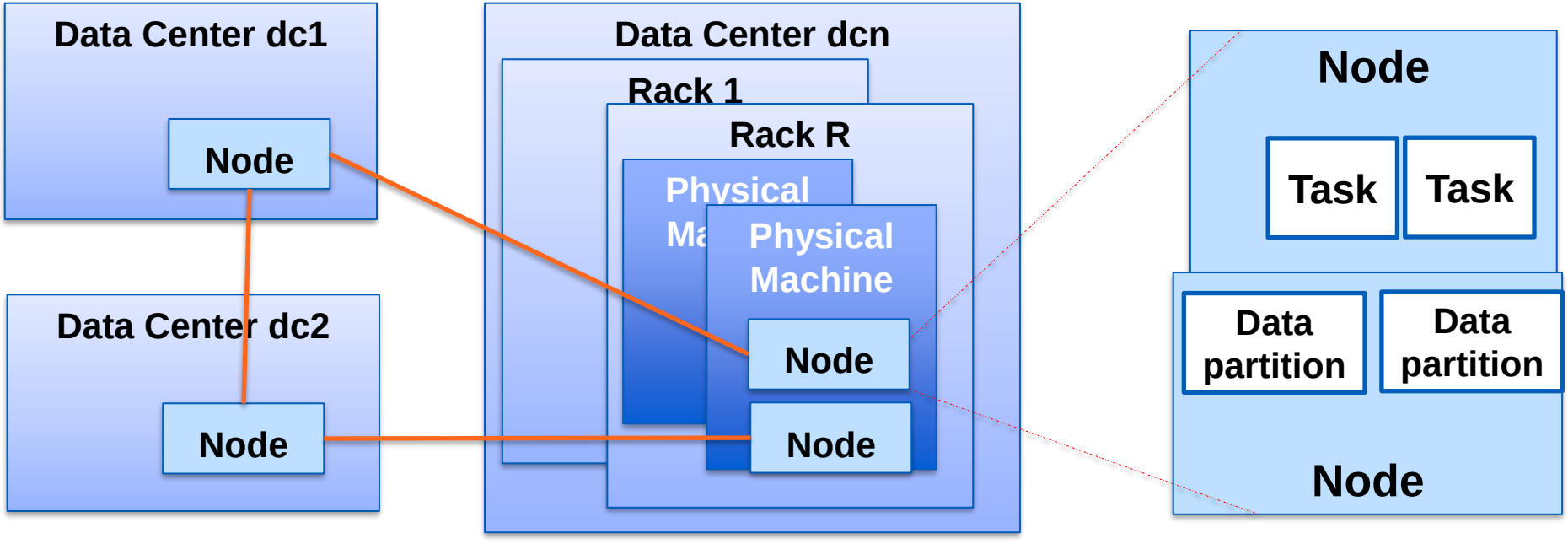
Your Big Data Platform story starts with Big Data Platform architectures!

To architect the platform centered around data!

Understanding the underlying computing infrastructures

- **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 …

Understanding the underlying infrastructures for big data platforms



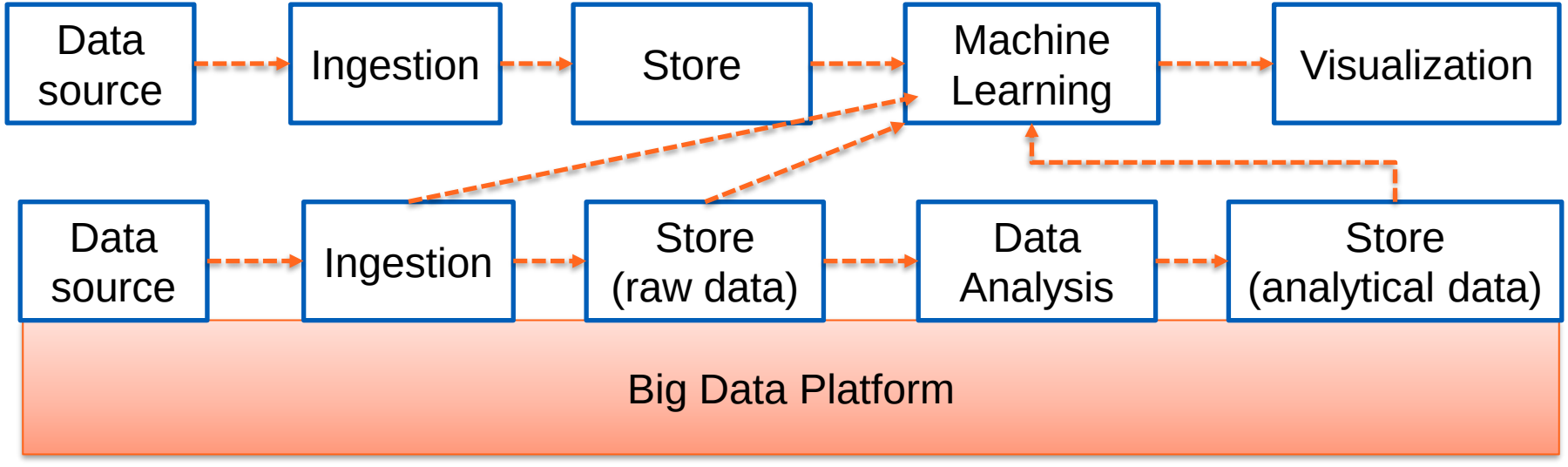
Remember: large-scale distributed infrastructures!

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 will be stored and managed using different types of storages and databases
- **Data analysis and (Machine) learning**
  - data within platforms will be processed, analyzed and learned to improve data, find insights and to create models
  - data at rest vs data in motion
- **Reporting and visualization**
  - patterns/insights in data will be interpreted and presented for decision-making, reporting and creating stories

Big Data Pipelines

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



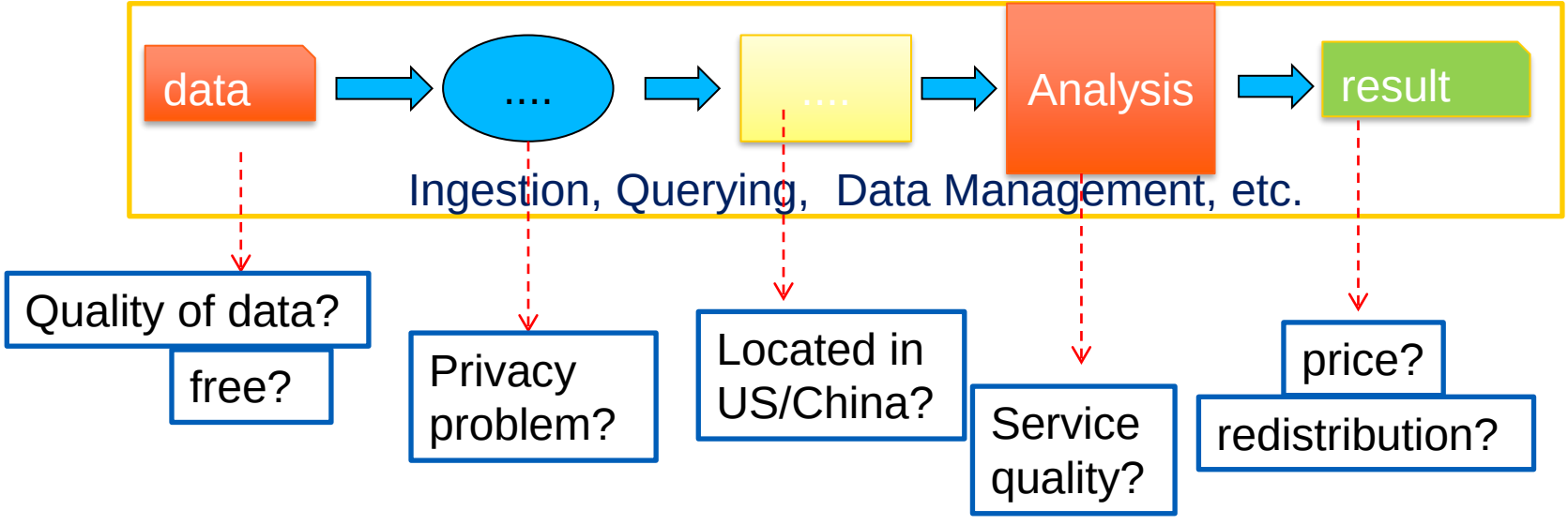
Handling multiple types of data?

- **First important aspect: you don't have to support multiple types of data**
  - but are you sure that you will not have this in the future?
- **Multiple types of data**
  - any linked models among them?
- **Any elastic solution that ensures minimum changes to support generalization and extensibility**
  - e.g., multi-model databases, microservices of multiple of databases or data lake

Data concerns and SLAs

- **Ingesting data**
  - mapping and transforming data
  - ingestion of data under V*
  - data validation/quality control during ingestion
- **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?

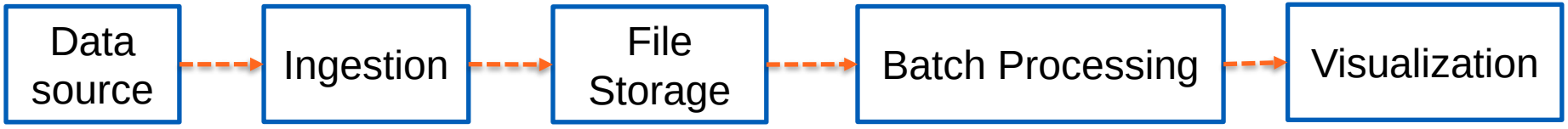
Data concerns: example



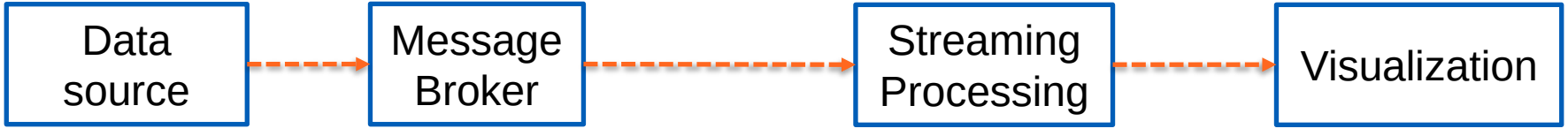
- Ethical consequences?
- Regulation-compliant platforms: e.g., GDPR

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



Design goals

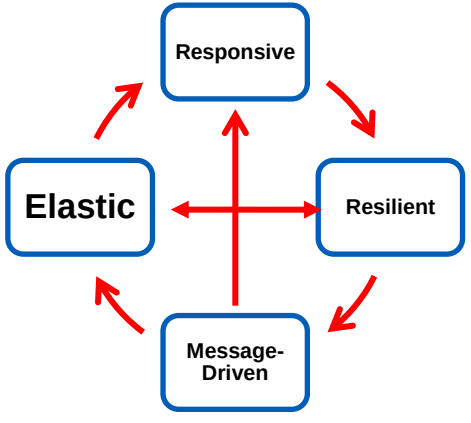
For dealing with V*

- **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

- **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?**
- **Sustainability within big data platforms**
  - energy consumption, reusability, extensibility

Design and implementation

- **Reliability**
- **Resilience**
- **Elasticity**

How to have it for

- ***Sustainability***

Sus. goal | Platform design | Concrete techniques | Verification

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

Example of functional and data partitioning

Service-oriented components

Microservices and domain-oriented microservices

Serverless functions/function as-a service

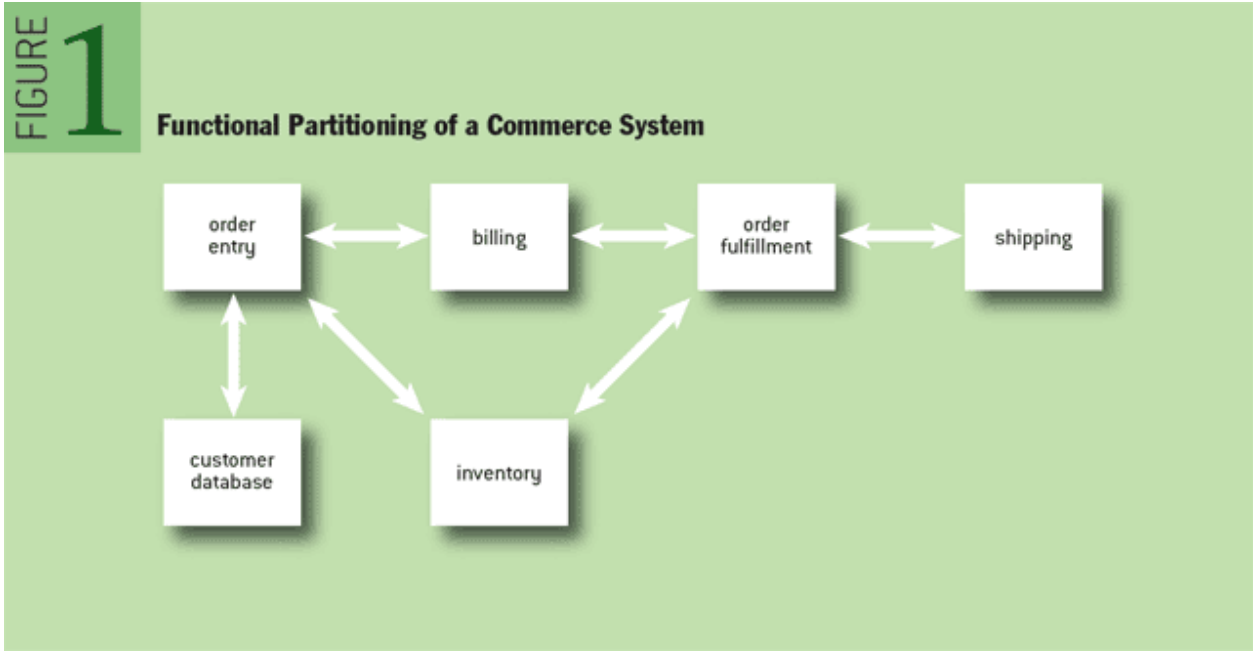


Figure source: http://queue.acm.org/detail.cfm?id=1971597

Example of functional and data partitioning

Data sharding

Multi data spaces

Multi data services

Multiple data infrastructures

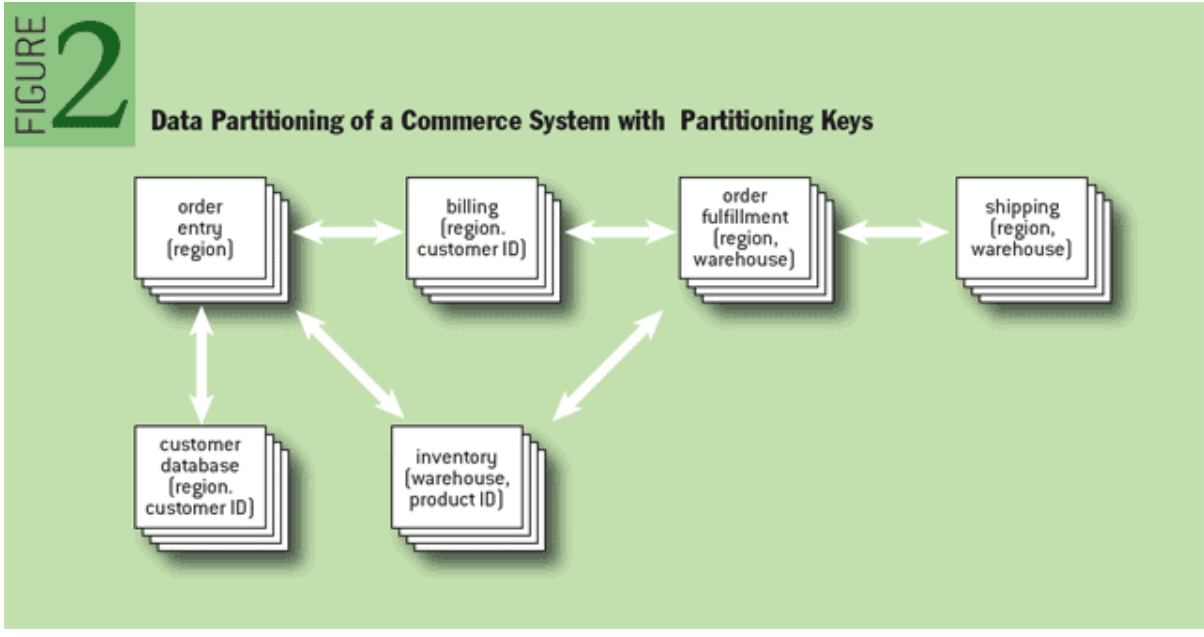


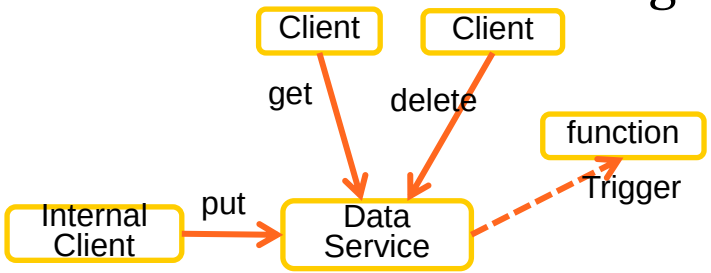
Figure source: http://queue.acm.org/detail.cfm?id=1971597

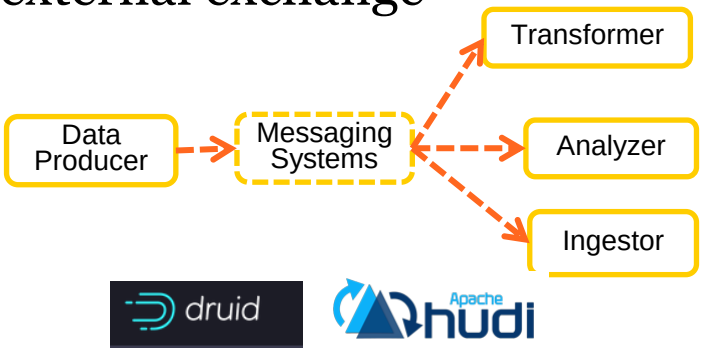
Aalto University
School of Science

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

Interaction: Complex interactions

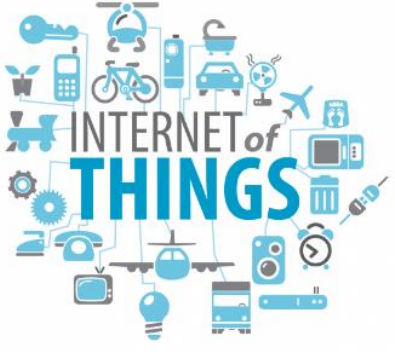
- **Protocols:**
  - REST, gRPC, Message Passing, Stream-oriented Communication
- **Models**
  - One-to-many, many-to-one, many-to-many
  - Synchronous/asynchronous calls
  - Internal data exchange versus open/external exchange

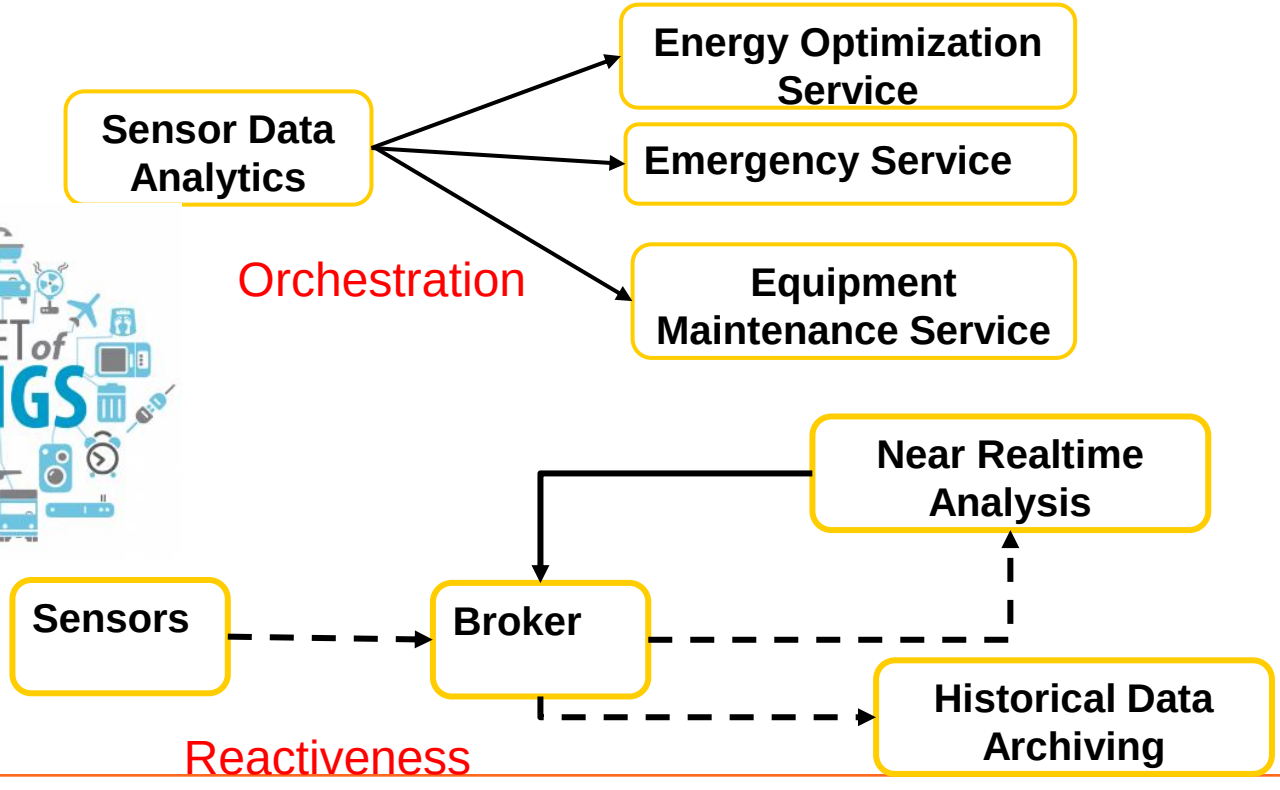




Amazon S3/MongoDB

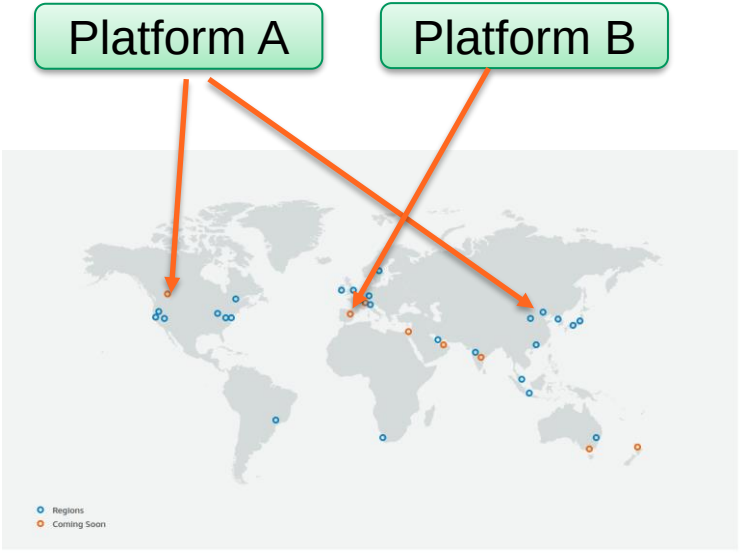
Coordination: Orchestration and Reactiveness





Distribution: Edge or Data Centers?

- **Big data & components components can be distributed in different places!**
- **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

Presto: https://prestodb.io/

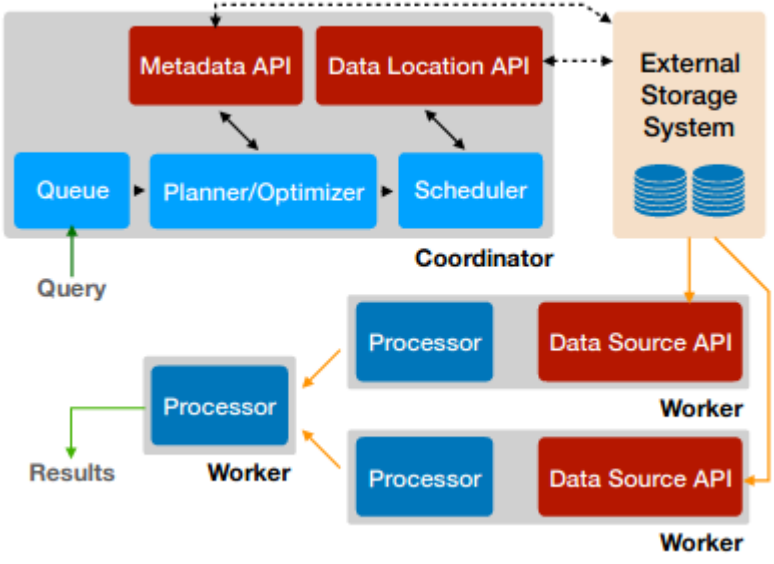


Figure source: Presto: SQL on Everything https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8731547&tag=1

Lyft Presto Gateway

"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"

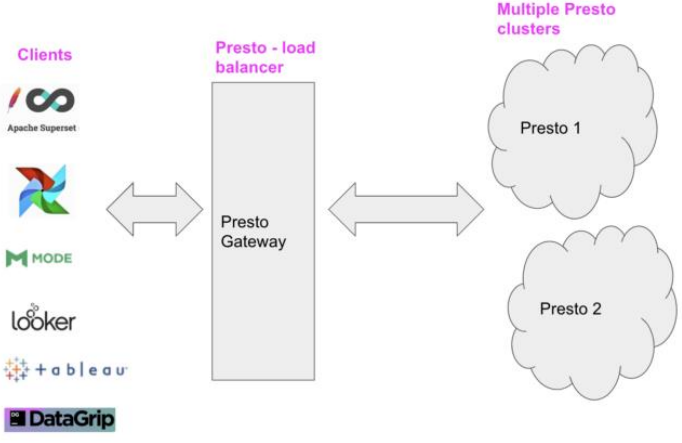
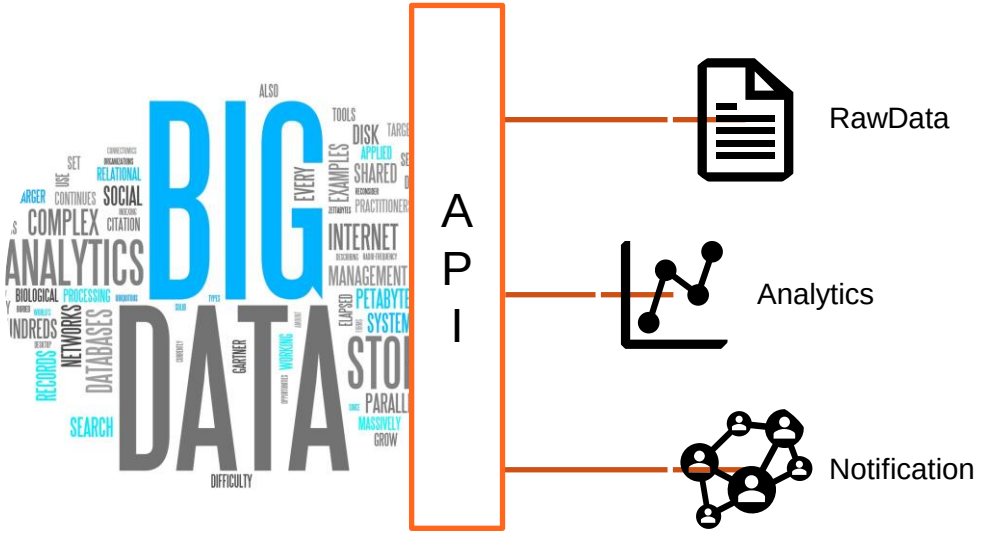


Figure and text source: https://eng.lyft.com/presto-infrastructure-at-lyft-b10adb9db01

API for Platform as a Service

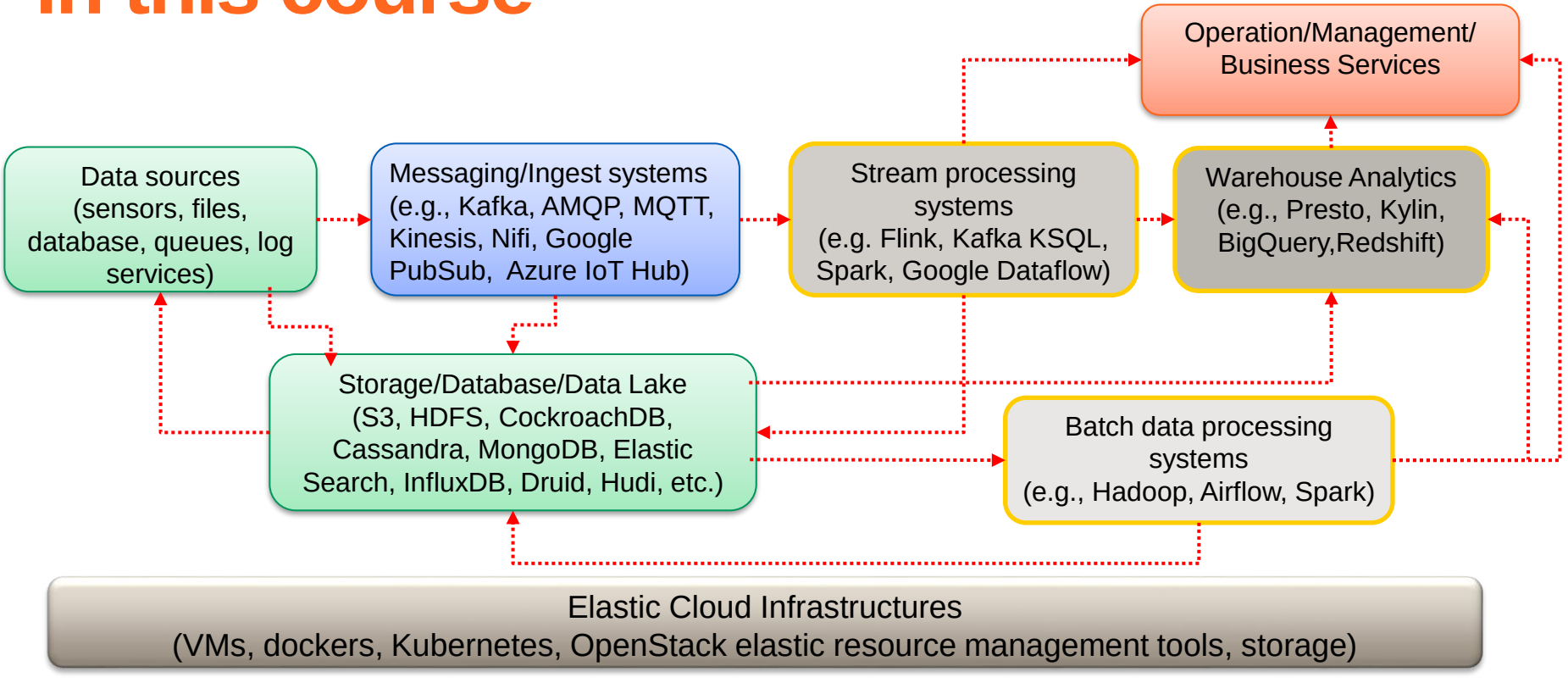
- **APIs are key! Why?**
  - enable customers access to data and analysis functions from your big data platforms without worrying about changes within your platforms
  - virtualization and management (hide internal, control access, throttling)



Which API would you publish? And how other concepts are related, e.g. API Gateways for Load balancing and Fault-Tolerance?

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

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



Thanks!

Hong-Linh Truong
Department of Computer Science

rdsea.github.io