Aalto University
School of Science

Architecting Big Data Platforms

Hong-Linh Truong
Department of Computer Science
linh.truong@aalto.fi

07.1.2026

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 example of a 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?
- Which are the main building blocks/sub systems?
- 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.

Starts with Big Data Platform architectures!

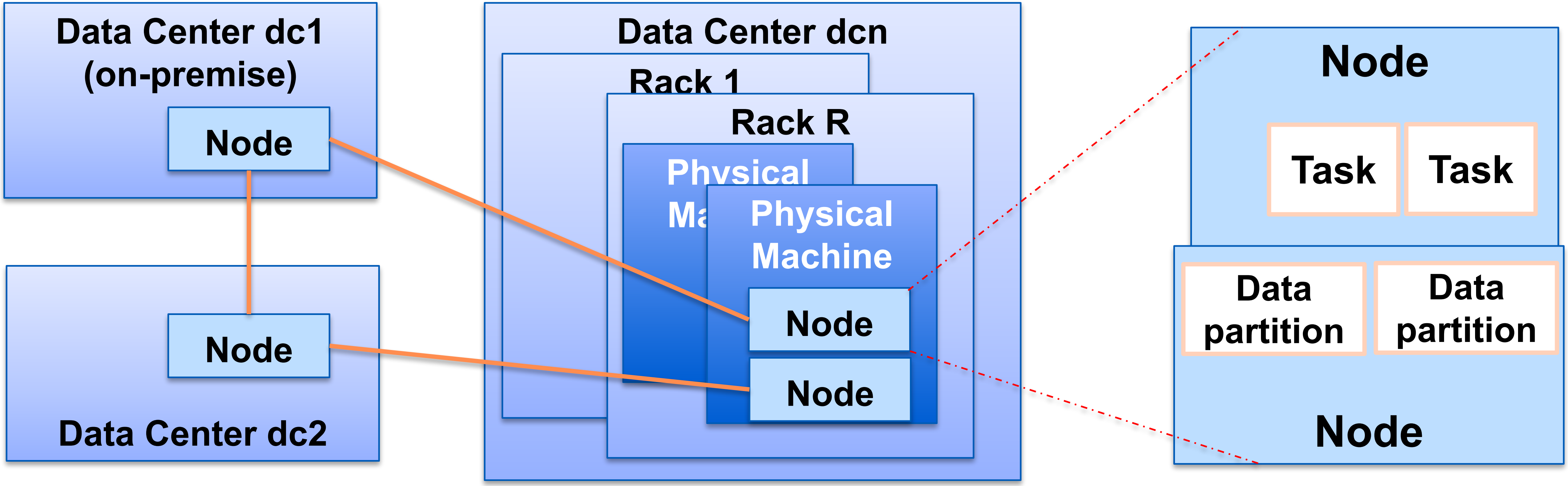
To architect the platform centered around big data and data intensive activities!

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 data contracts, service level objectives (SLO) and service level agreements (SLAs)
  - performance, service failure, cost, privacy/security …
  - data quality and data governance

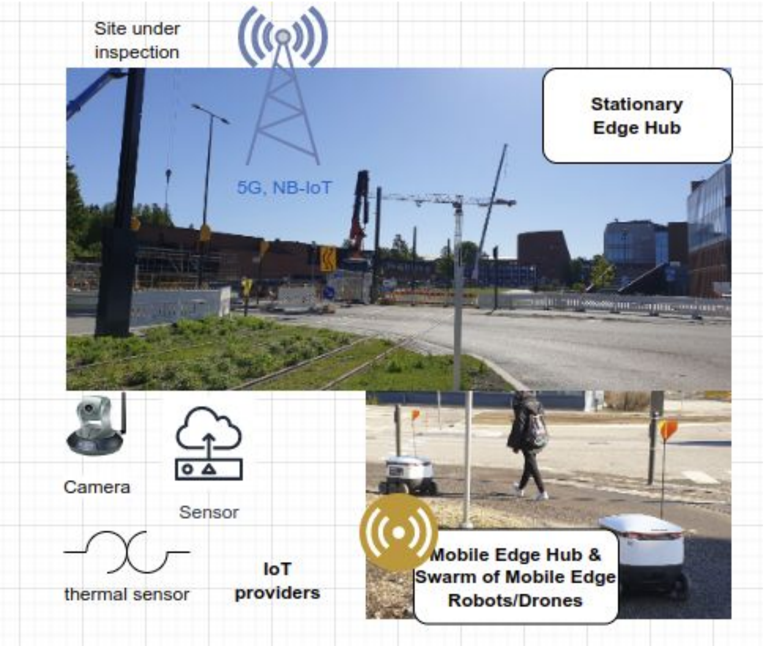
Understanding the underlying infrastructures for big data platforms

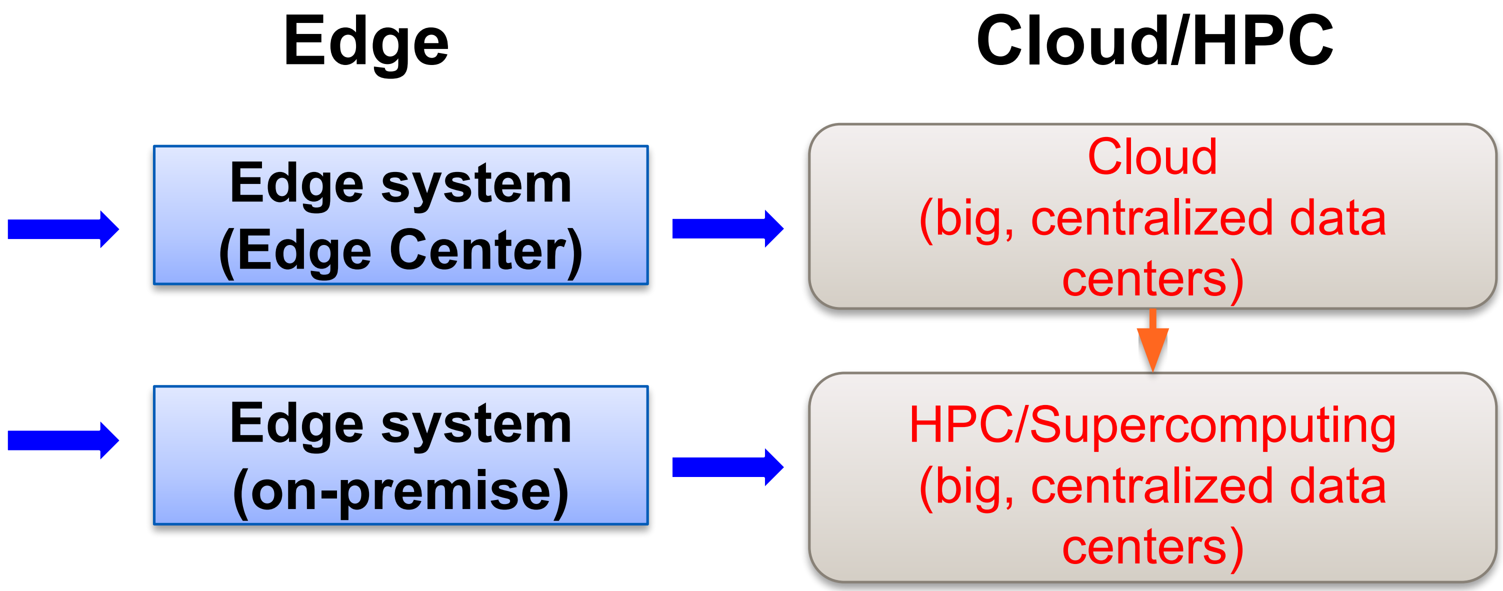
large-scale distributed infrastructures ⇒ hybrid and/or multi cloud/edge



Storage (NVMe/SSD, HDD), Interconnects (CXL, RoCE v2, InfiniBand) and Compute resources (CPU/GPU)

Emerging edge-cloud continuum affecting data platform designs: data storage and processing





Edge

Cloud/HPC

Edge system (Edge Center)

Edge system (on-premise)

Cloud (big, centralized data centers)

HPC/Supercomputing (big, centralized data centers)

close (physically)

far (physically)

"supercomputer in a box" like NVIDIA DGX Spark

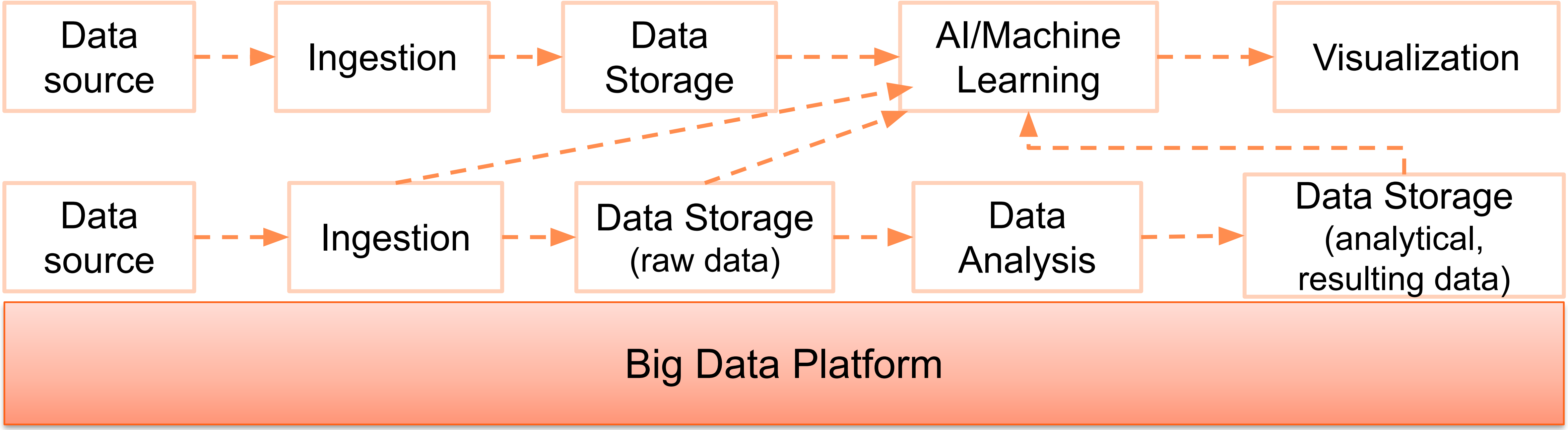
Data-centric development & operations

- 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
  - patterns/insights in data will be interpreted and presented for making decisions, reporting, and creating stories

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

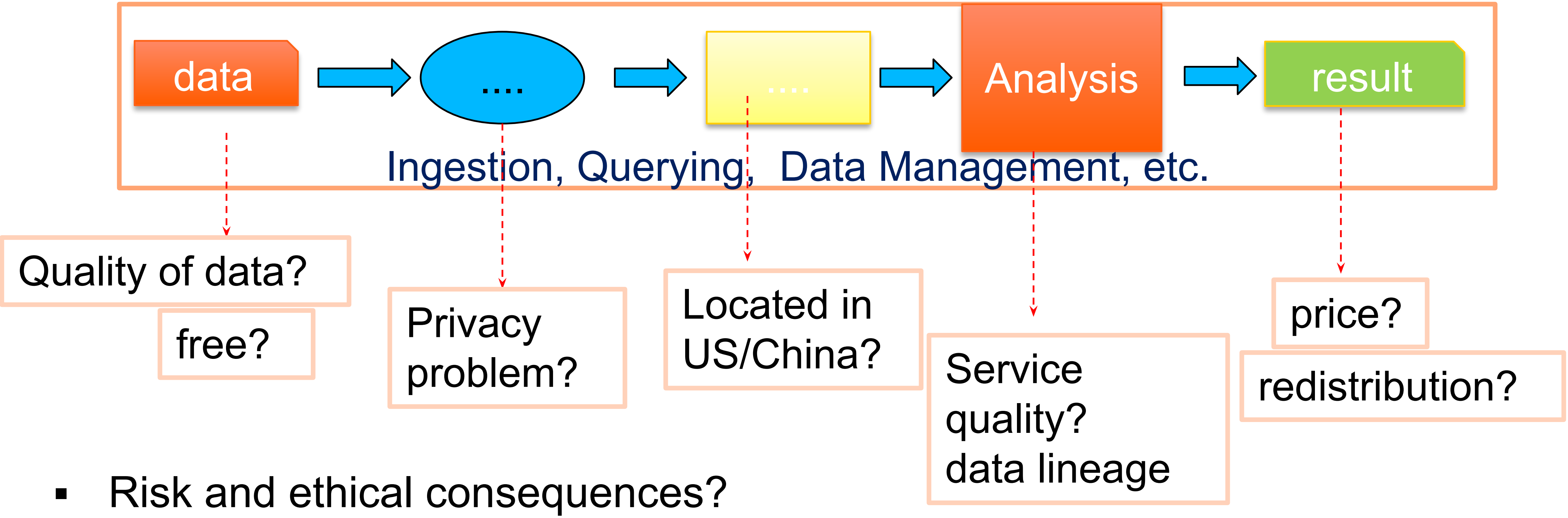
- Hardly to avoid the support for multiple types of data
  - tenant requirements and application needs, given data as “products”/”assets”
- Multiple types of data
  - different characteristics and values
- Any elastic solution that ensures minimum changes to support generalization and extensibility
  - 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 and 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

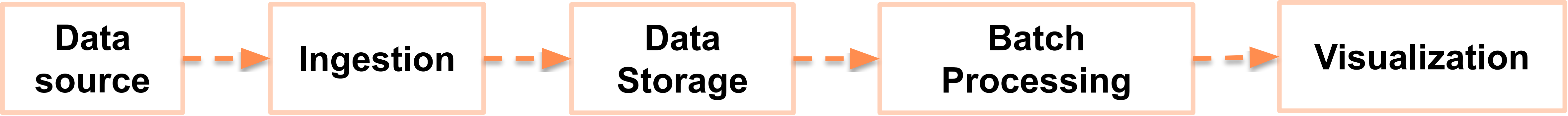
Data concerns: example



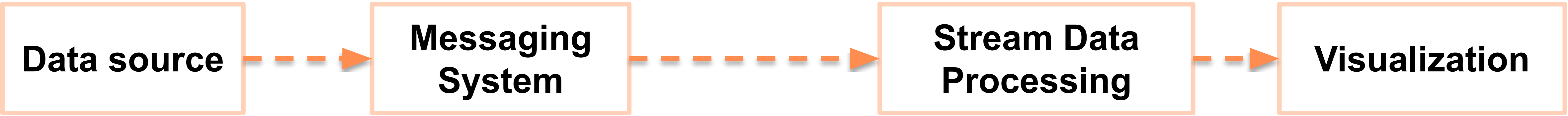
- Risk and ethical consequences?
- Regulation-compliant platforms: e.g., GDPR, AI Act

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

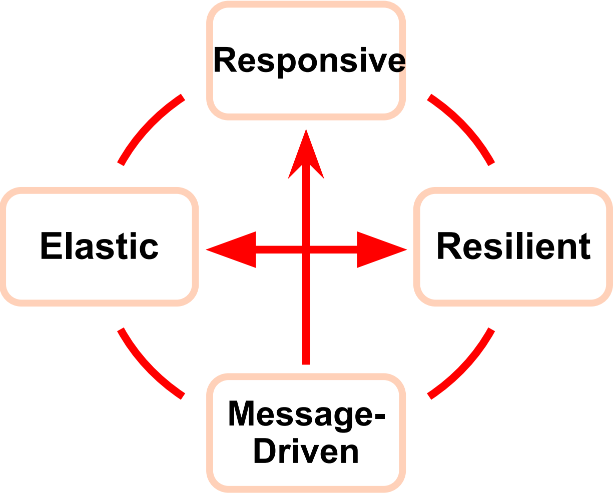
For dealing with V*

- 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 or resilience
- Sustainability within big data platforms
  - energy consumption, reusability, extensibility

- Reliability
- Resilience
- Elasticity

→

- *Sustainability*

→

Design and implementation

Sus. goal	Platform design	Concrete techniques	Verification

How to have them for

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

Partitioning and composability: splitting functionality & data

- Breakdown the complexity and responsibility
  - easy to implement, replace and compose
  - deal with performance, scalability, security, data quality, etc.
  - support teams in DevOps and data products
  - data responsibility and ownership
  - cope with technology changes
- Many things are related to the current trends
  - microservices, domain-driven designs, data mesh, data platform modernization (storage, query, and analysis/AI couplings)
- But tradeoffs with management w.r.t data resources

Example of functional and data partitioning and composability (1)

Service-oriented components

Microservices and domain-oriented microservices

Serverless functions/function as-a service

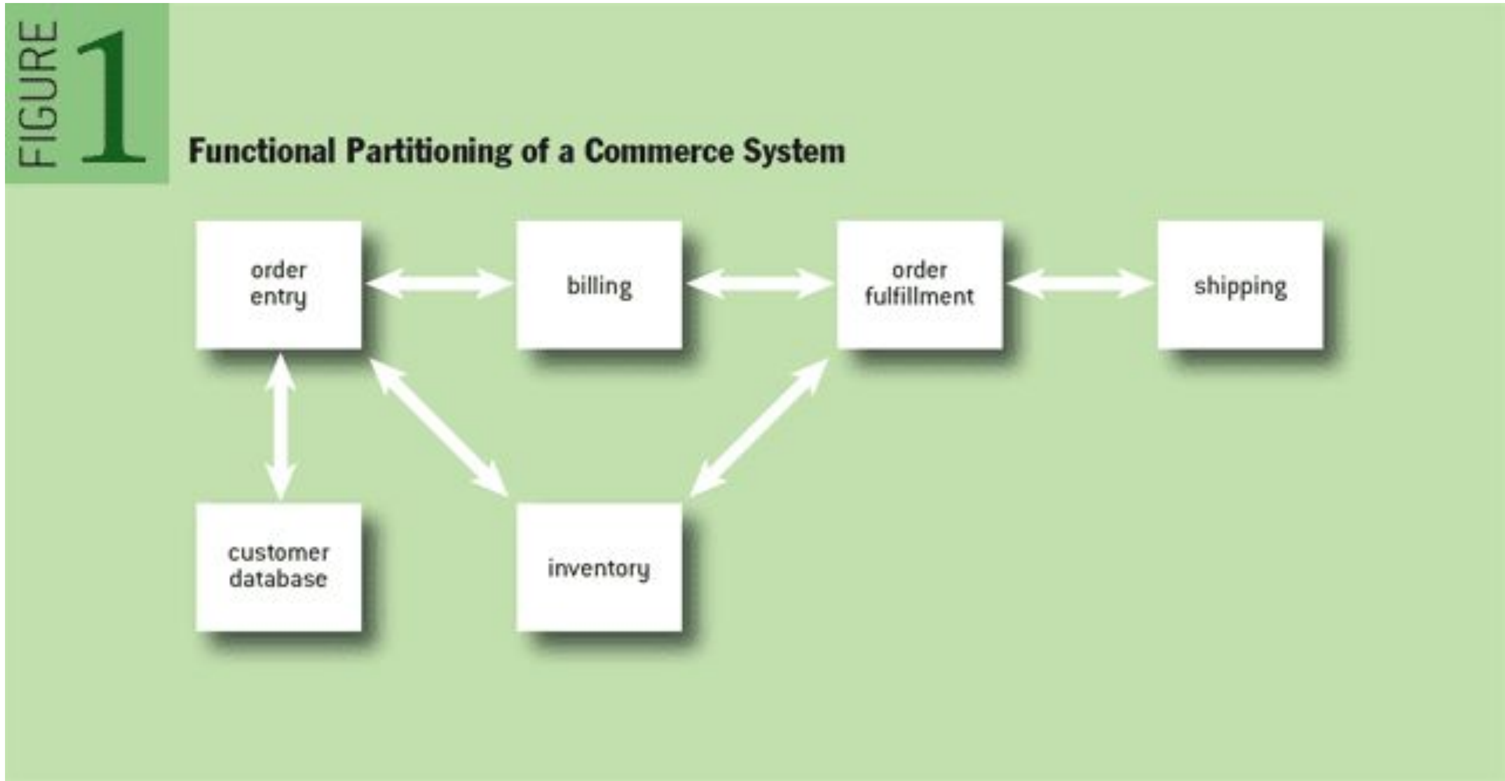


Figure source: *http://queue.acm.org/detail.cfm?id=1971597*

Example of functional and data partitioning and composability (2)

Multi (hardware) storage capabilities

Data sharding

Multi data spaces

Multi data services

Multiple data infrastructures

Data products

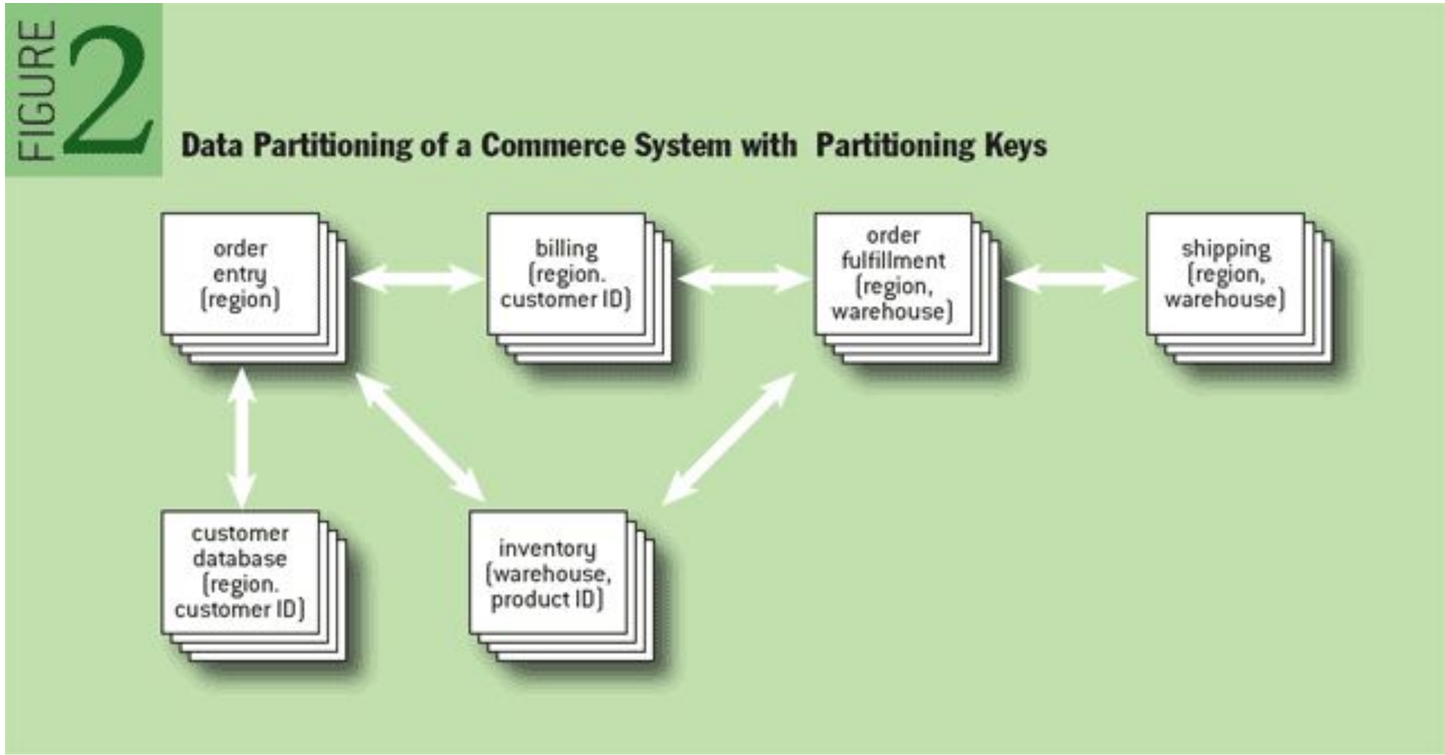
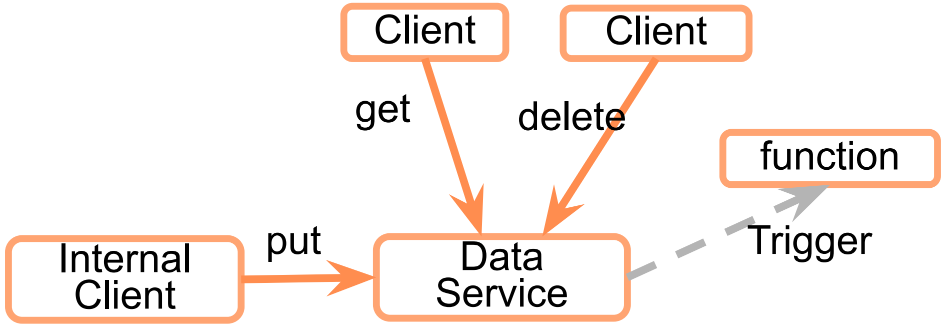


FIGURE 2
Data Partitioning of a Commerce System with Partitioning Keys
order entry (region)
billing (region. customer ID)
order fulfillment (region, warehouse)
shipping (region, warehouse)
customer database (region. customer ID)
inventory (warehouse, product ID)

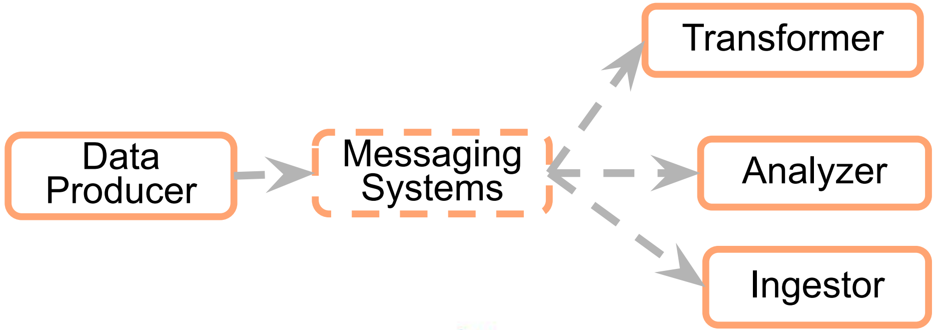
Figure source: *http://queue.acm.org/detail.cfm?id=1971597*

Interaction: multiple models

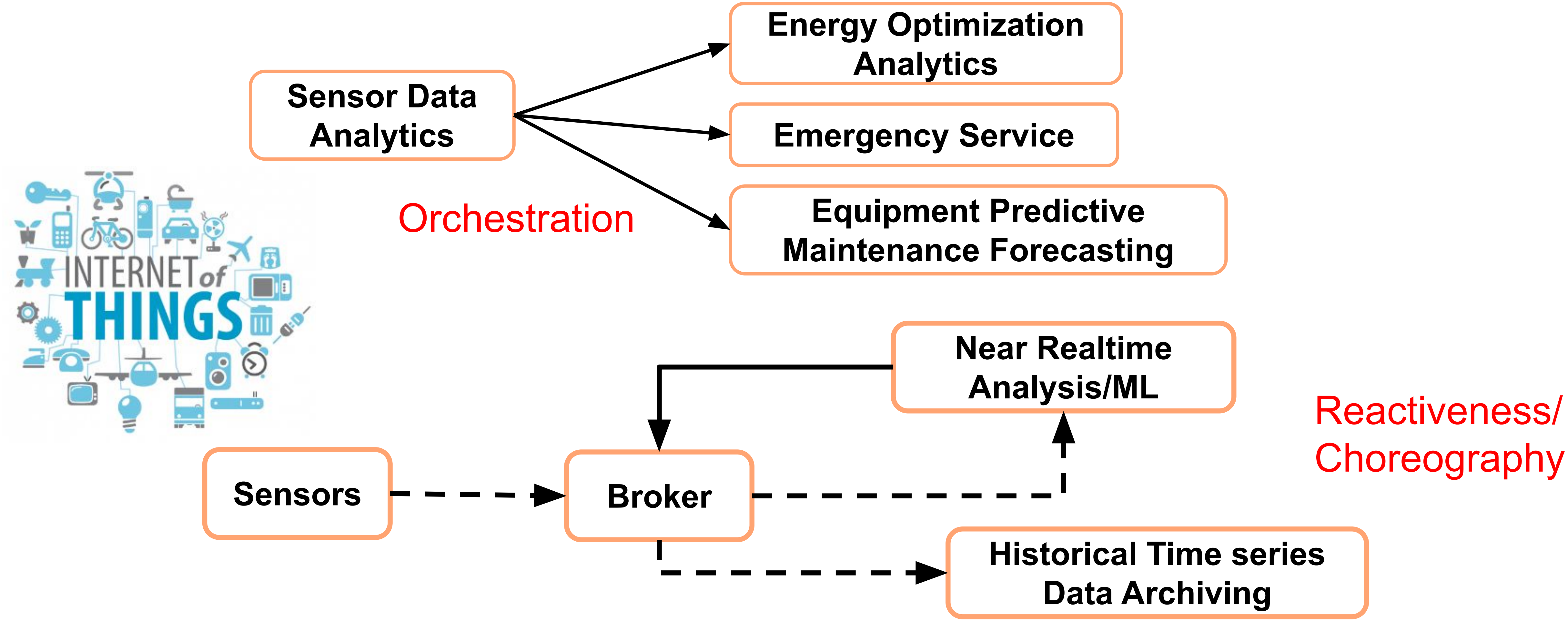
- Protocols
  - REST, gRPC, Message Passing, Stream-oriented Communication
- Models
  - one-to-many, many-to-one, many-to-many
  - synchronous/asynchronous calls
  - internal data exchange versus open/external exchange



Amazon S3/MongoDB



Coordination: Orchestration and Reactiveness/Choreography



Sensor Data Analytics

Energy Optimization Analytics

Emergency Service

Equipment Predictive Maintenance Forecasting

Orchestration

Near Realtime Analysis/ML

Reactiveness/ Choreography

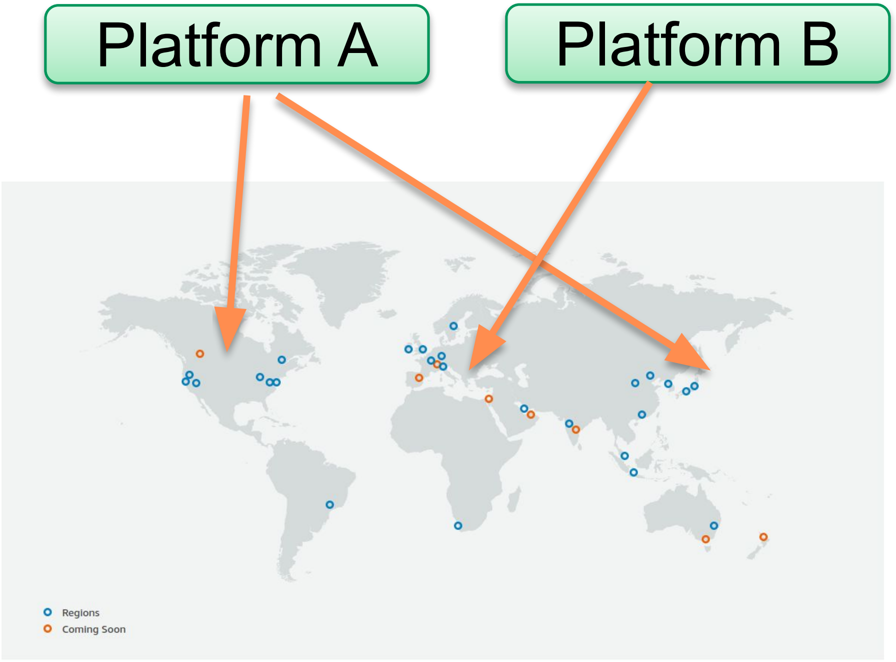
Sensors

Broker

Historical Time series Data Archiving

Distribution: Edge or Cloud Data Centers?

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



Platform A

Platform B

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/

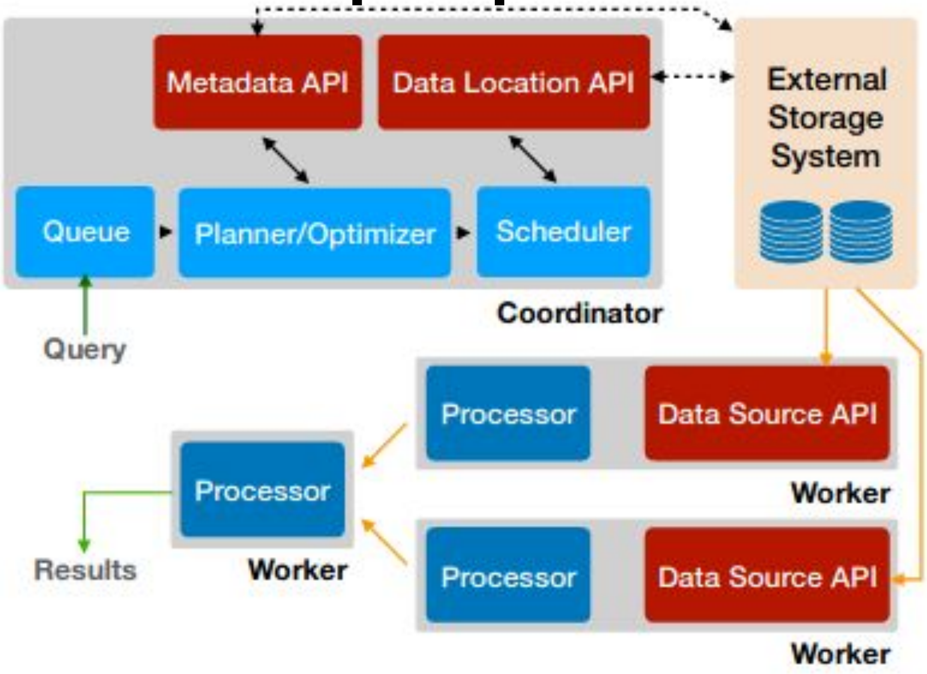


Figure source: *Presto: SQL on Everything https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8731547 &tag=1*

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"

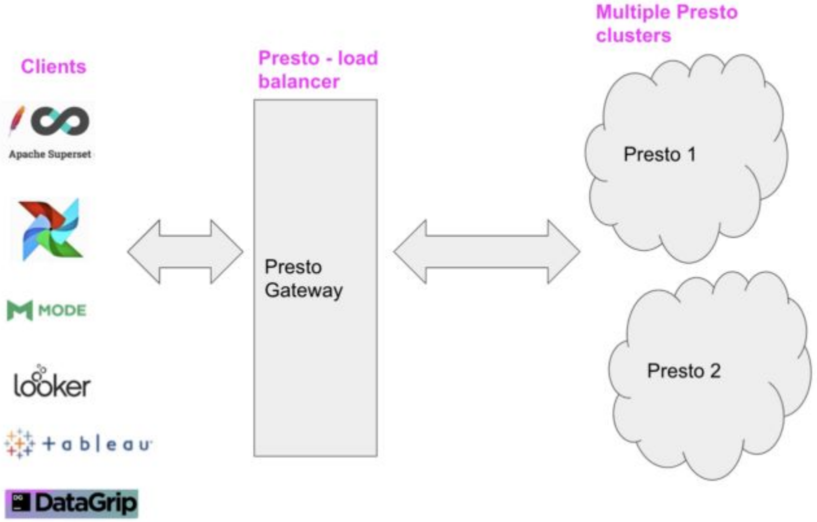
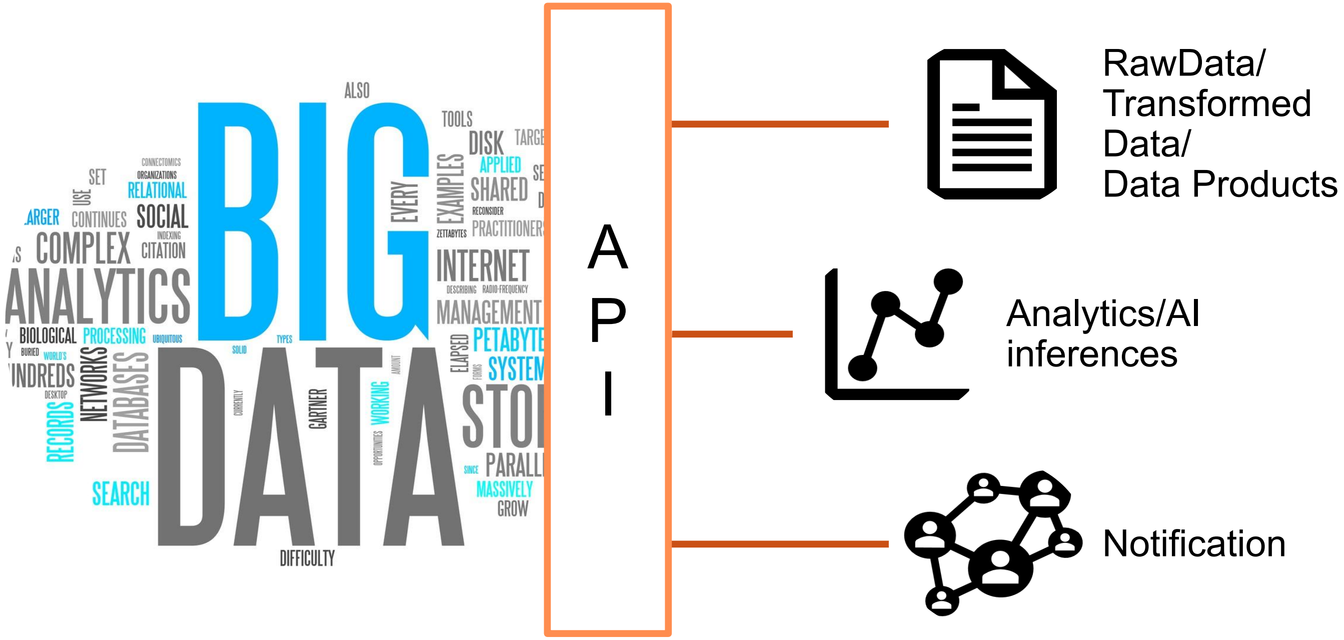


Figure and text source: *https://eng.lyft.com/presto-infrastructure-at-lyft-b10adb9db01*

API for Data Platform as a Service

- APIs are very important! Why?
  - enable consumers access to data and analytics 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 and governances



Delivery data as “asset”/”product”

- FAIR principles
  - Findable, Accessible, Interoperable and Reusable
- Product/asset characteristics

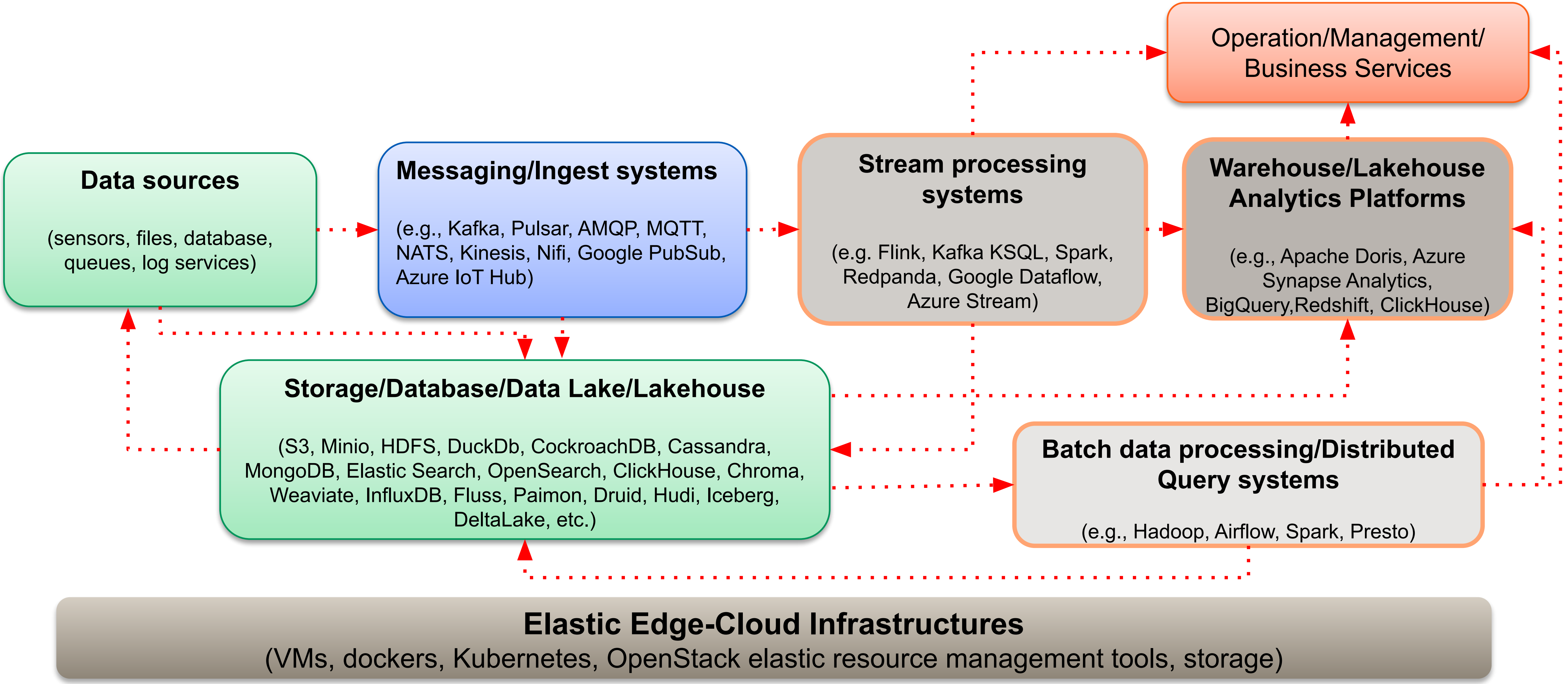
⇒ The platform must support

- Metadata and governance
- Service-level objectives and quality controls for data
- Search
- Access control policies

[FAIR principles]: https://www.nature.com/articles/sdata201618

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

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, Pulsar, AMQP, MQTT, NATS, Kinesis, Nifi, Google PubSub, Azure IoT Hub)

Stream processing systems

(e.g. Flink, Kafka KSQL, Spark, Redpanda, Google Dataflow, Azure Stream)

Warehouse/Lakehouse Analytics Platforms

(e.g., Apache Doris, Azure Synapse Analytics, BigQuery,Redshift, ClickHouse)

Storage/Database/Data Lake/Lakehouse

(S3, Minio, HDFS, DuckDb, CockroachDB, Cassandra, MongoDB, Elastic Search, OpenSearch, ClickHouse, Chroma, Weaviate, InfluxDB, Fluss, Paimon, Druid, Hudi, Iceberg, DeltaLake, etc.)

Batch data processing/Distributed Query systems

(e.g., Hadoop, Airflow, Spark, Presto)

Elastic Edge-Cloud Infrastructures

(VMs, dockers, Kubernetes, OpenStack elastic resource management tools, storage)

Thanks!

Hong-Linh Truong
Department of Computer Science

rdsea.github.io

Kiitos
aalto.fi