Aalto University
School of Science

# Architecting Big Data Platforms

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

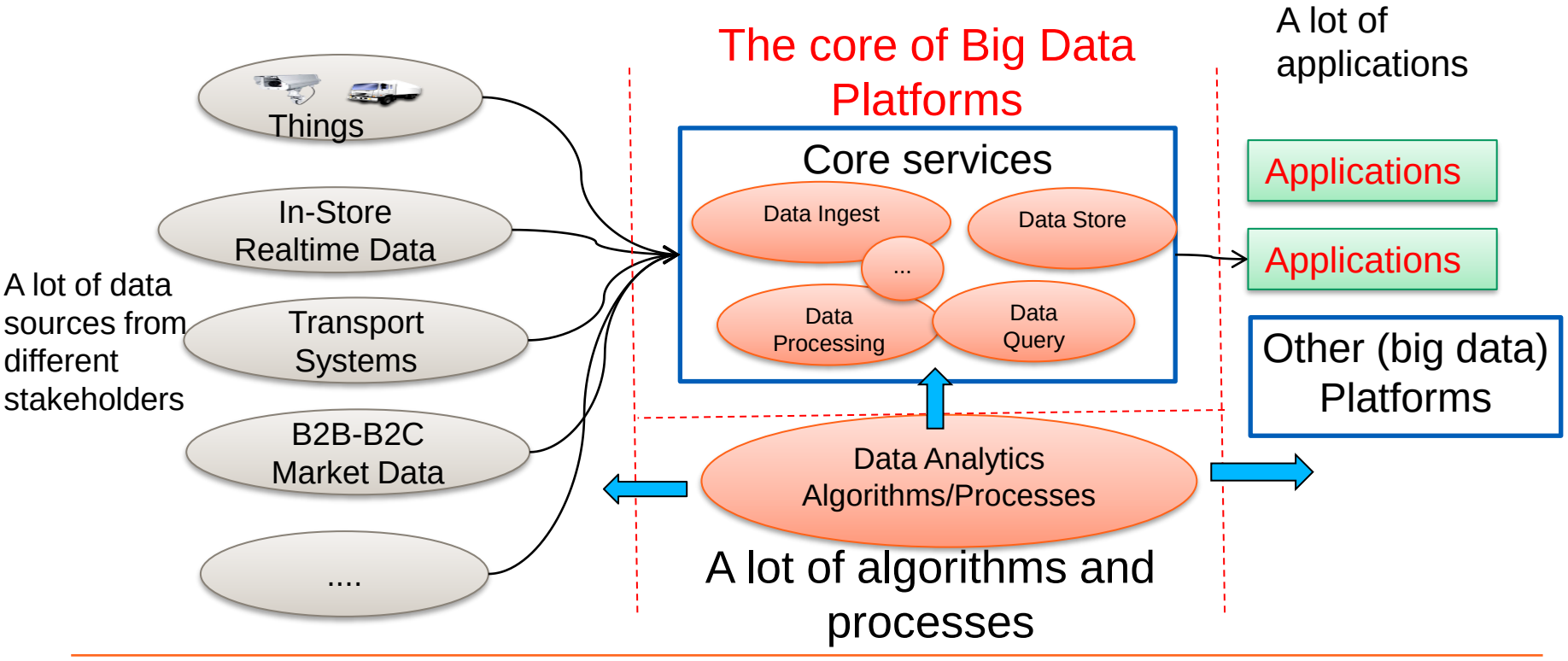
# Learning objectives

- **Understand key issues in designing a big data platform**
- **Study different big data architectures**
- **Learn key architecture design issues**
  - interaction, partitioning, elasticity, API
- **Understand 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 …"

# Recall - Big data platforms: system of systems view



# 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 do self-manage and which part will be fully managed  by other providers?
- How to design elastic big data infrastructures?
- Etc.

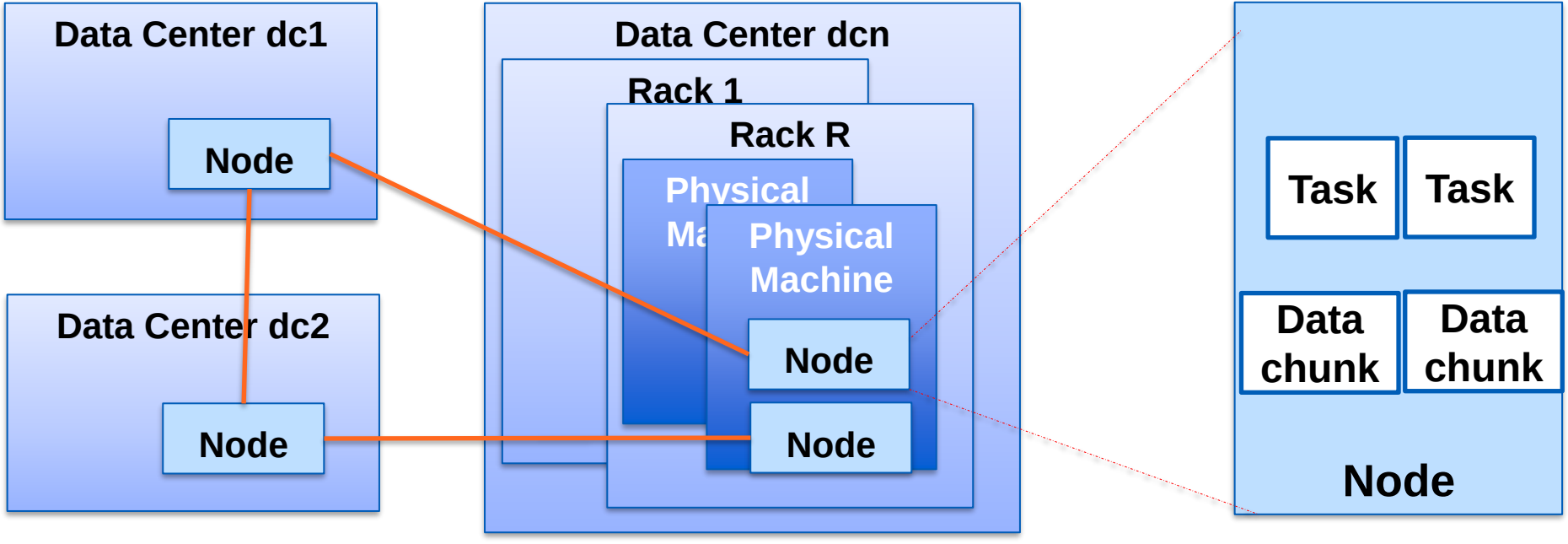
# Your Big Data Platform story starts with Big Data Platform architectures!

**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 computing infrastructures



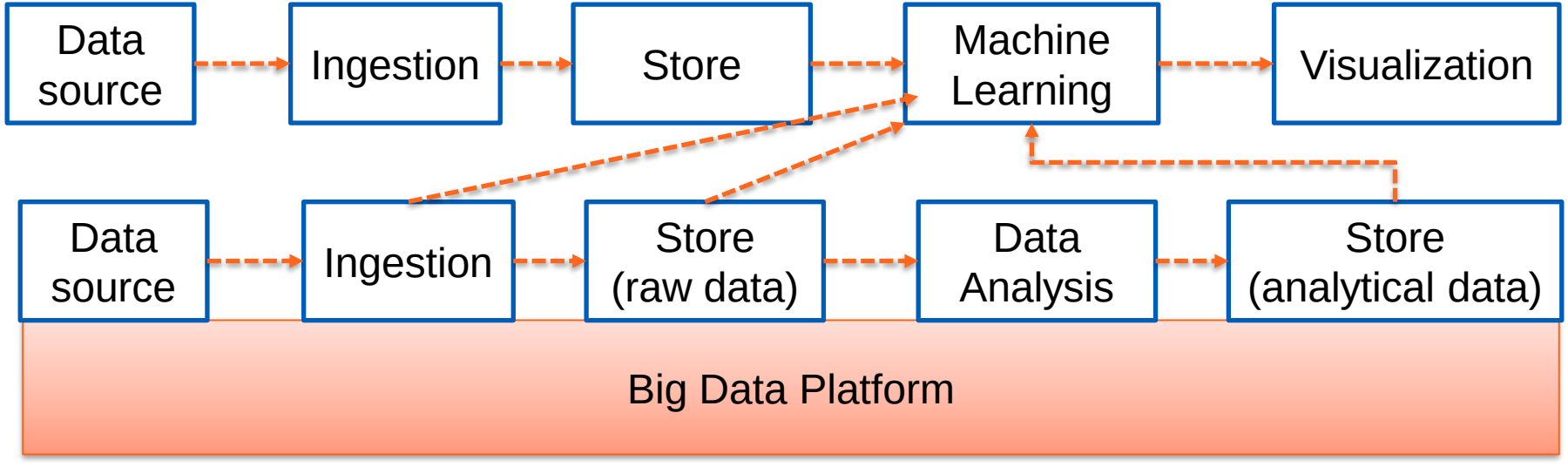

**Remember: large-scale distributed infrastructures!**

# Data-centric development & operations

- **Data ingestion and data movement**
  - 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 analyzing and (Machine) learning**
  - data within platforms will be processed, analyzed and learned to improve data, find insights and to create models
- **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)**



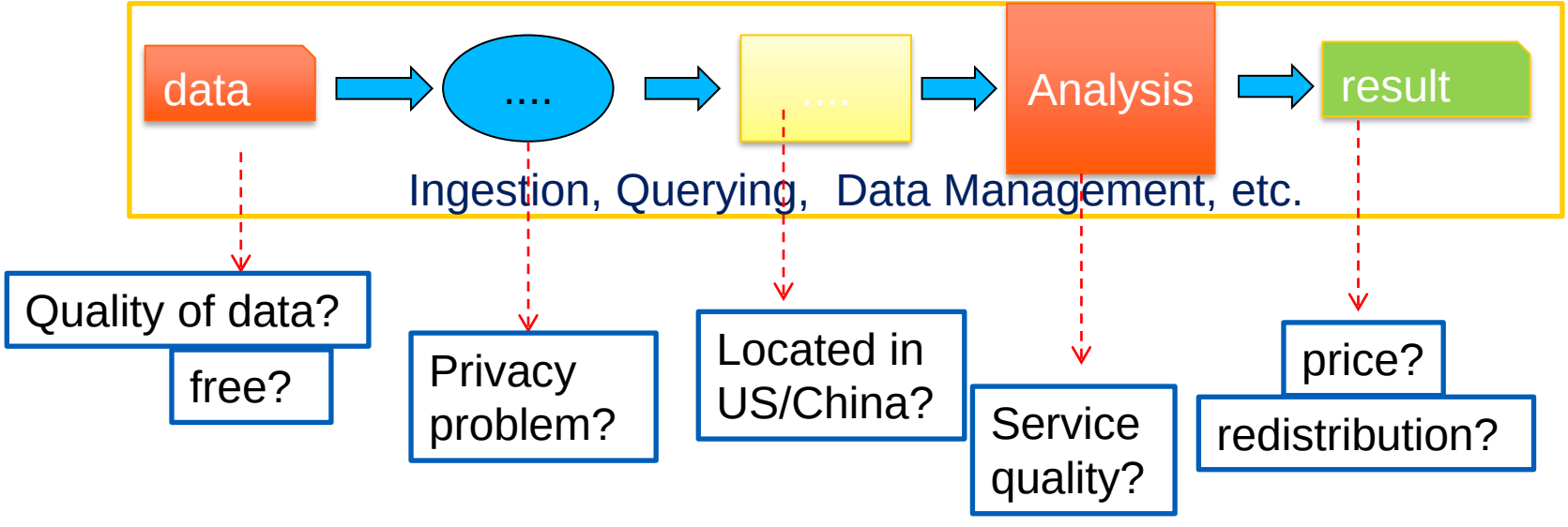
# Handling multiple types of data?

- **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.
- **SLA multitenancy versus single tenancy**
  - security, privacy, performance, reliability and maintenance?

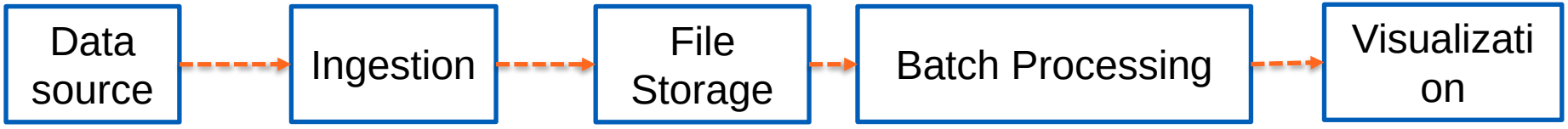
# Data concerns: data validation and quality of analytics



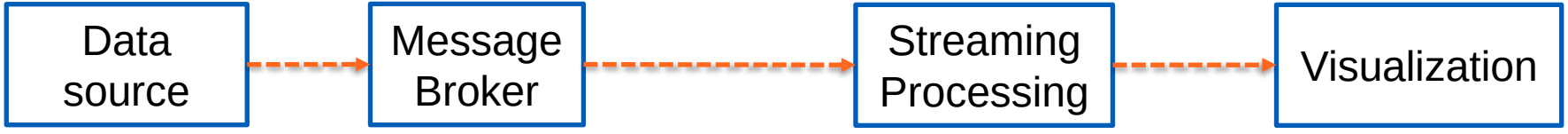

- Ethical consequences?
- Regulation-compliant platforms: e.g., GDPR

# Fast versus slow 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**



**But**

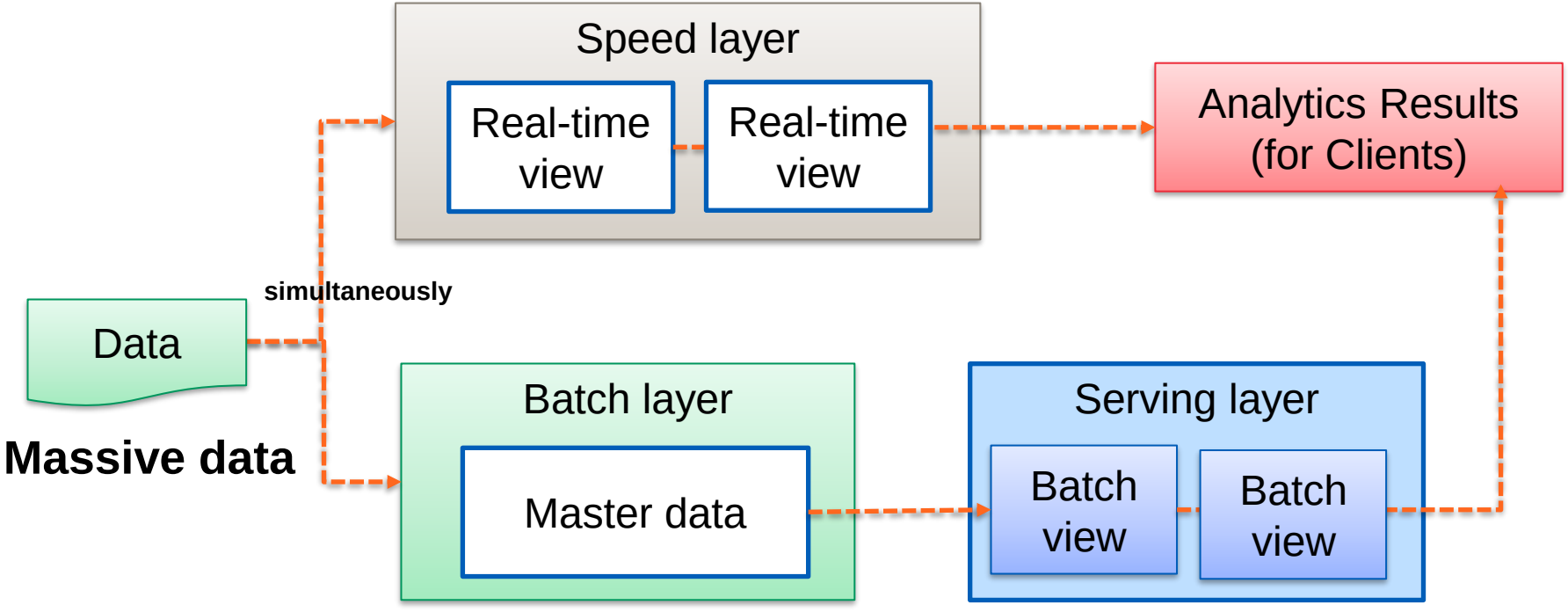
**if you have mixed types of data**

**Or**

**if you have big data you want to do analytics with different quality of analytics (cost, performance, quality of data)?**

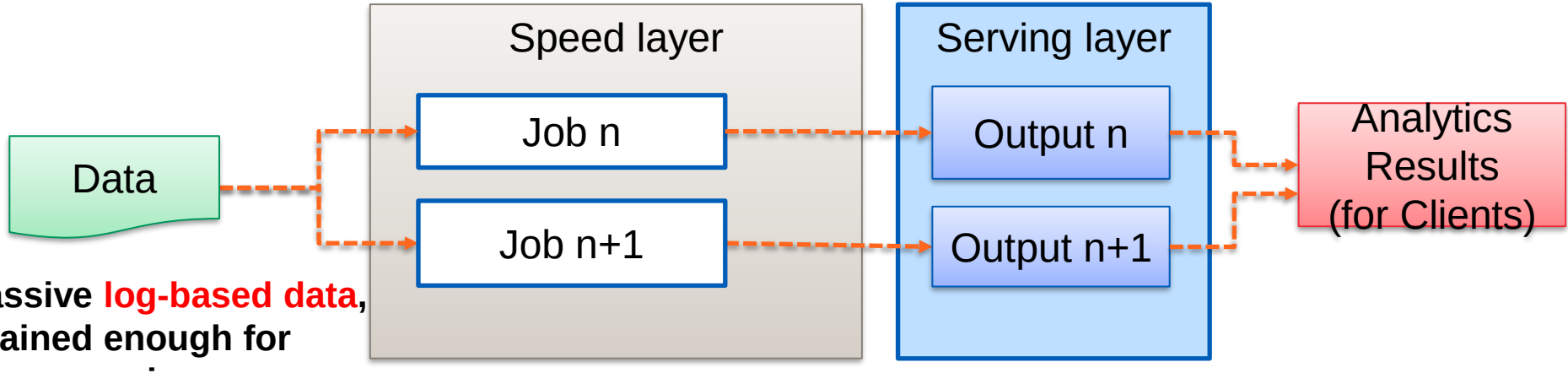
**Then ?**

# Lambda architectural style




**Check: http://lambda-architecture.net/**

# Kappa architectural style




**Check: https://milinda.pathirage.org/kappa-architecture.com/**

**The set of big data tools/frameworks (and configurations) used is dependent on the big data architecture**

**be aware of your personal techradar!**

# Quick check

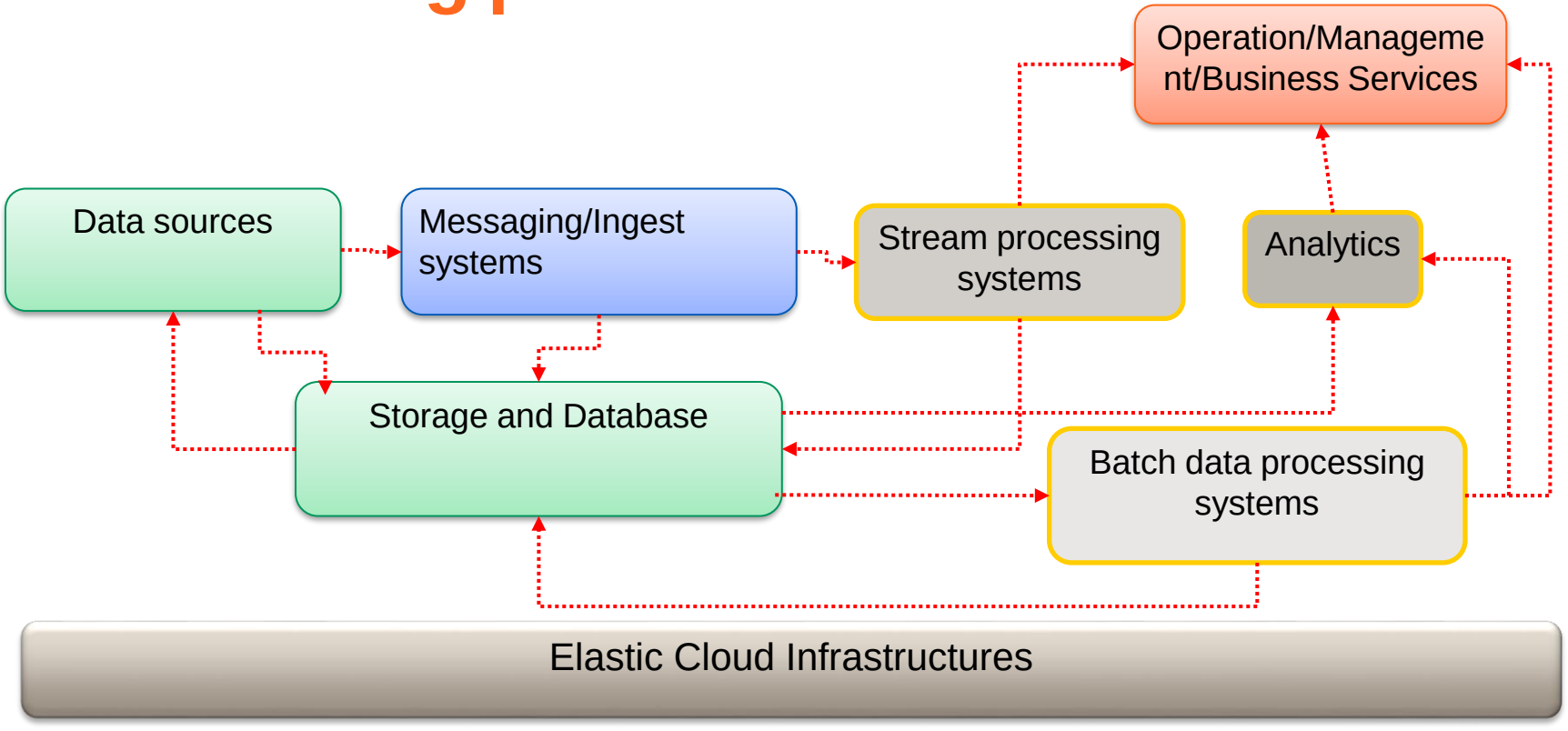
**“A big data platform monitors network usage of devices from million+ customers. We have different levels: Sensor/Customer, Node (concentrator of multiple customers), Agent (concentrator of multiple Nodes) and the whole network. In a region, the real operator can generate 1.4 billion records per day ~ 72GB per day”**

# Quickcheck

**First: breakout room discussion and then vote your choice**

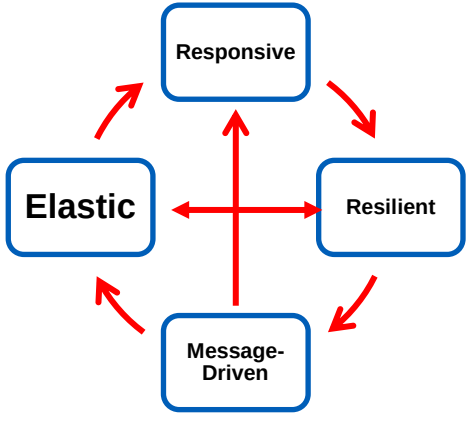
**https://presemo.aalto.fi/bdp**

# Basic building blocks for big data at large-scale: the big picture in this course



# How to architect big data platforms and pipelines as reactive systems?

## Reactive systems




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

**Why? For dealing with V***

- **Responsive: quality of services**
- **Resilient: deal within failures**
- **Elastic: deal with different workload and quality of analytics**
- **Message-driven: allow loosely coupling, isolation, asynchronous among many components**

# 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 of services with messages, non-blocking protocols, location-independent

# Partitioning: splitting functionality & data

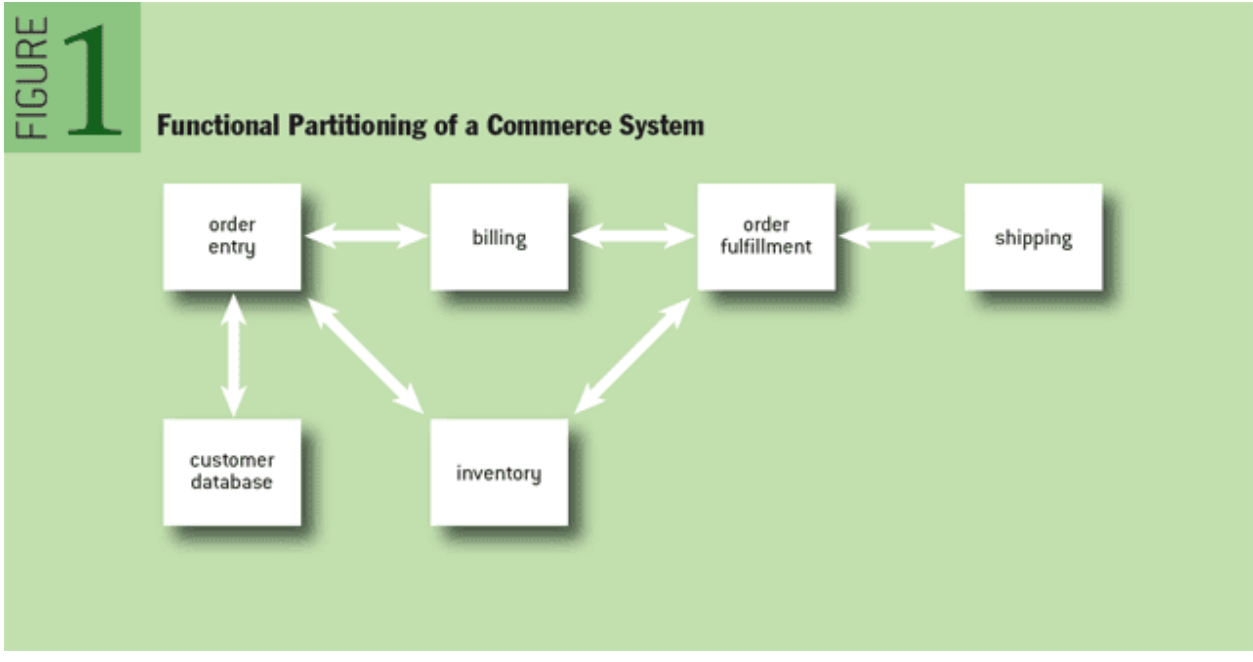
- breakdown the complexity
- easy to implement, replace and compose
- deal with performance, scalability, security, etc.
- support teams in DevOps
- cope with technology changes

# Example of functional and data partitioning

**Service-oriented components**

**Microservices**

**Serverless functions**



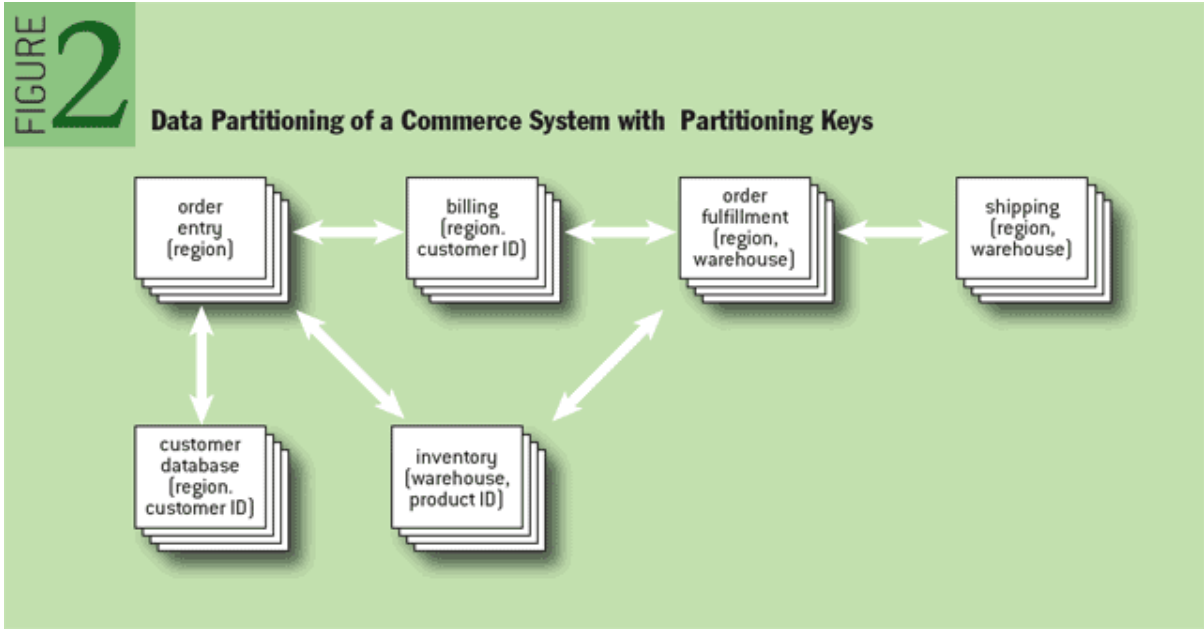

Figures source: http://queue.acm.org/detail.cfm?id=1971597

# Example of functional and data partitioning

**Data sharding**

**Multi data spaces**

**Multi data services**




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

Aalto University
School of Science

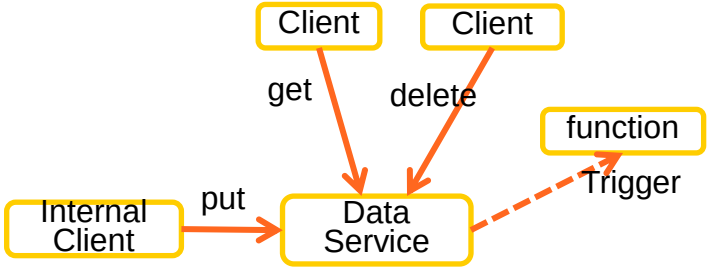
# Distributed systems of components are used to manage, ingest data and process data

# Interaction: protocols & interfaces

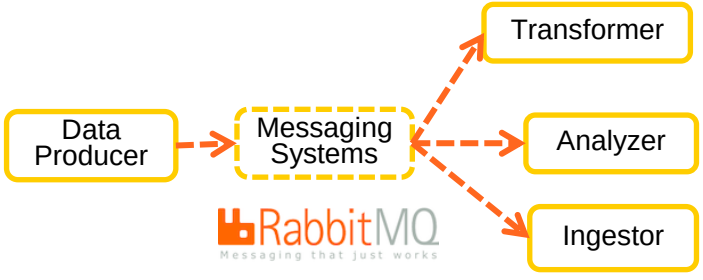
- **Large number of communication protocols and interfaces**
- **Interaction styles, protocols and interfaces**
  - REST, gRPC, Message Passing, Stream-oriented Communication
  - your own protocols
- **Other criteria**
  - architectural styles: microservices/serverless
  - scalability, Elasticity, Performance, Monitoring, Logging, etc.

# Interaction: Complex interactions

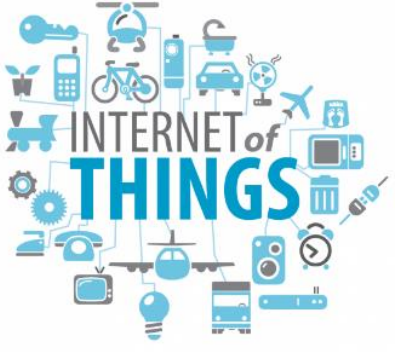
- One-to-many, many-to-one, many-to-many
- Synchronous/asynchronous calls
- Public/Subscribe, Message-oriented Middleware
- Internal data exchange versus open/external exchange

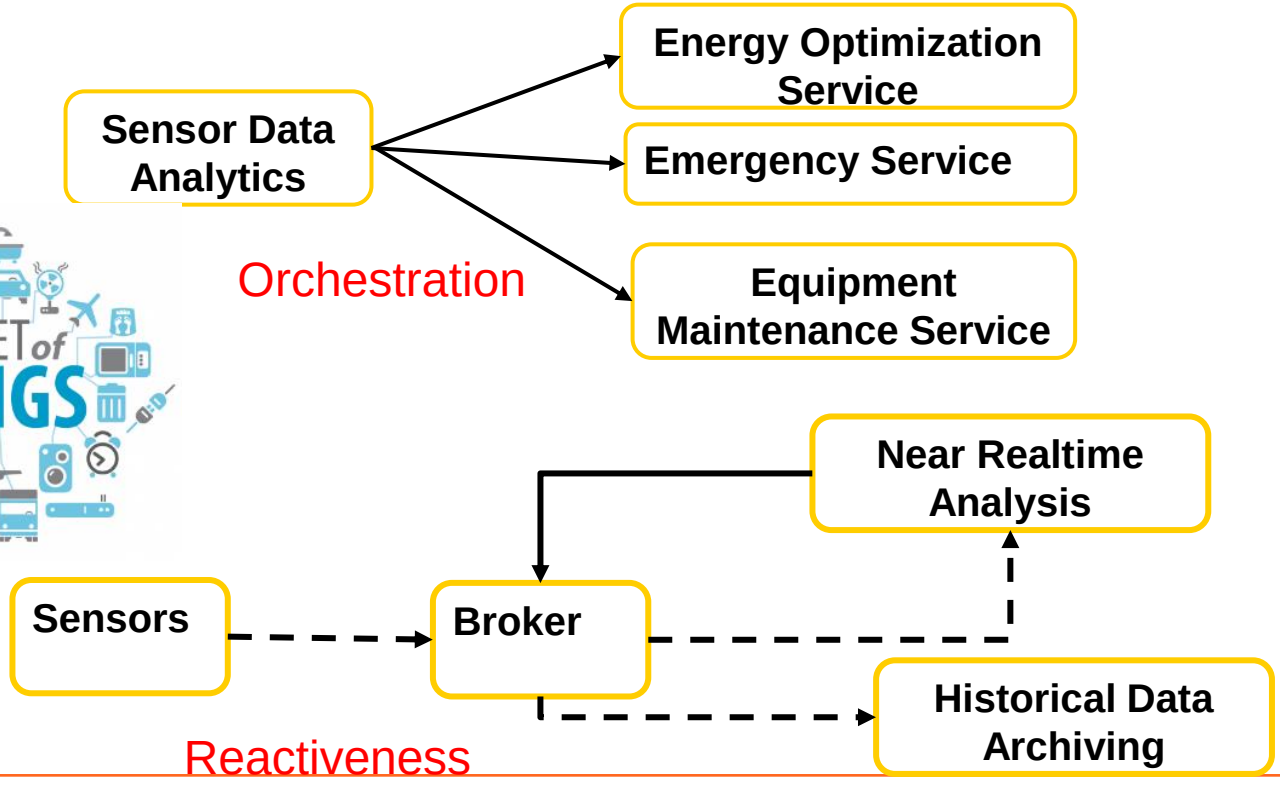



Amazon S3/MongoDB



# 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?**

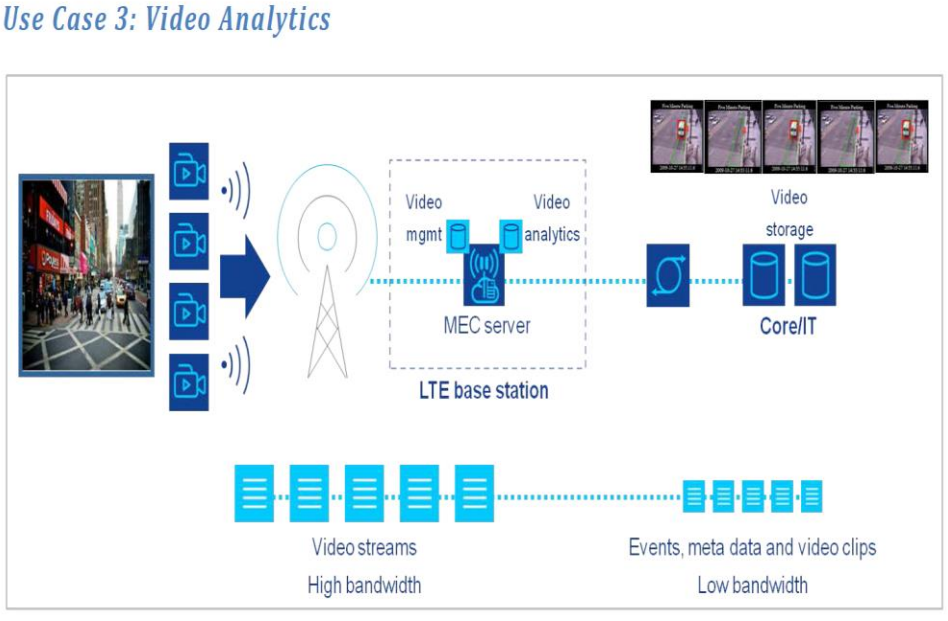



Figure 4: Example of video analytics

Figure source: https://portal.etsi.org/portals/0/tbpages/mec/docs/mobile-edge_computing_-_introductory_technical_white_paper_v1%2018-09-14.pdf

# Quick check

**“A big data platform monitors network usage of devices from million+ customers. We have different levels: Sensor/Customer, Node (concentrator of multiple customers), Agent (concentrator of multiple Nodes) and the whole network. In a region, the real operator can generate 1.4 billion records per day ~ 72GB per day”**

# Quick check

**https://presemo.aalto.fi/bdp**

# Scalability and elasticity: scale out

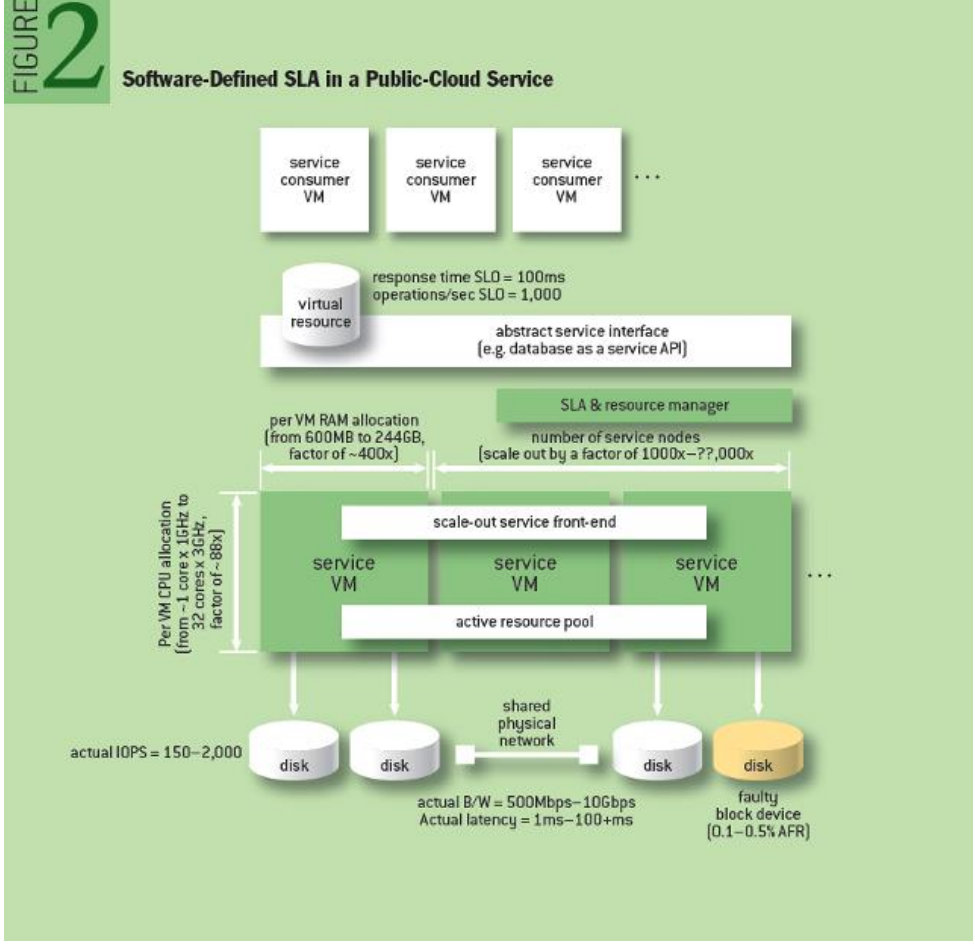



Figure source: http://queue.acm.org/detail.cfm?id=2560948

# Scalability and elasticity: load balancing

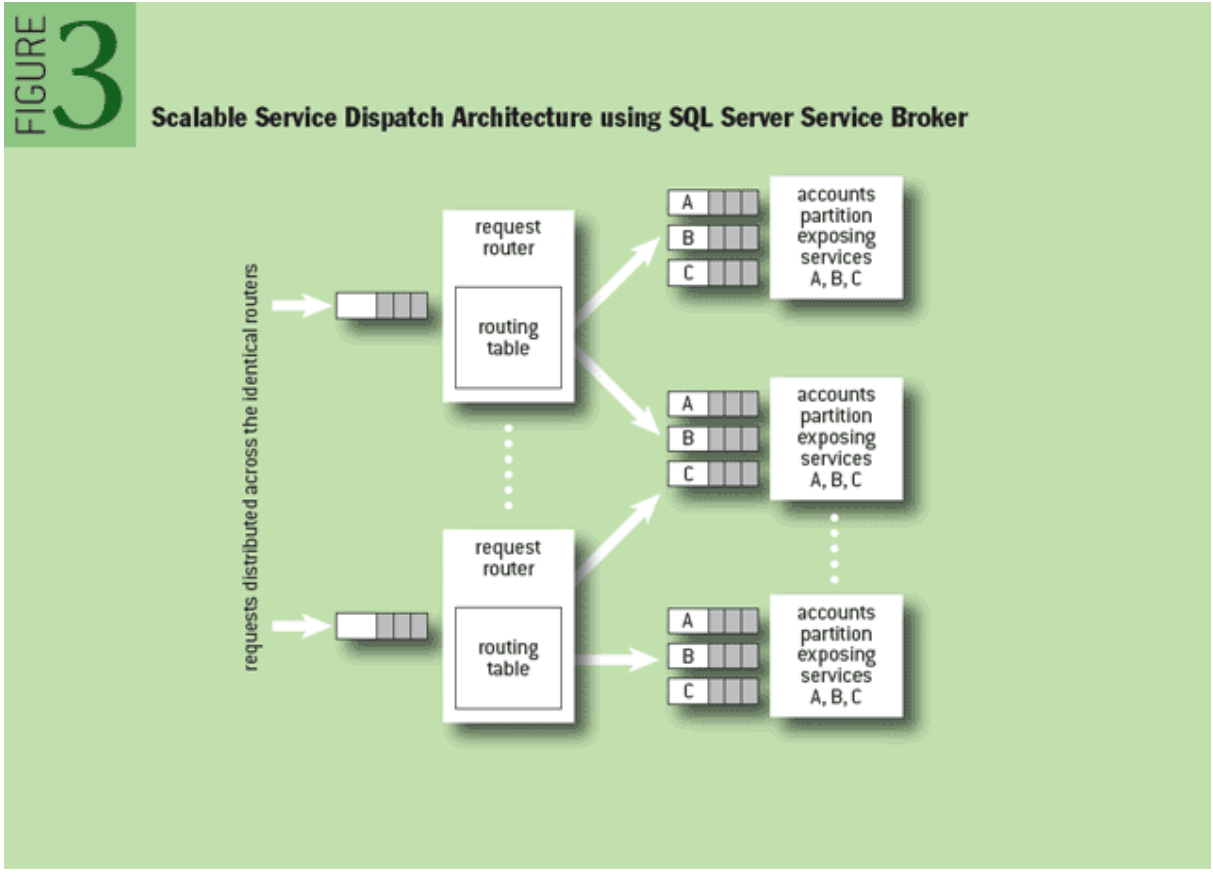
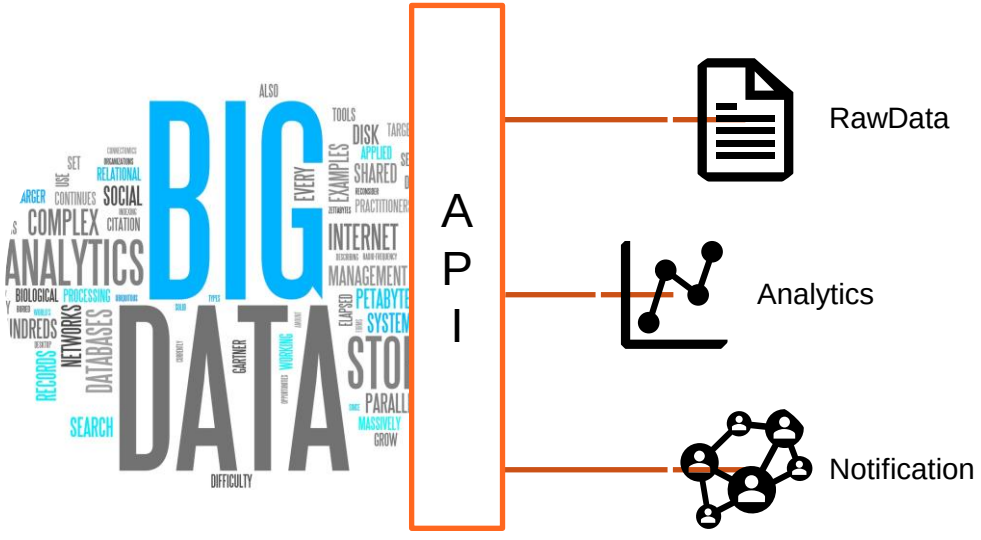



Figure source: http://queue.acm.org/detail.cfm?id=1971597

# API for Platform as a Service

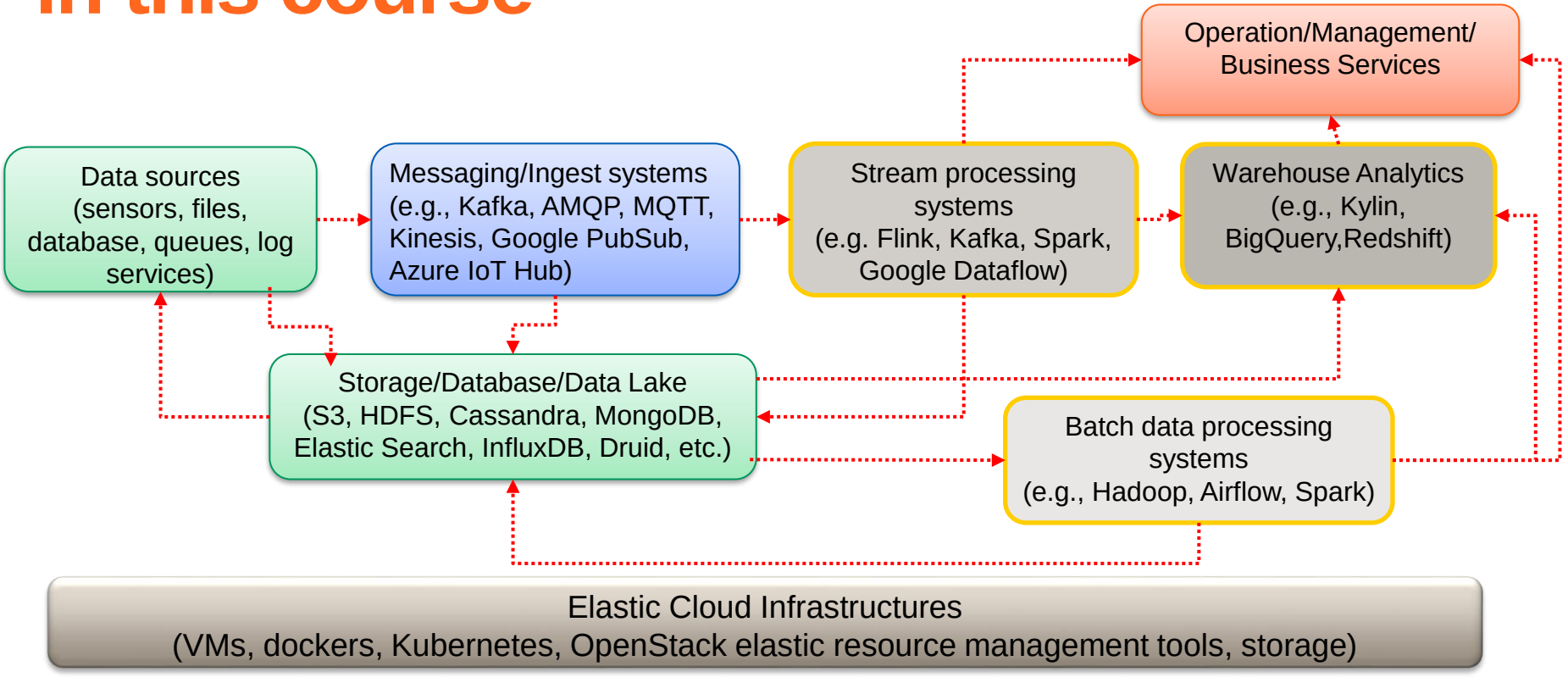
- **APIs are key! Why?**
  - enable customers access to data and analysis functions from your big data platforms without worrying about changes within your platforms
  - virtualization and management (hide internal, control access, throttling)




**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 tool 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**