DSLab

The Data Science Lab

Start your engines!

- Head over to the course webpage
  - https://dslab2019.github.io/

- If you want to try the big data platform at home (with small data)
  - Download, configure & start the HDP (not HDF) Sandbox
    - *https://hortonworks.com/downloads/#sandbox (registration required)*
    - *You can choose between the VM for VirtualBox or Docker*

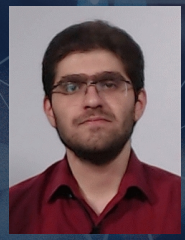
Final project

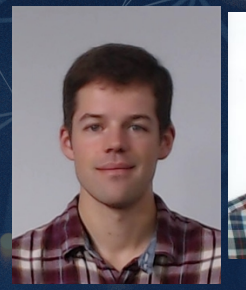
- **Description**: https://dslab2019.github.io/final_project/

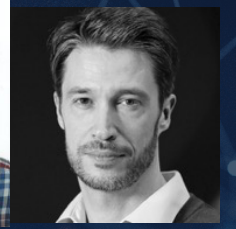
- Any questions?

Dealing with big data

Eric







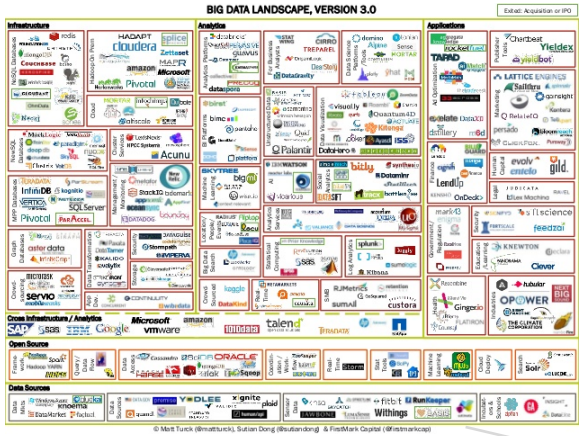
Tao, Ramtin, Mark, Apostolos, Dorina, Christine & Olivier

Spring 2019 - **week #4**

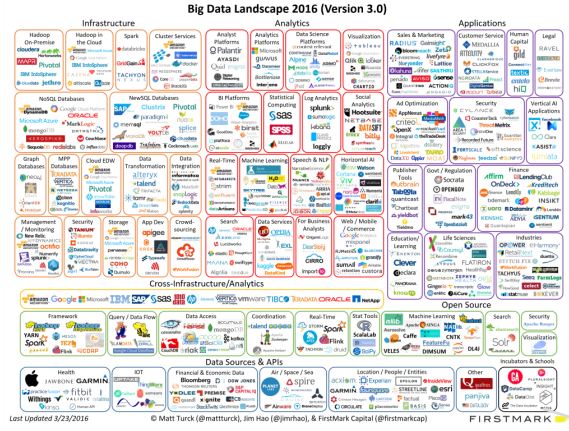
This Module

- Review Big Data concepts
- Experiment with Big Data technologies
  - Distributed storage
  - Distributed computing
  - Getting familiar with SQL on big data

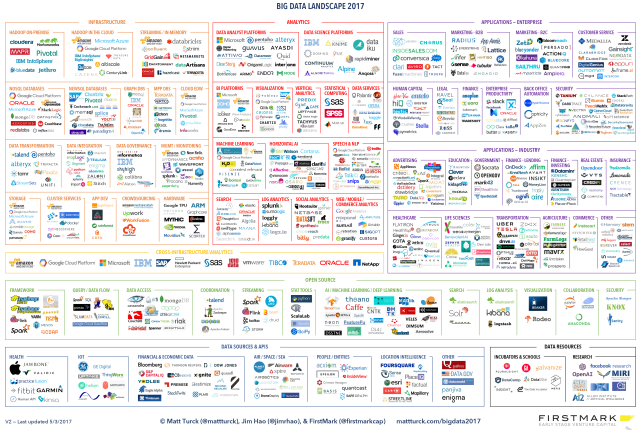
Big Data Landscape Evolution – A Moving Target



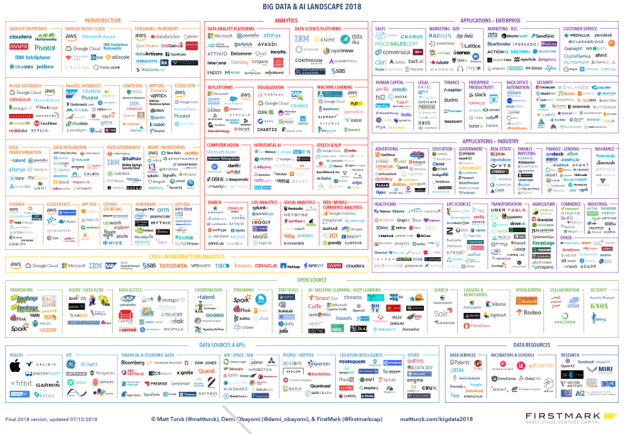
2014



2016



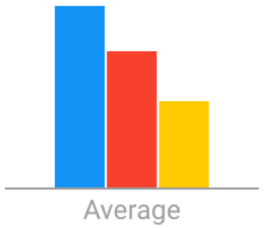
2017

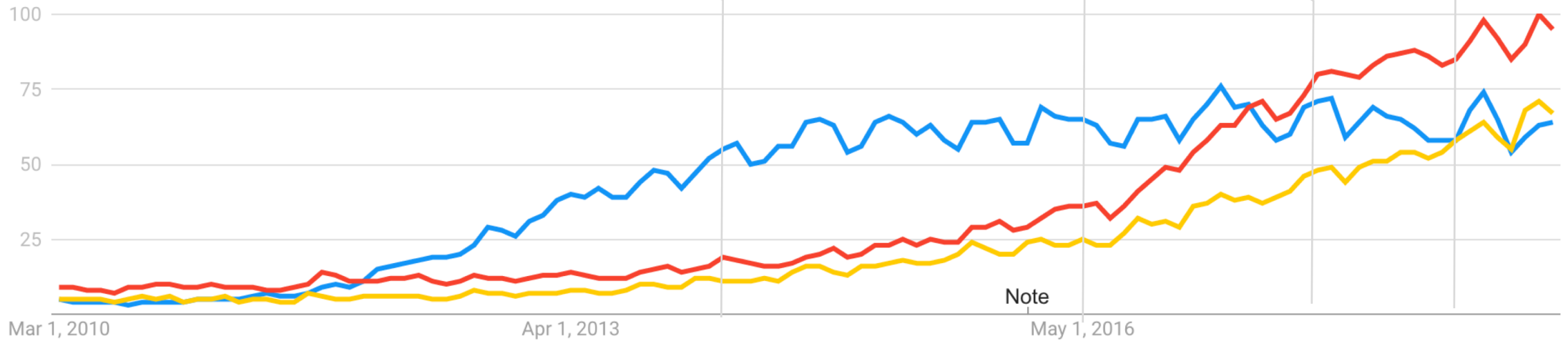


2018

Google Trends

Big Data
Machine Learning
Data Science

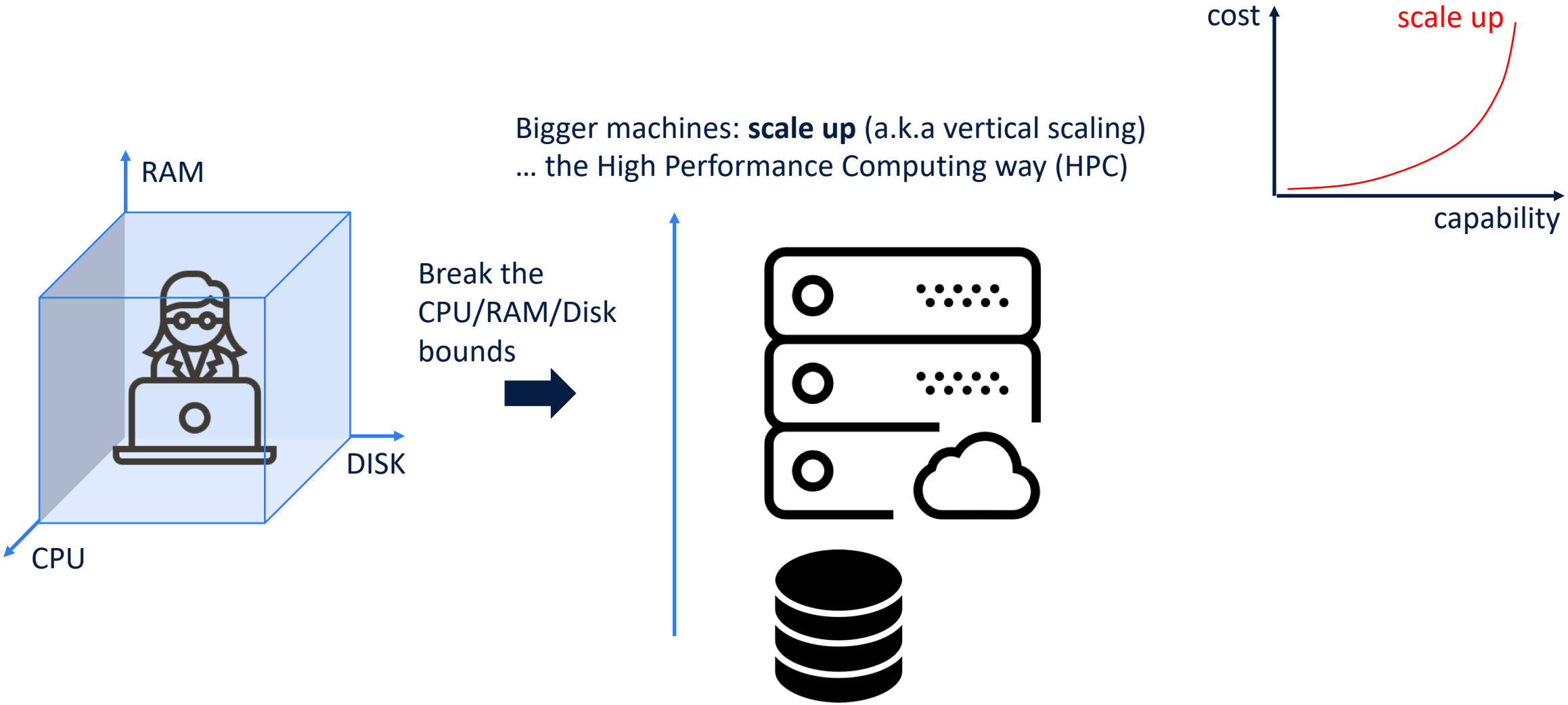




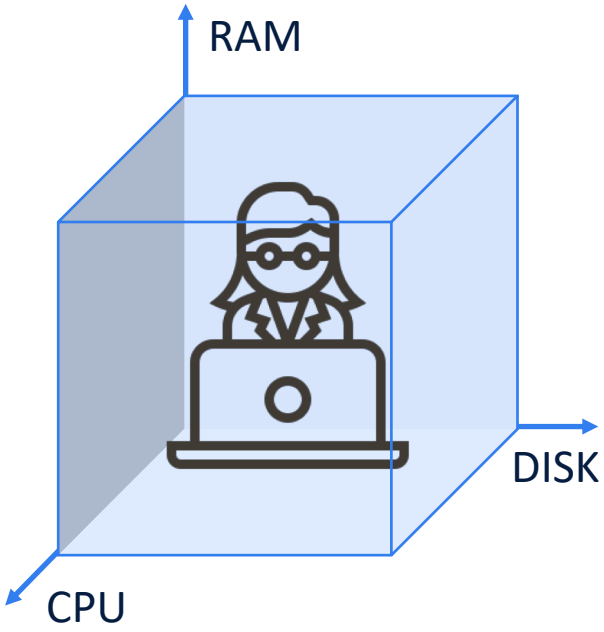
Today's Objectives

- Get acquainted with the Big Data ecosystems
  - Find your way in the Big Data jungle, explore it more efficiently

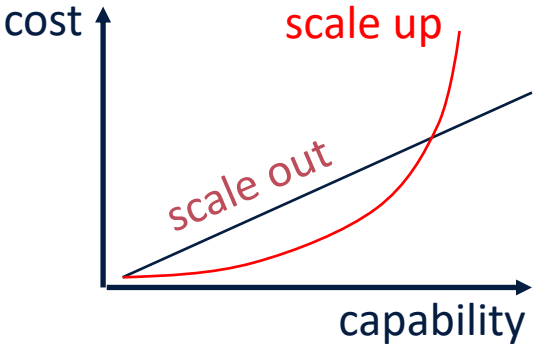
Addressing the Big Data Challenges

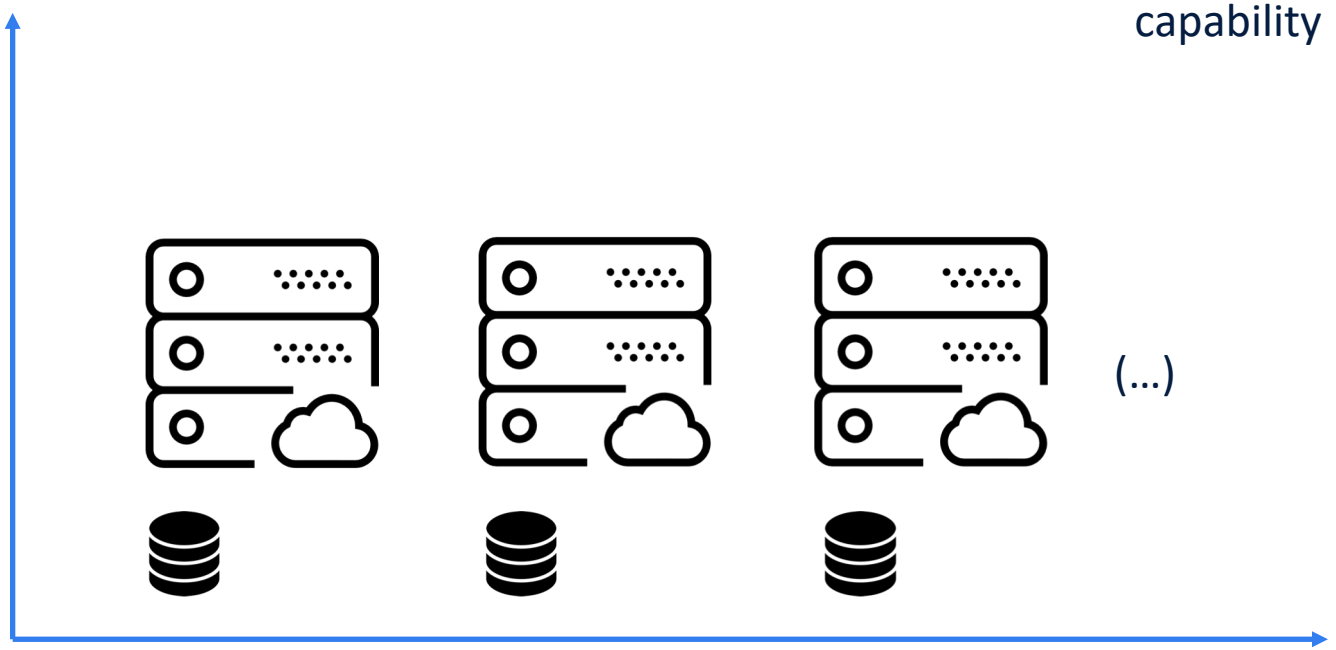


Addressing the Big Data Challenges



Break the CPU/RAM/Disk bounds





More machines: **scale out** (a.k.a horizontal scaling)
… the commodity hardware (cloud) way

Addressing the Big Data Challenge

- Horizontal scaling (scaling out) entails distributed computing across a large number of compute servers
- Challenges of distributed computing are:
  - The same code should transparently work on 1, 10, or 10,000 servers
    - Assume problem can be broken down into chunks and each chunk calculated locally
  - The data must be accessible from anywhere
  - High availability: the systems must survive one or more server failures with no impact on operations
  - Resource utilization must be optimized
    - Minimize hot-spots with a good load-balancing strategy
    - Bring compute to data
  - The systems should support elastic scaling
    - Add/remove machines without requiring maintenance down-times

Addressing the Big Data Challenge - Technologies

Distributed Computing

Spark, MapReduce(2), Storm, TEZ, Flink, …

Distributed Storage

HDFS, Alluxio(Tachyon), Ceph, S3, GPFS, …

Distributed data warehouses

Hive, SparkSQL, Hbase, Accumulo, Druid, Impala, Cassandra, MongoDB, CouchDB, Redis, Phoenix, JanusGraph, Neo4j, …

Provisioning, monitoring, management, security

Ambari, Zookeeper, Ranger, Knox, …

Data governance

Falcon, Atlas

Moving data around

Kafka, Flume, Scoop, Nifi

Workflow orchestration

Oozie, …

Resource Management

YARN, mesos, …

End user interfaces

Zeppelin, Jupyter Notebook, …

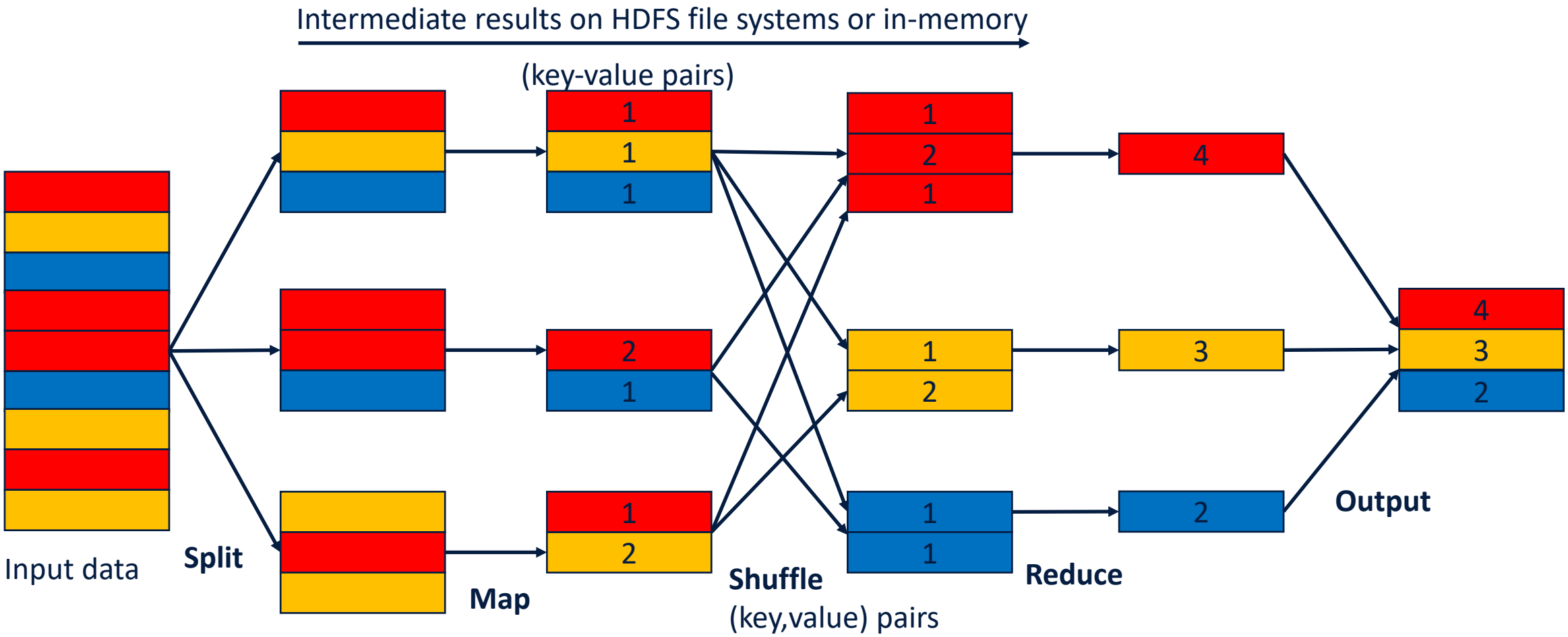
Hadoop Distributed File Systems (HDFS) Essentials

- What you need to know
  - **NameNode**: (master node) manage namespace, must have at least one, preferably two for high availability
  - **DataNode**: (worker node) serve the data, one per server
  - **Write-once**, you cannot modify a file in place (it must be replaced)
  - **Data blocks** are in units of 64MB (default)
    - Not for small files, use HDFS for large files only
  - **Redundancy**, all blocks replicated x3 (default)
    - Redundancy against failures
    - Statistically easier to move computation next to the data and load-balance the CPU usage
  - HDFS command line, with POSIX like interface (Hadoop2):
    - hdfs dfs [–help]

Popular Big Data Formats

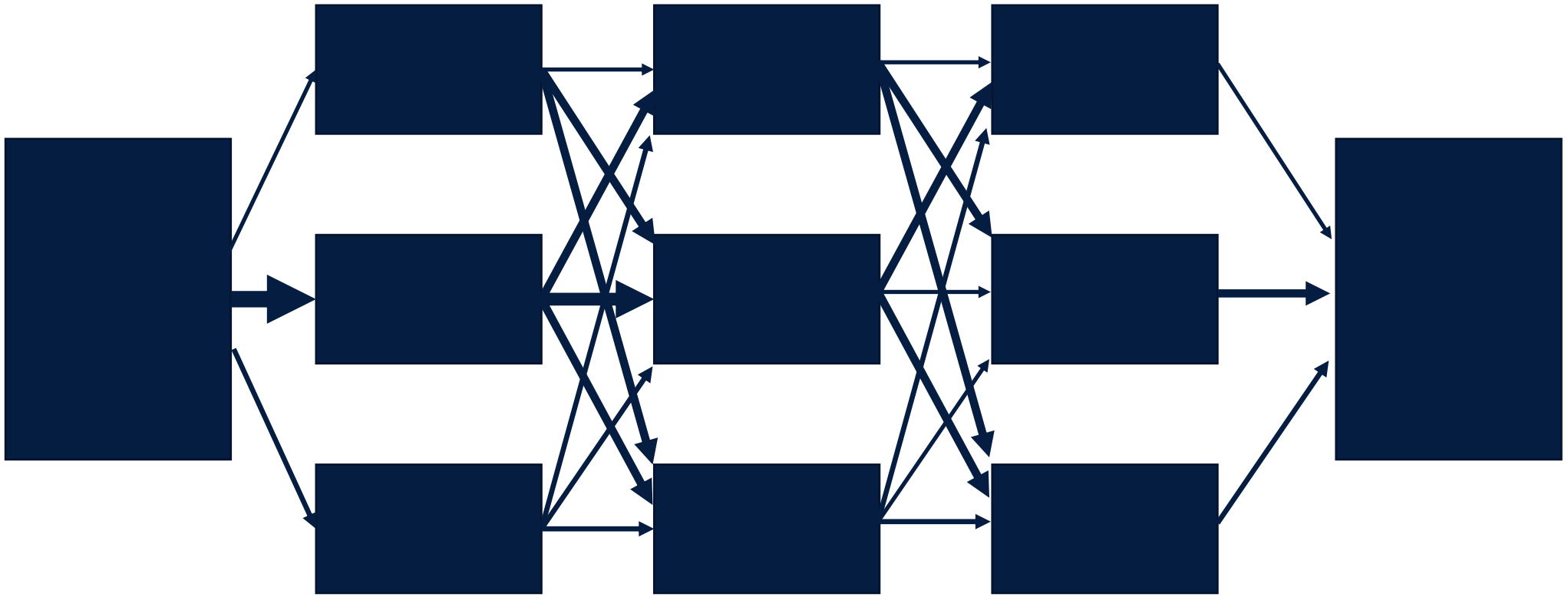
- Big data components (Hive, Spark, etc) give you several formatting options for exchanging data, the most popular are
  - **Plain text (**csv, json, …)
  - **Parquet**:
    - column-oriented
    - integrated compression
  - ORC:
    - column-oriented in collections of rows, splittable by row collections
    - integrated compression
    - indexed
  - Avro:
    - row-oriented, splittable and support block compression

MapReduce in a Nutshell



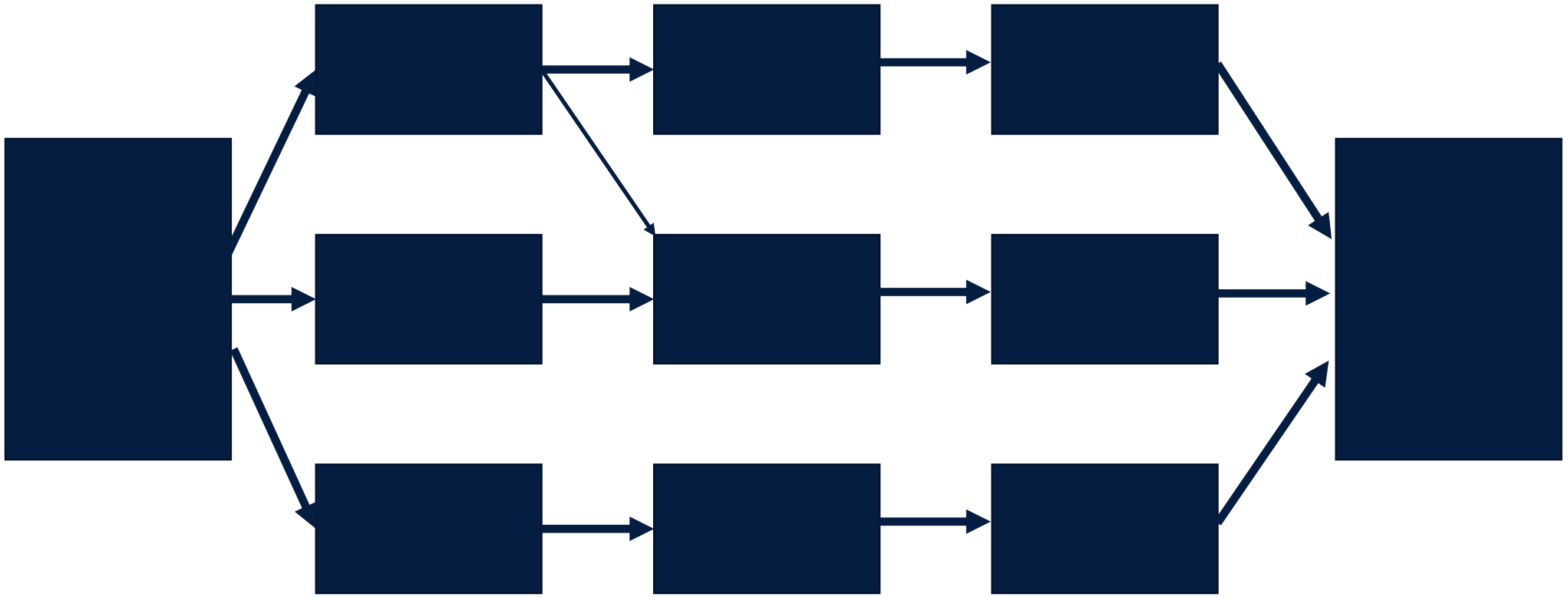
Notes: In HDFS the data is already ”split” into blocks and ”Mapping” can be done on all worker nodes where data blocks are located. However, beware of compressed data! not all compression algorithm are “splittable” in blocks, and MapReduce will not chunk it (splittable: bzip2, LZA, not splittable: gzip, snappy).

MapReduce gotchas



Shuffling is the network bottleneck of MapReduce operations, because placement of "reducers" cannot be optimized based on data locality.

MapReduce gotchas

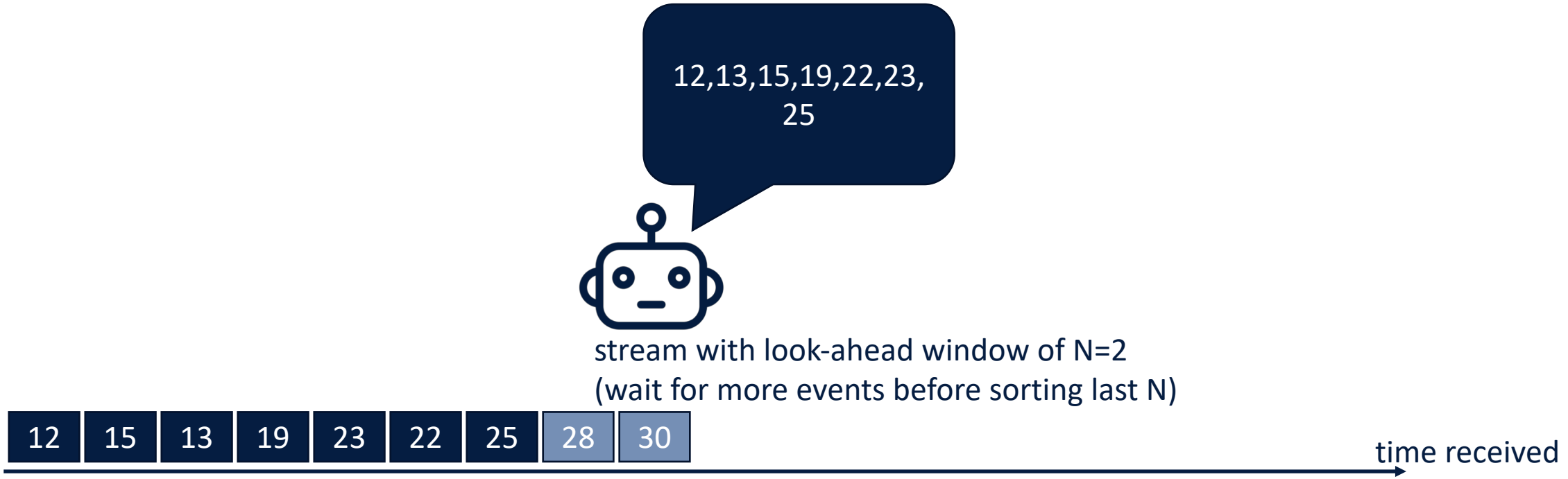


Optimization starts with good data partitioning practices to better balance the load (on CPU, RAM), and minimize data shuffling (network bottlenecks)

The Clash: should I Stream, or should I Batch?

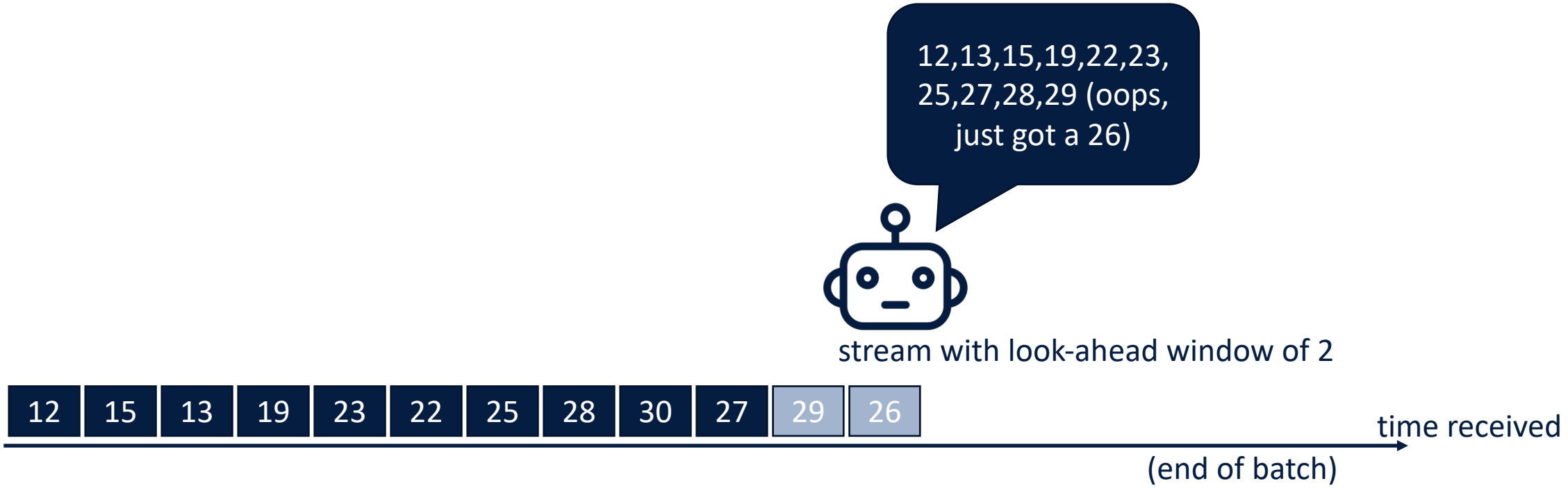
- You want an updated answer as more information becomes available? streams
  - AKA: Data in motion, or Fast data
  - Continuous computation that never stop, process infinite amount of data on the fly
    - Designed to keep size of in-memory state bounded, regardless of how much data is processed
  - Update the answer as more data becomes available
    - Operate on small time windows
  - E.g.:
    - Spark Streaming, Flink, Storm, Kafka Streams,
- Can wait until all information is available for a more accurate answer? batch
  - AKA: Data at rest
  - Operates on finite size data sets, and terminate when all data has been processed
    - Hadoop MR, Spark

The Clash: should I Stream, or should I Batch? (Illustrated)



Example: process events by timestamps, events can arrive out of order.

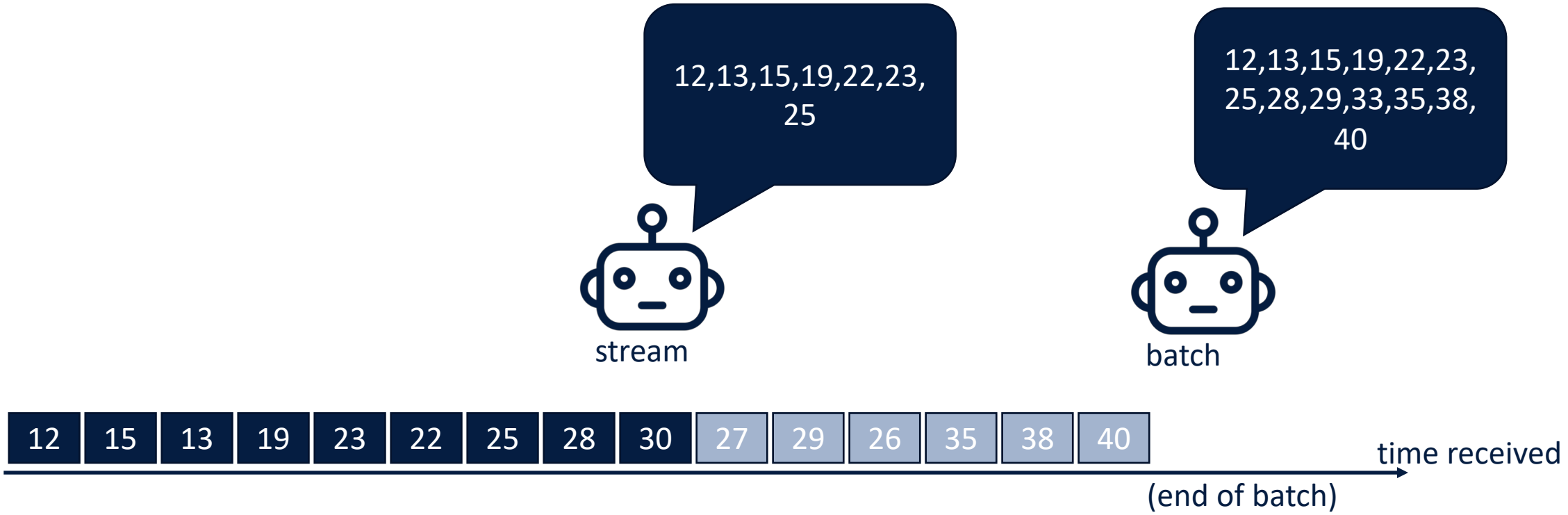
The Clash: should I Stream, or should I Batch? (Illustrated)



Example: sort events by timestamps, events can arrive out of order.

Streams: small windows (response is behind N=2 events in example), hence faster answers but can be inaccurate because can't predict events after look-ahead window, e.g. 26 is out of order in example. Response must be amended or event thrown out. We can increase N, which delays the response.

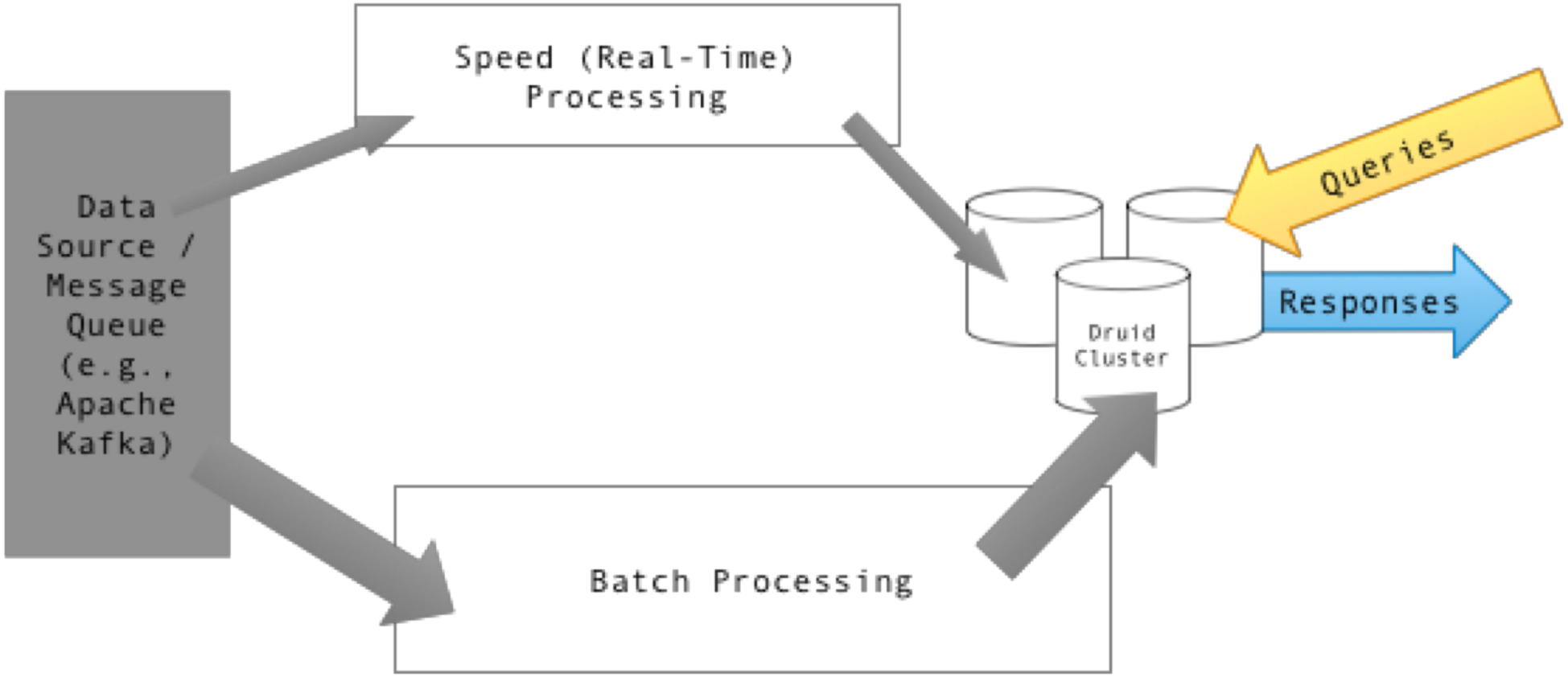
The Clash: should I Stream, or should I Batch? (Illustrated)



Example: process events by timestamps, events can arrive out of order.

Streams: faster answers but can be inaccurate (e.g. next events are out of order)
Batch: accurate, all information is available, but response is slower (must wait for all the events)

The Clash: should I Stream, or should I Batch? λ architecture



Big Data Illustrated – A Popular Real World Pattern

Real-time GPS Data

Data Stream (Kafka)

Prediction (Storm or Spark streaming)

ETA

Streams (continuous processing)

Model

Import (Scoop)

Historical Data DBMS (Hive)

Model Training (Spark)

Archives

Batch (e.g. learn a new model once a month)

Hadoop MapReduce Distributions

- On-Premise (self-hosted) Big Data Distributions
  - Hortonworks/Cloudera Hadoop Distribution (used in this lab)
  - MapR Hadoop Distribution
  - IBM Open Platform with Apache Hadoop
  - DC/OS, Mesosphere
  - …
- Cloud-based (all-purposes) Big Data Platforms as a Service
  - EPFL IC Cluster (used in this lab)
  - Amazon EMR
  - Microsoft Azure HDInsight
  - Alibaba Cloud E-MapReduce
  - …

This week's exercises

- Get the **DSLab Week 5** exercise instructions:
  - https://github.com/dslab2019/dslab2019.github.io/tree/master/labs/week5
- Confirm that you can login on iccluster042 and have a home directory in HDFS, contact us if you don't
- Communications:
  - https://mattermost-dslab.epfl.ch
  - Remember we have office hours on Fridays for the graded projects