MapReduce and Spark: Overview

2015

Professor Sasu Tarkoma

Overview

Computing Environment

History of Cluster Frameworks

Hadoop Ecosystem

Overview of State of the Art

MapReduce Explained

Computing Environment

- Scaling up
  - More powerful servers
- Scaling out
  - More servers
- Clusters provide computing resources
  - Space requirements, power, cooling
  - Most power converted into heat
- Datacenters
  - Massive computing units
  - **Warehouse-sized computer** with hundreds or thousands of racks
- Networks of datacenters

Cluster Computing Environment

Big Data compute and storage nodes are stored on racks based on common off the shelf components

Typically many racks in a cluster or datacenter

The compute nodes are connected by a high speed network (typically 10 Gbit/s Ethernet)

- Different datacenter network topologies

Intra-rack and inter-rack communication have differing latencies

Nodes can fail

- Redundancy for stored file (replication)
- Computation is task based

Software ensures fault-tolerance and availability

Typical Hardware

CSC Pouta Cluster running on the Taito supercluster in Kajaani

The nodes are HP ProLiant SL230s servers with two Intel Xeon 2.6 GHz E5-2670 CPUs

16 cores per server

Most with 64 GB of RAM per server

Taito extension in 2014: 17 000 cores

The nodes are connected using a fast FDR InfiniBand fabric

Big Data Tools for HPC and Supercomputing

MPI (Message Passing Interface, 1992)

Communication between parallel processes

Collective communication operations

Broadcast, Scatter, Gather, Reduce, Allgather, Allreduce, Reduce-scatter

Operations defined for certain data types and primitives (such as multiplication etc)

For example OpenMPI (2004)

http://www.open-mpi.org/

Cloud Computing

Definition by NIST:

Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.

IaaS, PaaS, SaaS, XaaS

Big Data Frameworks are typically run in the cloud

Big Data Environment

Typically common-of-the-shelf servers

Compute nodes, storage nodes, …

Virtualized resources running on a cloud platform

Heterogeneous hardware, choice of OS

Contrasts traditional High Performance Computing (HPC)

History of Cluster Frameworks

2003: Google GFS

2004: Google Map-Reduce

2005: Hadoop development starts

2008: Apache Hadoop (in production)

2008: Yahoo! Pig language

2009: Facebook Hive for data warehouses

2010: Cloudera Flume (message interceptor/filtering model)

2010: Cloudera S4 (continuous stream processing)

2011: LinkedIn Kafka (topic-based commit log service)

2011: Storm (Nathan Marz)

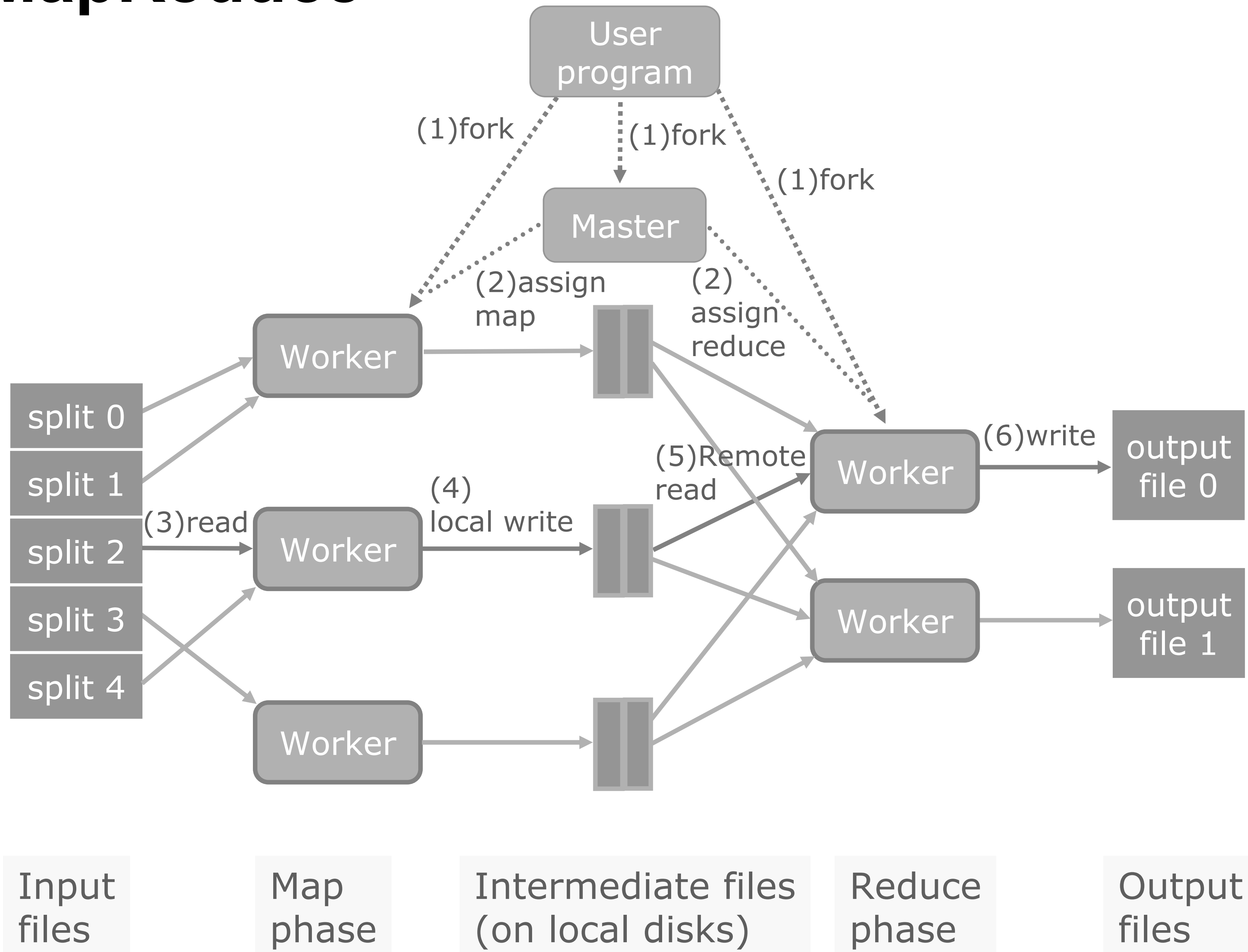
2011: Apache Mesos cluster management framework

2012: Lambda Architecture (Nathan Marz)

2012: Spark for iterative cluster programming

2013: Shark for SQL data warehouses

MapReduce



User program
(1)fork
(1)fork
(1)fork
Master
(2)assign map
(2) assign reduce
Worker
Worker
Worker
Worker
Worker
split 0
split 1
split 2
split 3
split 4
(3)read
(4) local write
(5)Remote read
(6)write
output file 0
output file 1
Input files
Map phase
Intermediate files (on local disks)
Reduce phase
Output files

Major trends

Apache Hadoop

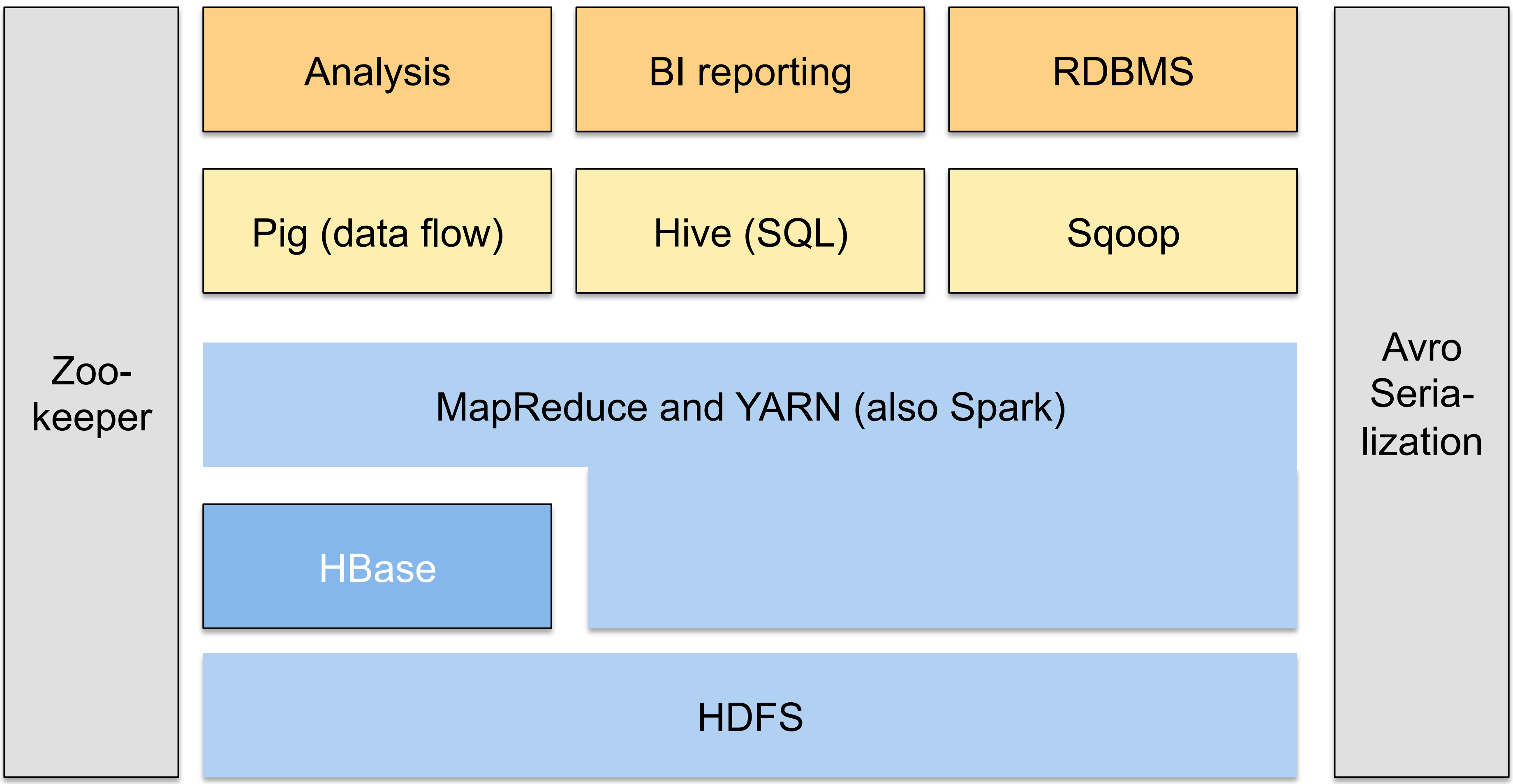
Hive, R, and others

Berkeley Data Analytics Stack (BDAS)

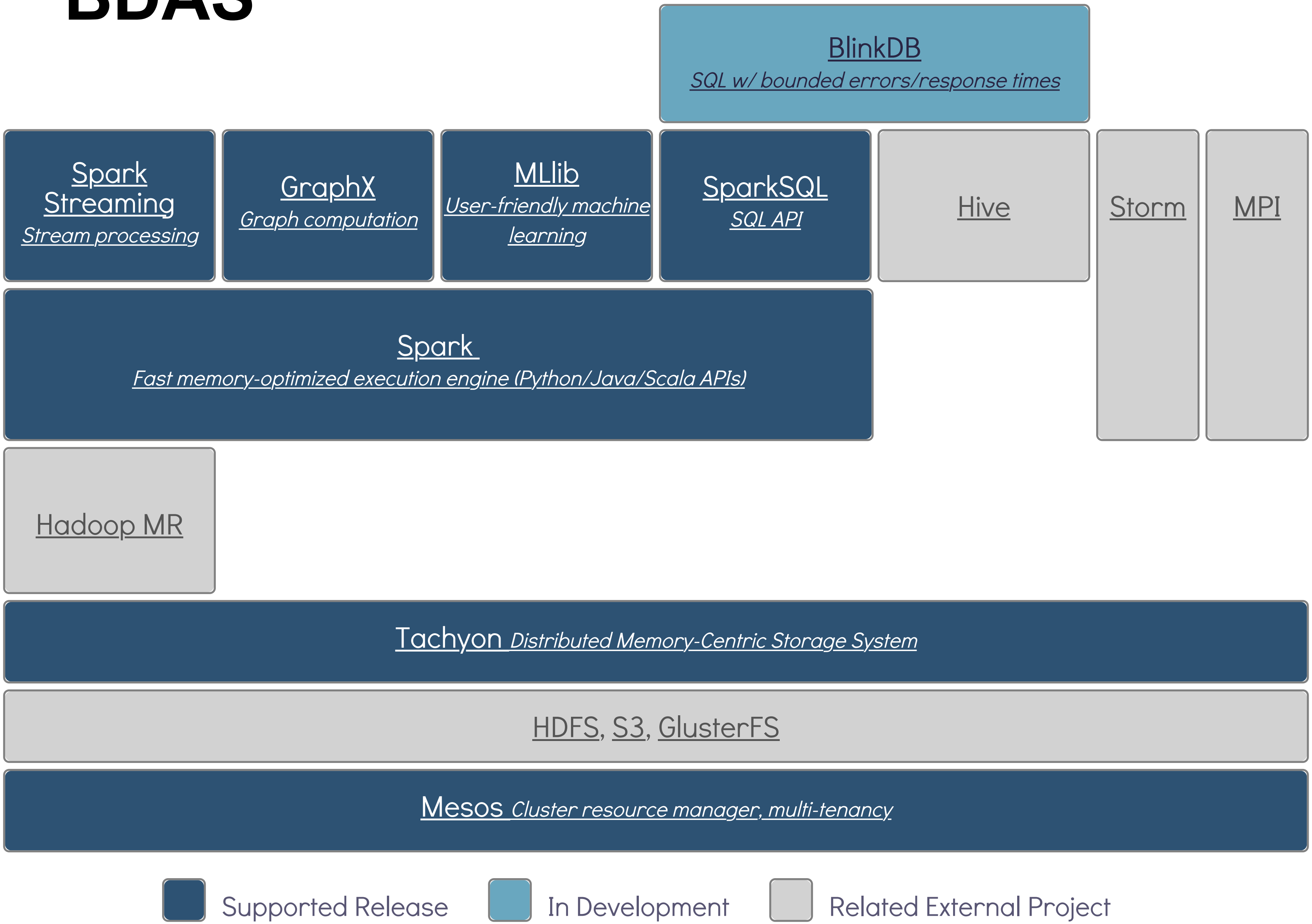
Mesos, Spark, Mlib, GraphX, Shark, …

Apache Spark is part of Apache Hadoop

Apache Hadoop Ecosystem



BDAS



https://amplab.cs.berkeley.edu/software/

Key idea in Spark

Resilient distributed datasets (RDDs)

- Immutable collections of objects across a cluster
- Built with parallel transformations (map, filter, …)
- Automatically rebuilt when failure is detected
- Allow persistence to be controlled (in-memory operation)

Transformations on RDDs

- Lazy operations to build RDDs from other RDDs
- Always creates a new RDD

Actions on RDDs

- Count, collect, save

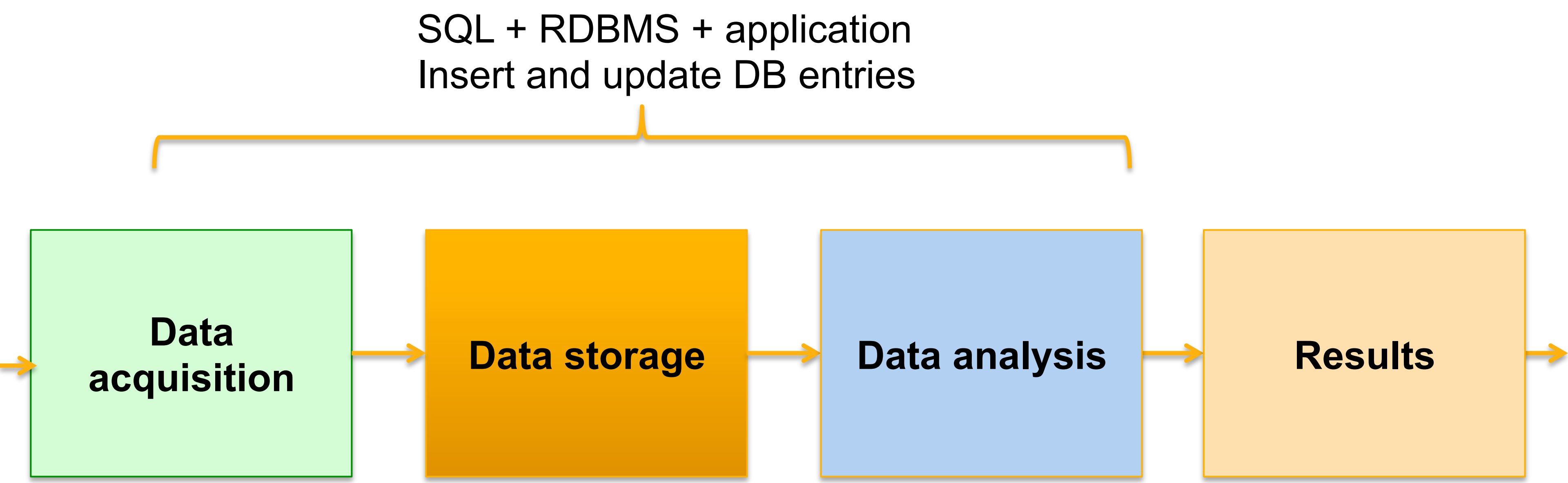
MPP Databases

Massive Parallel Processing Databases (MPP)

- Vertica, SAP HANA, Teradata, Google Dremel, Google PowerDrill, Cloudera Impala…

- Fast but typically not fault-tolerant
- Scaling up can be challenging
- Lack of rich analytics (machine learning and graphs)

Traditional SQL Approach

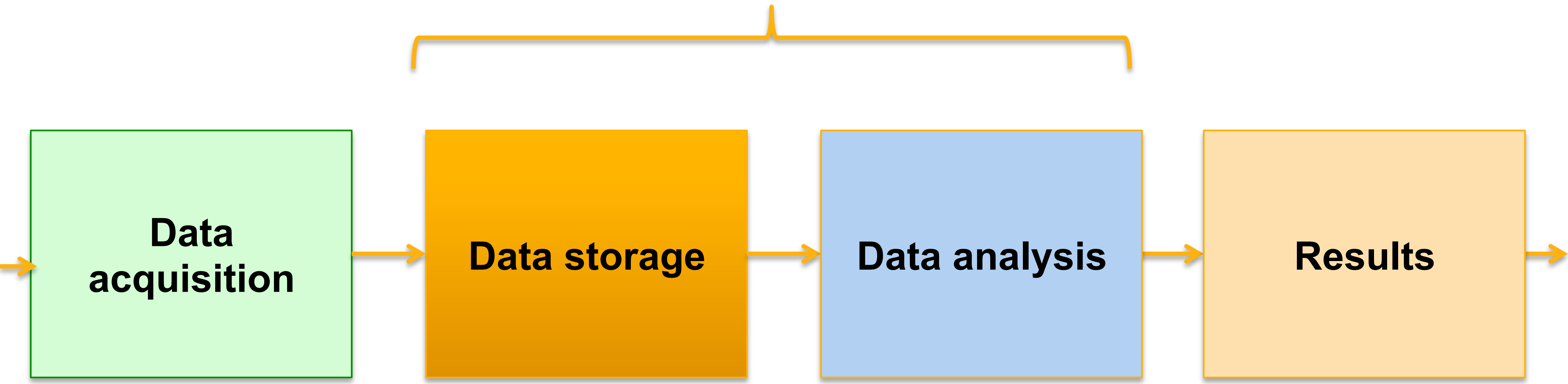


Example: counting twitter hashtags
1. INSERT VALUES of new tweets
2. Create a new table every 5 minutes with counts: CREATE .. SELECT … COUNT(*) GROUP BY time, tag.
3. Combine new table with old count table (UNION), this is the new table

Inspiration: http://www.slideshare.net/Dataiku/dataiku-devoxx-lambda-architecture-choose-your-tools

Functional programming

Append only new data
Intrinsically parallel operations
MapReduce
Iterative computing



Example: counting twitter hashtags

1. Map (#tag, time) -> list (#tag, intermediate count)
2. Reduce (#tag, hashmap) -> list (#tag, count)

Inspiration: http://www.slideshare.net/Dataiku/dataiku-devoxx-lambda-architecture-choose-your-tools

Overview of State of the Art

- Data storage
- Data storage for real-time
- Data analysis
- Real-time data analysis
- Statistics and machine learning

State of the Art: Data Storage

GFS (Google File System) and HDFS (Hadoop Distributed File System)

- Data replicated across nodes
- HDFS: rack-aware placement (replicas in different racks)
- Take data locality into account when assigning tasks
- Do not support job locality (distance between map and reduce workers)

Hbase

- Modeled after Google's BigTable for sparse data
- Non-relational distributed column-oriented database
- Rows stored in sorted order

Sqoop

- Tool for transferring data between HDFS/Hbase and structural datastores
- Connectors for MySQL, Oracle, … and Java API

Example: HDFS Architecture

HDFS has a master/slave architecture

NameNode is the master server for metadata

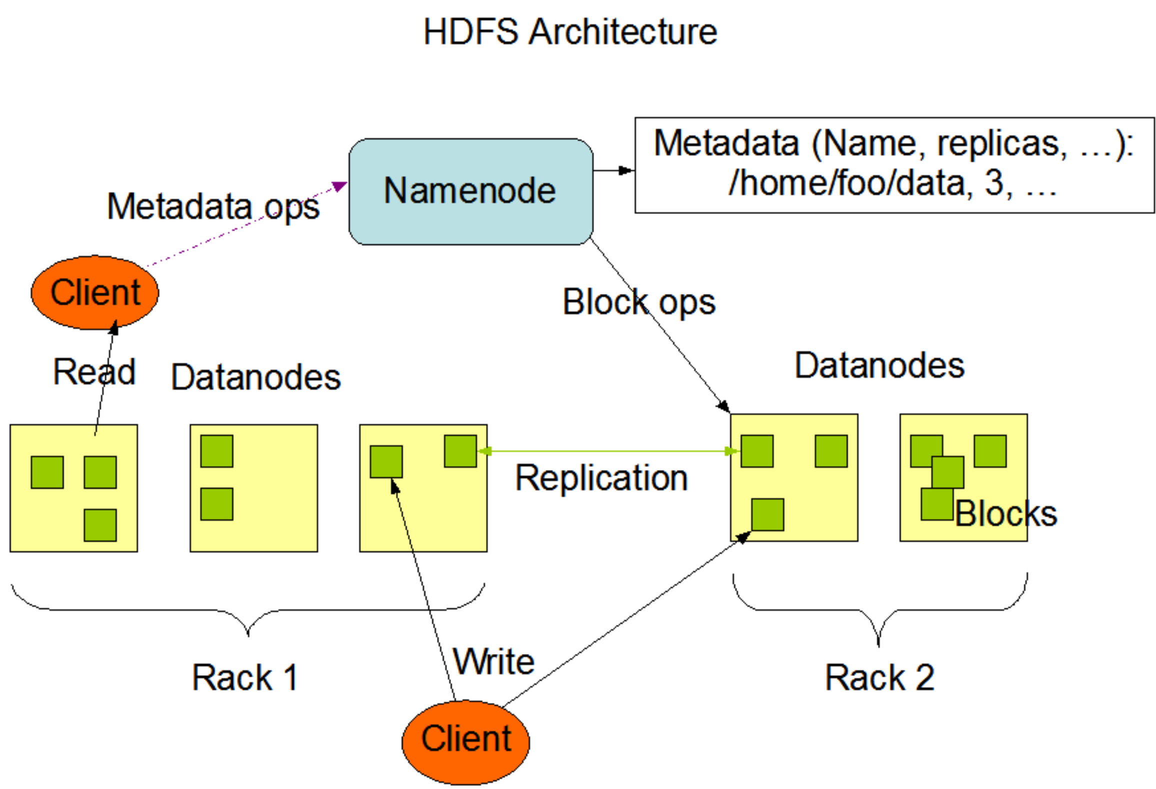
DataNodes manage storage

A file is stored as a sequence of blocks

The blocks are replicated for fault-tolerance

Common replication scheme: factor of 3, one replica local, two in a remote rack

Rack-aware replica placement



Namenode provides information for retrieving blocks
Nearest replica is used to retrieve a block

http://hadoop.apache.org/docs/r1.2.1/images/hdfsarchitecture.gif

State of the Art: Data Storage for Real-time

Kafka

Distributed, partitioned, replicated commit log service

Keeps messages in categories

Topic based system

Coordination through Zookeeper (through distributed consensus)

Kestrel

Distributed message queue (server has a set of queues)

A server maintains queues (FIFO)

Does not support ordered consumption

Simpler than Kafka

State of the Art: Data Analysis I/II

MapReduce

- Map and reduce tasks for processing large datasets in parallel

Hive

- A data warehouse system for Hadoop
- Data summarization, ad-hoc queries, analysis for large sets
- SQL-like language called HiveQL

Pig

- Data analysis platform
- High-level language for defining data analysis programs, Pig Latin, procedural language

Cascading

- Data processing API and query planner for workflows
- Supports complex Hadoop Map-Reduce workflows

Apache Drill

- SQL query engine for Hadoop and noSQL

State of the Art: Data Analysis II

Spark

- Cluster computing for data analytics
- In-memory storage for iterative processing

Shark

- Data warehouse system (SQL) for Spark
- Up to 100x faster than Hive

Spark/Shark is a distinct ecosystem from Hadoop

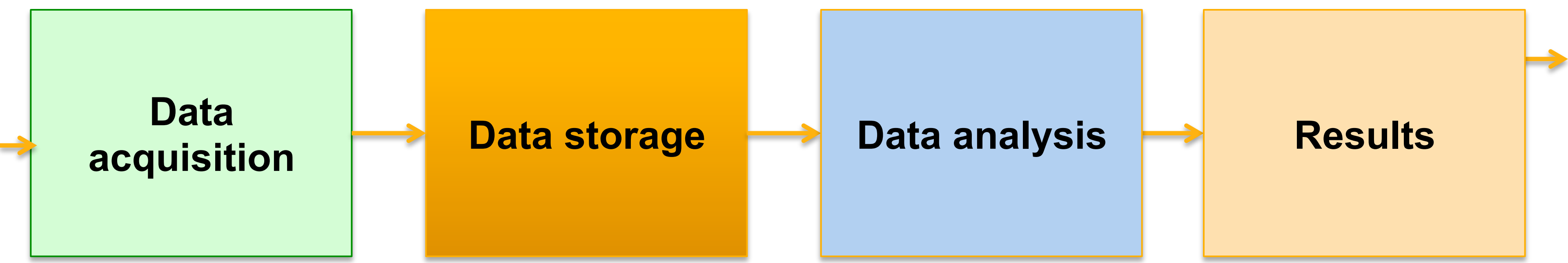
- Faster than Hadoop
- Support for Scala, Java, Python
- Can be problematic if reducer data does not fit into memory

Summary of batch systems

HDFS import
Flume
Sqoop
…

HDFS
Hbase
…

MapReduce
Hive
Pig
Spark
Shark
…



State of the Art: Real-time Data Analysis I/II

Flume

- Interceptor model that modifies/drops messages based on filters
- Chaining of interceptors
- Combine with Kafka

Storm

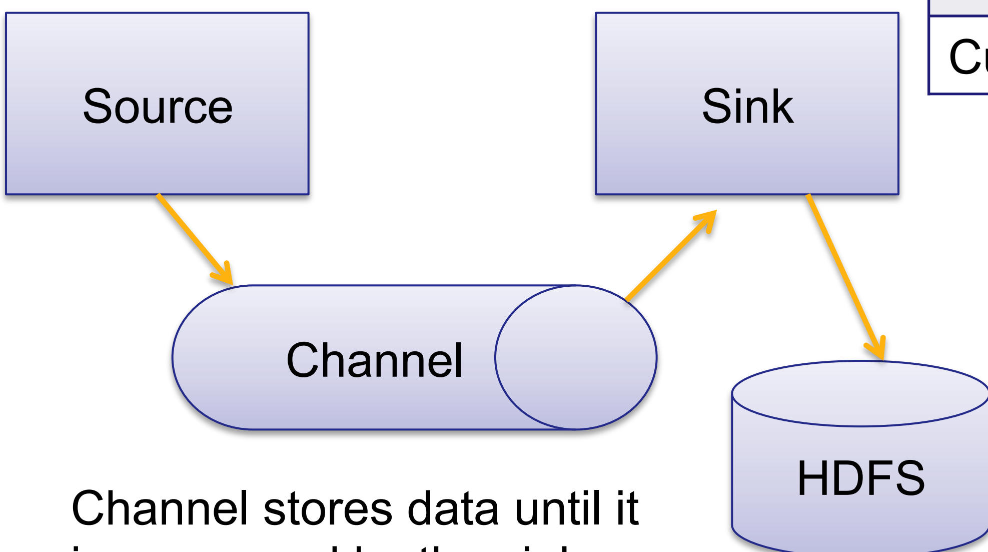
- Distributed realtime computation framework
- “Hadoop for realtime”
- Based on processing graph, links between nodes are streams

Trident

- Abstraction on top of Storm
- Operations: joins, filters, projections, aggregations, ..
- Exactly once-semantics (replay tuples for fault tolerance, stores additional state information)
- https://storm.apache.org/documentation/Trident-state

Flume example

Source	Channel	Sink
Avro	Memory	HDFS
Thrift	JDBC	Logger
Exec	File	Avro
HTTP		Null
JMS		Thrift
Syslog TCP/IP		File roll
		Hbase
Custom		Custom



Channel stores data until it is consumed by the sink.

Storm

Developed around 2008-2009 at BackType, open sourced in 2011

Spout: is a flow of tuples

Bolt: accepts tuples and operates on those

Topologies: spouts → bolts → spouts

Example:

Tweet spout → parse Tweet bolt → count hashtags Bolt

Tweet spout → store in a file

http://www.slideshare.net/Dataiku/dataiku-devoxx-lambda-architecture-choose-your-tools

State of the Art: Real-time Data Analysis II

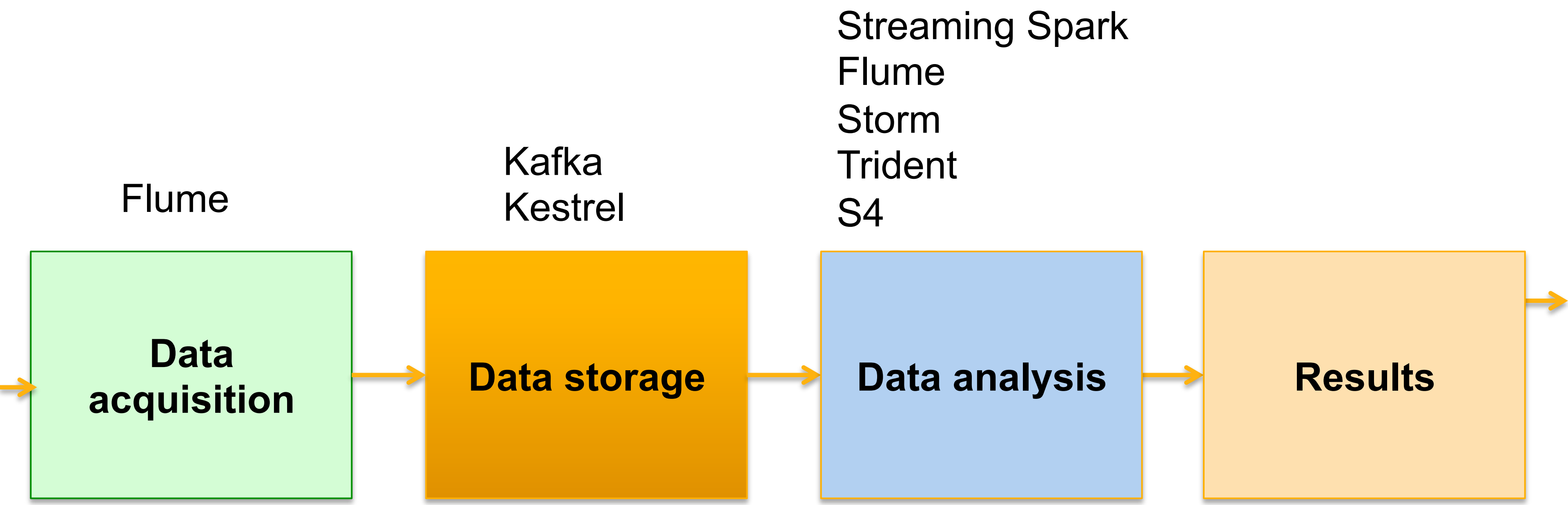
Simple Scalable Streaming System (S4)

- Platform for continuous processing of unbounded streams
- Based on processing elements (PE) that act on events (key, attributes)

Spark streaming

- Spark for real-time streams
- Computation with a series of short batch jobs (windows)
- State is kept in memory
- API similar to Spark

Summary of real-time processing



State of the art: Hybrid models

Lambda architecture combined batch and stream processing

Supports volume (batch) + velocity (streaming)

Hybrid models

SummingBird (Hadoop + Storm)

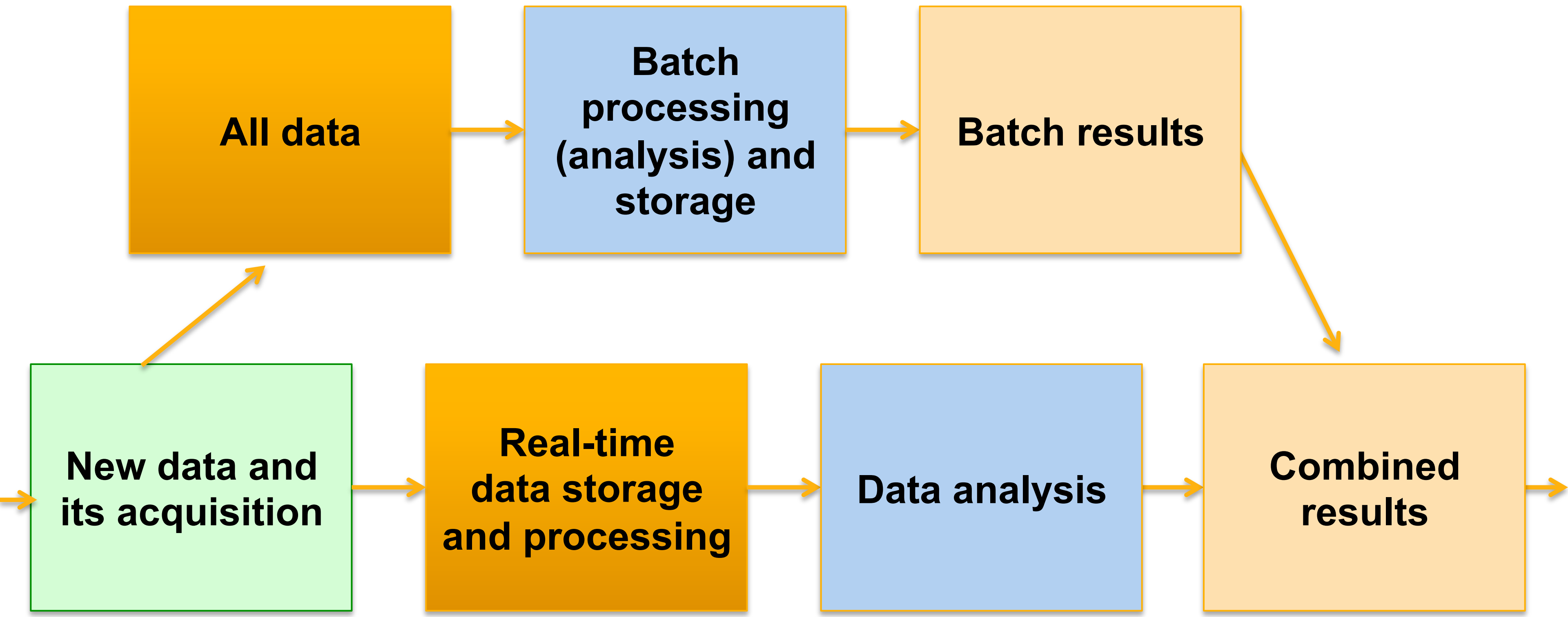
MapReduce like process with Scala syntax

Lambdoop (abstraction over Hadoop, HBase, Sqoop, Flume, Kafka, Storm, Trident)

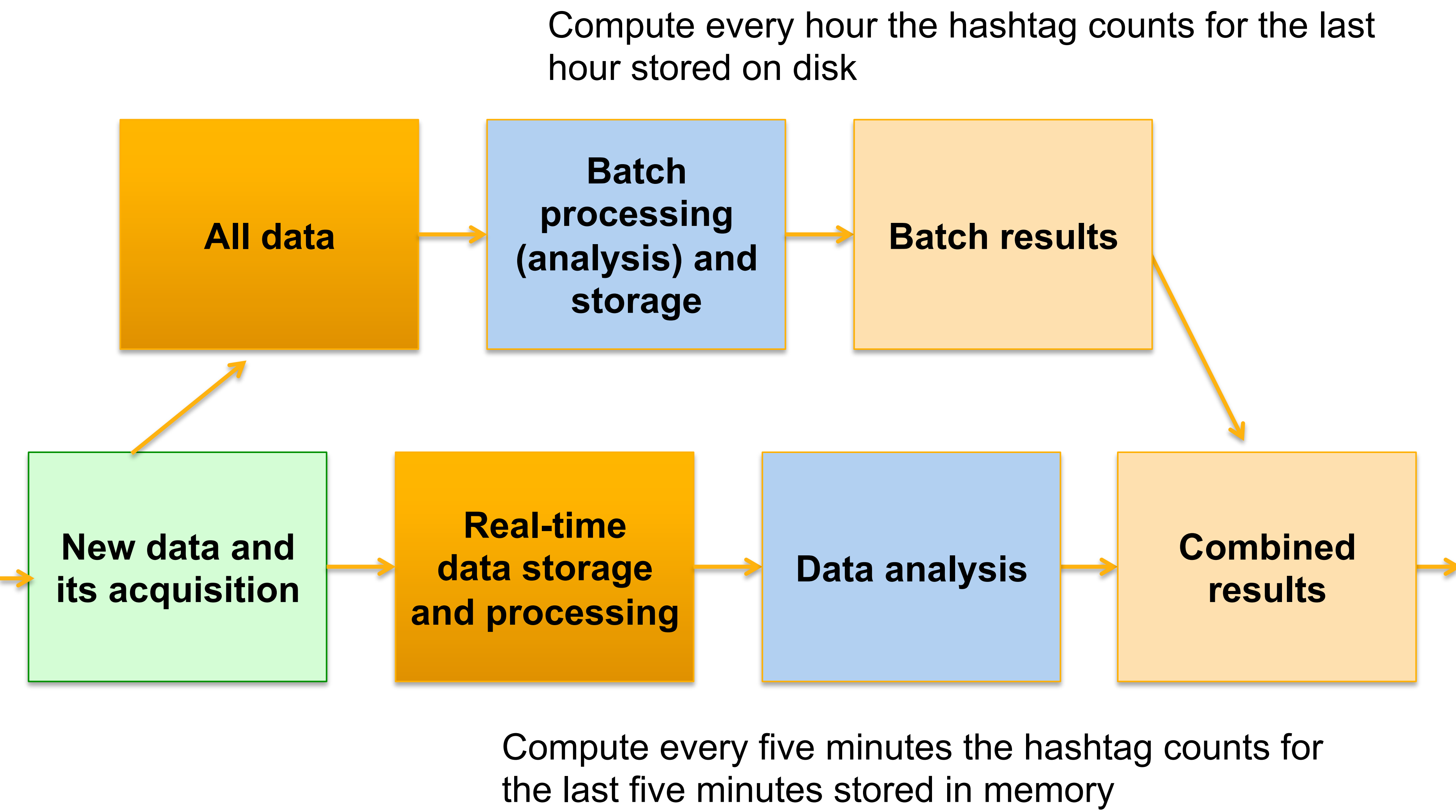
Common patterns provided by platform

No MapReduce like process

Lambda Architecture

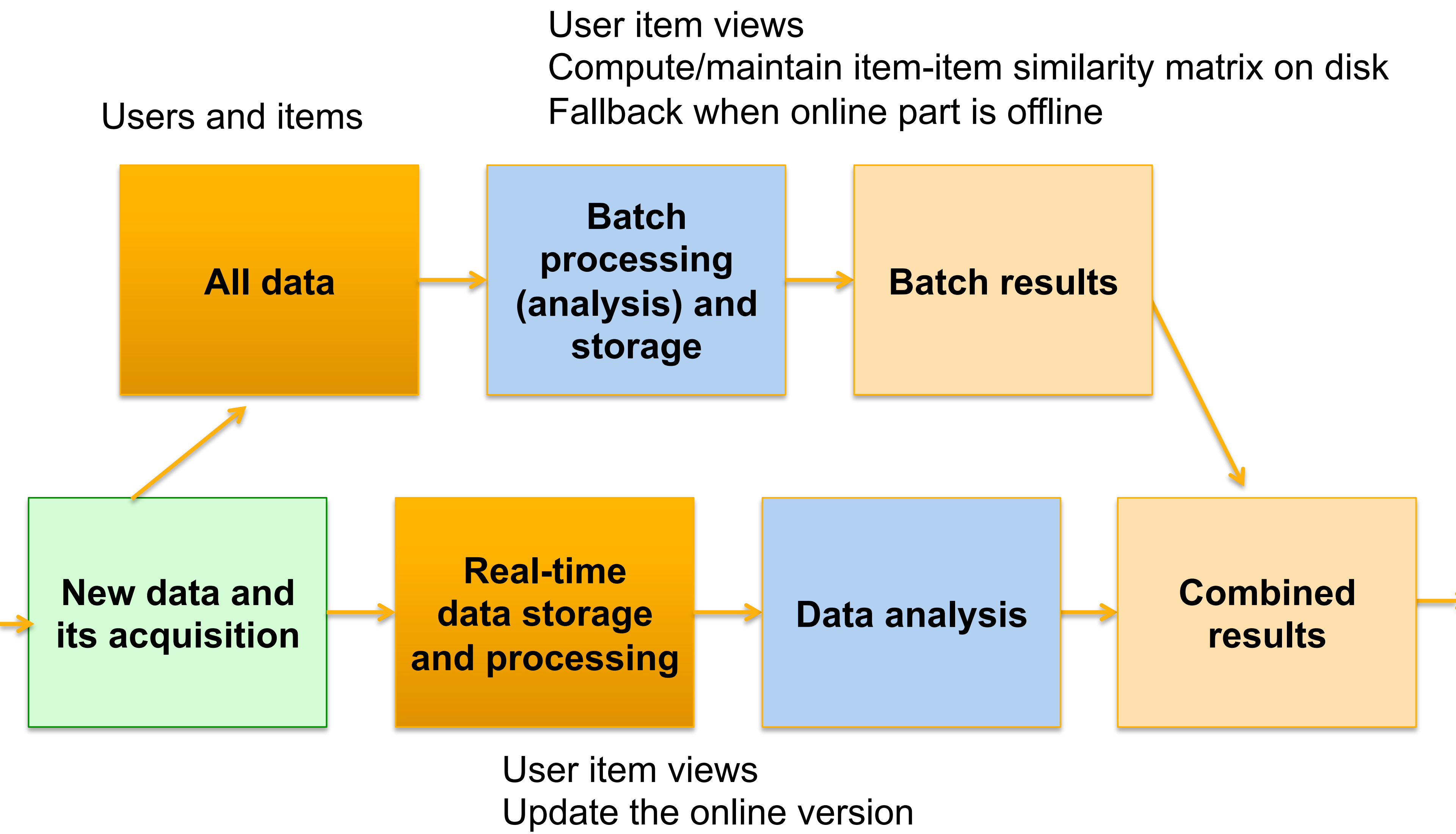


Lambda Architecture: Twitter hashtags



Inspiration: http://www.slideshare.net/Dataiku/dataiku-devoxx-lambda-architecture-choose-your-tools

Lambda Architecture: Recommendations



Inspiration: http://www.slideshare.net/Dataiku/dataiku-devoxx-lambda-architecture-choose-your-tools

Challenges for the platform

Exactly-once semantics

- Requires costly synchronization

High velocity: how to go to thousands of messages per second

Changes to structures and schemas

- Data versioning in a production system

Solution pipelines in Lambda architecture

Batch pipeline

- Flume → HDFS → MapReduce → HBase → combined view → App

Realtime pipeline

- RabbitMQ → Storm → Memcache → MongoDB combined view→ App

State of the Art: Statistics and Machine Learning

R for Hadoop

- Distributed R for the cluster environment

R for Spark

Mahout

- Currently Hadoop, next Spark

Weka

- State of the art machine learning library
- Does not focus on the distributed case
- Hadoop support, Spark wrappers

MLLib

- Machine learning for Spark

Summary of Big Data Tools for Data Mining

- Apache Mahout
  - Originally Hadoop, now Spark
  - Scalable machine learning library
  - Collaborative filtering, clustering, classification, frequent pattern mining, dimensionality reduction, topic models, …
- Weka
- R: software environment for statistical computing
  - Spark-R
  - Rhadoop
  - Revolution R: commercial
- Spark
  - MBase and MLlib
- **Division into efficient tools that do not scale to clusters and emerging cluster solutions (Hadoop / Spark)**

State of the Art Distributed Toolbox