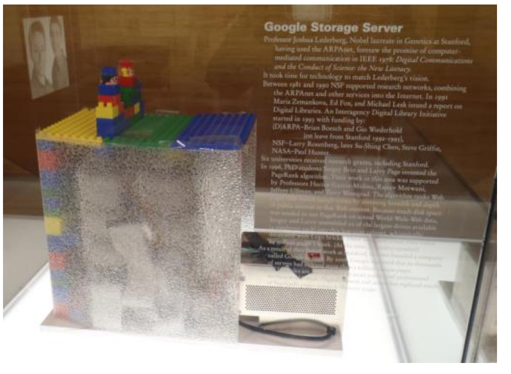
POZNAN UNIVERSITY OF TECHNOLOGY

Big Data Architectures

Robert Wrembel
Poznan University of Technology
Institute of Computing Science
Poznań, Poland
Robert.Wrembel@cs.put.poznan.pl
www.cs.put.poznan.pl/rwrembel



Outline

- Introduction to Big Data
- NoSQL data storage
- Big Data ingest architectures
- GFS, HDFS, Hadoop
- Types of data processing
- Big Data architectures
- Data ingest tools
- Big Data integration taxonomy
- Other technologies

Big Data

➲ **Huge data Volume and Velocity**

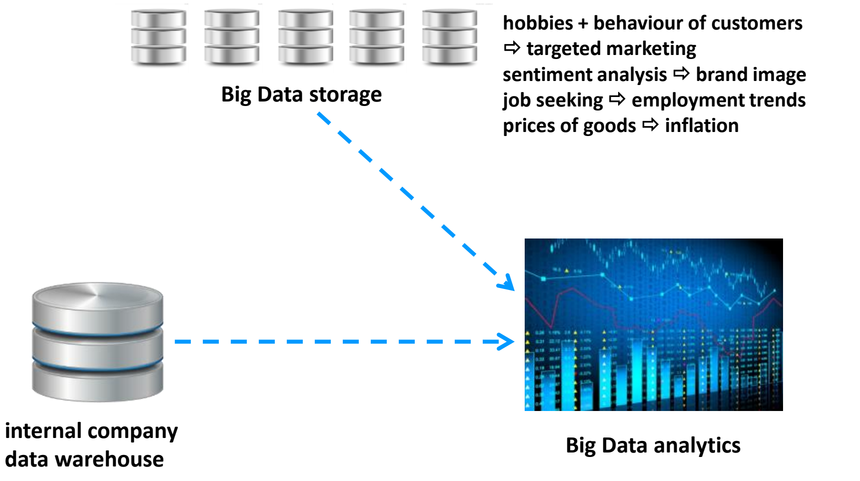
- **every minute:**
  - **over 200 million e-mail messages are sent**
  - **over 100 000 tweets are sent (~ 80GB daily)**
  - **a single jet engine can generate 10TB of data in 30 minutes**
- **human generated:**
  - **social portals**
  - **foras and blogs**
- **machine generated:**
  - **web logs**
  - **sensors**

Big Data

➲ **Variety (heterogeneity) of data formats**

- **structured (relational)**
- **structured (time series)**
- **semistructured (e.g., XML, JSON)**
- **unstructured**
- **semantic Web (e.g., XML, RDF, OWL)**
- **geo-related data**
- **graphs**
- **large texts**
- **multimedia**

Big Data



Big Data characteristics

	Traditional	Big Data
storage	relational DBMS	NoSQL + HDFS
scaling	vertical	horizontal
processing	batch, offline	real-time, streaming, batch, offline
data quality	very high	very low

Data Stores

- **NoSQL**
- **Key-value DB**
  - **data structure ⇨ collection**
  - **collection is represented as a pair: key and value**
  - **data have no defined internal structure ⇨ the interpretation of complex values must be made by a program**
  - **operations ⇨ create, read, update (modify), and delete (remove) individual data - CRUD**
  - **the operations process one data item (selected by the value of its key) at a time**
  - **Voldemort, Riak, Redis, Scalaris, Tokyo Cabinet, MemcacheDB, DynamoDB**

Data Stores

- **Column family (column oriented, extensible record, wide column)**
  - **definition of a data structure includes**
    - **key definition**
    - **column definitions**
    - **column family definitions**

	column family CF1			column family CF2	
	Col1	Col2	Col3	Col4	Col5
row key K1	value	value			value
row key K2		value	value		value
row key K3	value	value	value	value	value
row key K4					
row key Kn	value			value	

Data Stores

- column family ⇨ stored separately, common to all data items (~ shared schema)
- column ⇨ stored with a data item, optional, specific for the data item
- CRUD interface
- HBase, HyperTable, Cassandra, BigTable, Accumulo, SimpleDB

Data Stores

➲ Document DB

- typically JSON-based structure of documents
- SimpleDB, MongoDB, CouchDB, Terrastore, RavenDB, Cloudant

➲ Graph DB

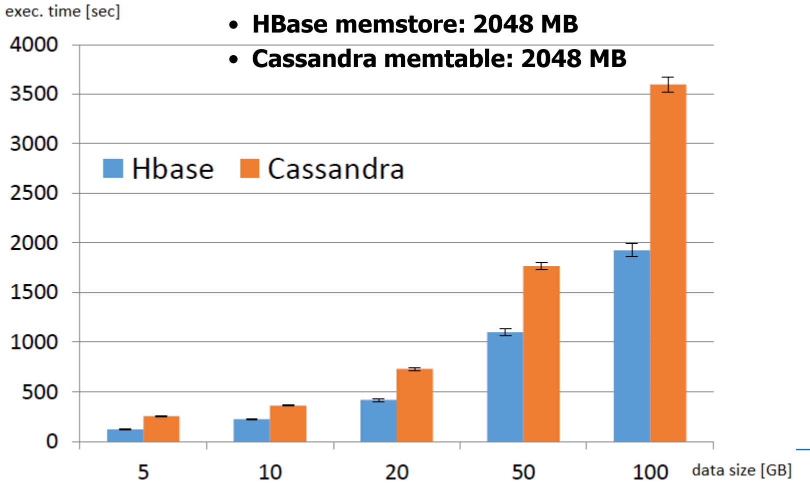
- nodes, edges, and properties to represent and store data
- every node contains a direct pointer to its adjacent element
- Neo4j, FlockDB, GraphBase, RDF Meronymy SPARQL

Performance evaluation

➲ **HBase ⇔ Cassandra**
- **virtual machines 8 CPUs, 16 GBs RAM, 480 GB HDD**
- **Ubuntu (14.04.1 LTS)**
- **Cassandra 2.0.14**
- **HBase 1.0.0 + Hadoop 2.5.2**
- **2 Cassandra data nodes**
- **2 nodes (HBase RegionServer + Hadoop DataNode) + 1 node (HBase MasterServer + Hadoop NameNode)**
- **Yahoo Cloud Serving Benchmark with modified workloads**

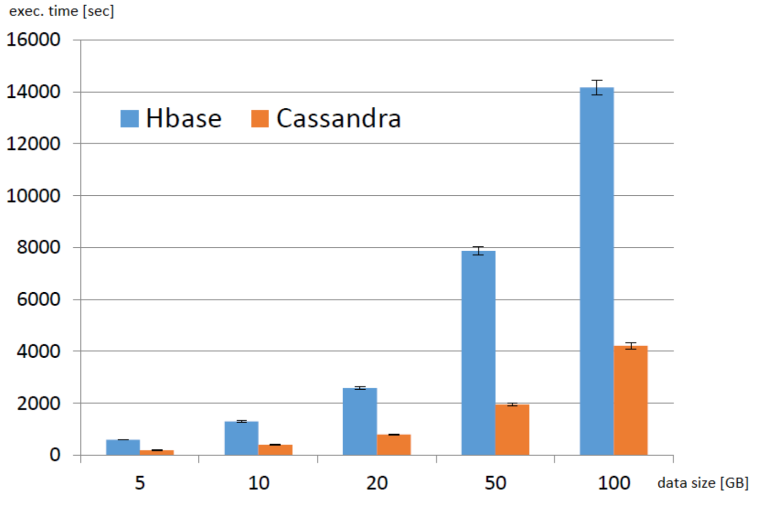
HBase - Cassandra

➲ **Read-only workload**
- **# of threads for HBase and Cassandra: 256**
- **cache size:**
  - **HBase memstore: 2048 MB**
  - **Cassandra memtable: 2048 MB**



HBase - Cassandra

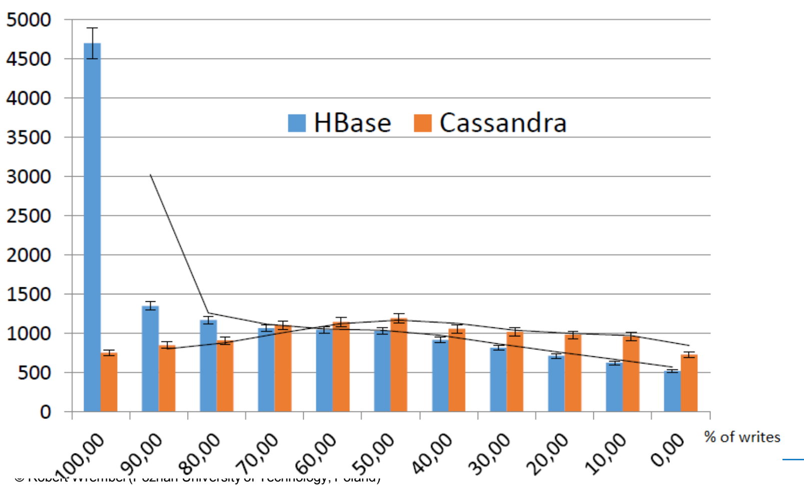
➲ **Write-only workload**



HBase - Cassandra

➲ **Read-write workload**

- **data volume: 20 GB**



GFS

- ➲ **Google implementation of a distributed file system**
  (The Google File System - whitepaper)
- ➲ **Developed to handle**
  - **hundreds of TBs of storage, thousands of disks, over a thousands of cheep commodity machines**
- ➲ **The architecture is failure sensitive ⇨ therefore**
  - **fault tolerance**
  - **error detection**
  - **automatic recovery and**
  - **constant monitoring are required**

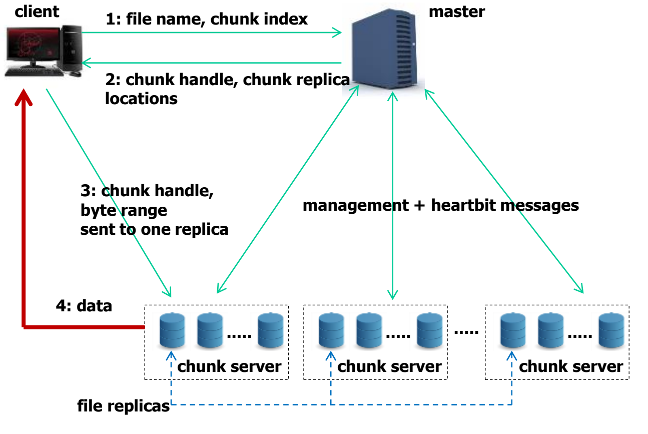
GFS

- ➲ **Files are organized hierarchically in directories**
- ➲ **Files are identified by their pathnames**
- ➲ **File size at least 100MB**
- ➲ **Typical file size: multiple GB**
- ➲ **Millions of files**
- ➲ **File usage**
  - **mostly appending new data ⇨ multiple large sequential writes**
  - **no updates of already appended data**
  - **mostly large sequential reads**
  - **small random reads occur rarely**
  - **replicated files (default 3)**

GFS

- **Operations on files**
  - create
  - delete
  - open
  - close
  - read
  - write
  - snapshot (creates a copy of a file or a directory tree)
  - record append (appends data to the same file concurrently by multiple clients)
- **Simple GFS cluster includes**
  - single master
  - multiple chunk servers

GFS

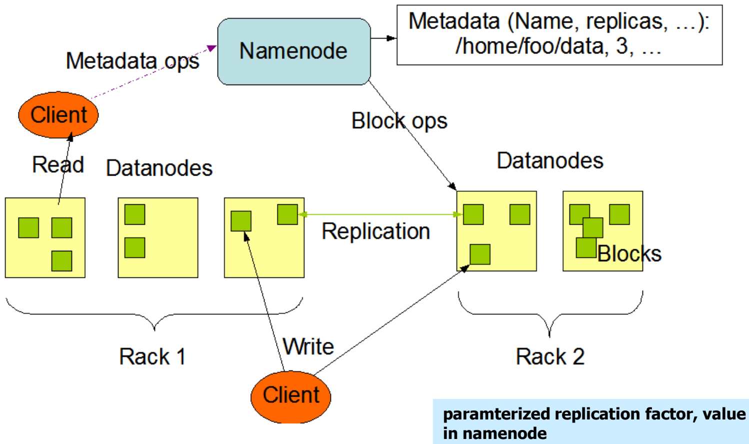


S. Ghemawat, H. Gobioff, S-T. Leung. The Google File System. http://research.google.com/archive/gfs.html

HDFS

➲ **Apache implementation of DFS**

http://hadoop.apache.org/docs/stable/hdfs_design.html



paramterized replication factor, value stored in namenode

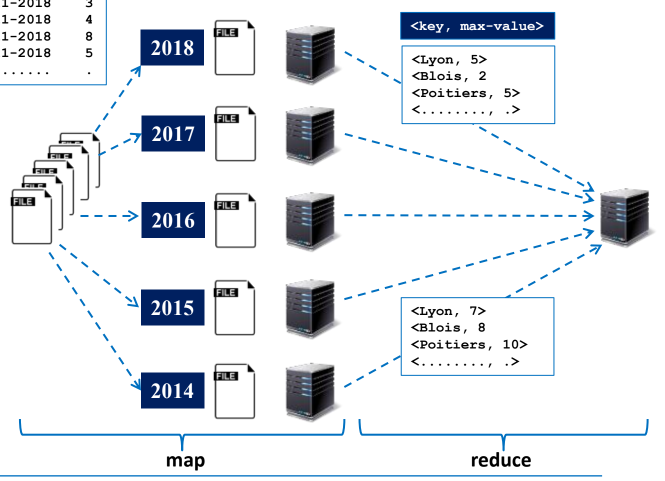
Storage

➲ **Distributed file systems**
- **Amazon Simple Storage Service (S3)**
- **Gluster (open source)**

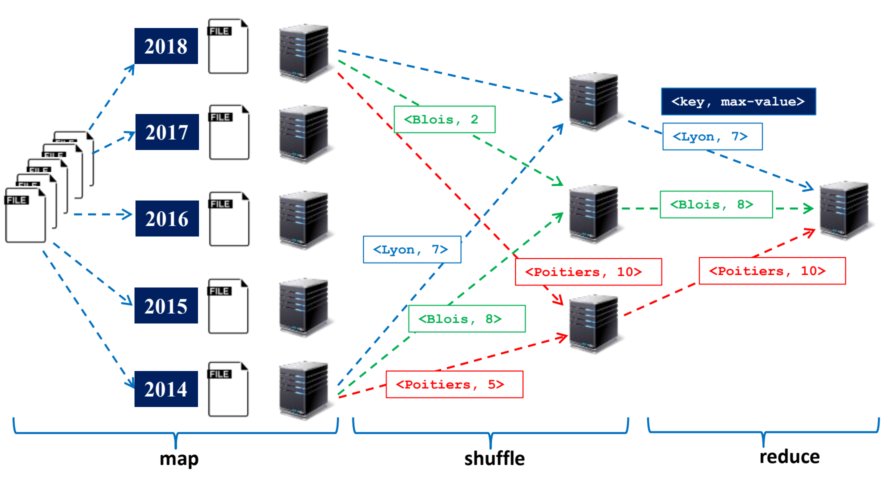
➲ **Storage formats**
- **Apache Avro for storing serialized data in JSON for Hadoop**
- **Apache Parquet - column oriented data store for Hadoop**

Map reduce

stationID	cityID	timestamp	PM10
A01	Lyon	16:00:16-20-11-2018	5
A02	Blois	14:11:01-20-11-2018	2
A03	Poitiers	13:05:22-20-11-2018	3
A01	Lyon	11:02:10-21-11-2018	4
A02	Blois	13:11:09-21-11-2018	8
A03	Poitiers	01:02:08-21-11-2018	5
...	....	....	.



Map reduce



Data landscape: past - today

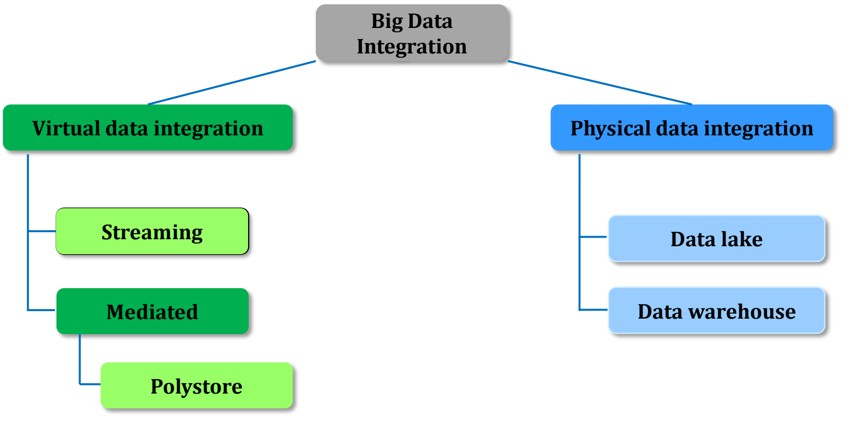
- Data models
  - relational
  - object-oriented
  - semi-structured
  - ...
- Data formats
  - numbers, dates, strings
  - ...
- Veolocity
  - OLTP systems

- Data models
  - relational
  - graphs
  - NoSQL
  - semi-structured
  - unstructured
  - ...
- Data formats
  - numbers, dates, strings
  - HTML, XML, JSON
  - time series and sequences
  - texts
  - multimedia
  - ...
- Veolocity
  - frequently changing (e.g., Facebook)
  - constantly changing (streams)

Data integration: past

- Virtual integration
  - federated
  - mediated
- Physical integration
  - ETL + data warehouse
- Common integration data model
  - relational
  - sometimes semistructured or object-oriented

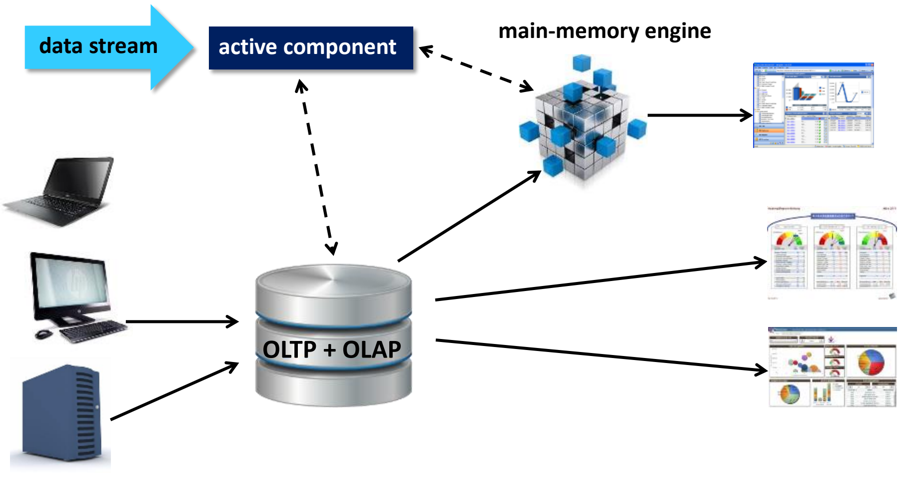
Big Data integration taxonomy



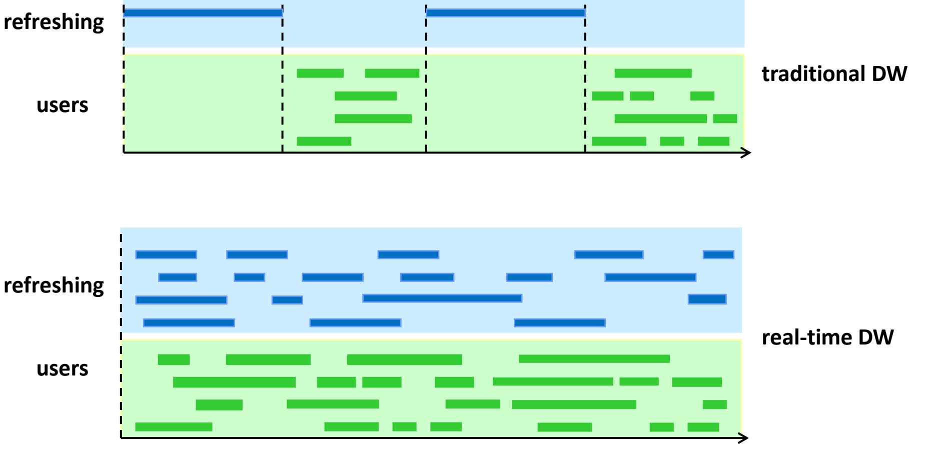
Types of processing

- **Offline**
  - **batch DW refreshing**
  - **analytics on stable data**
- **Real-time / near real-time**
  - **streaming of new data**
  - **analytics on the most up-to-date data up to the moment the query was sent**
  - **queries executed in (near) real-time**

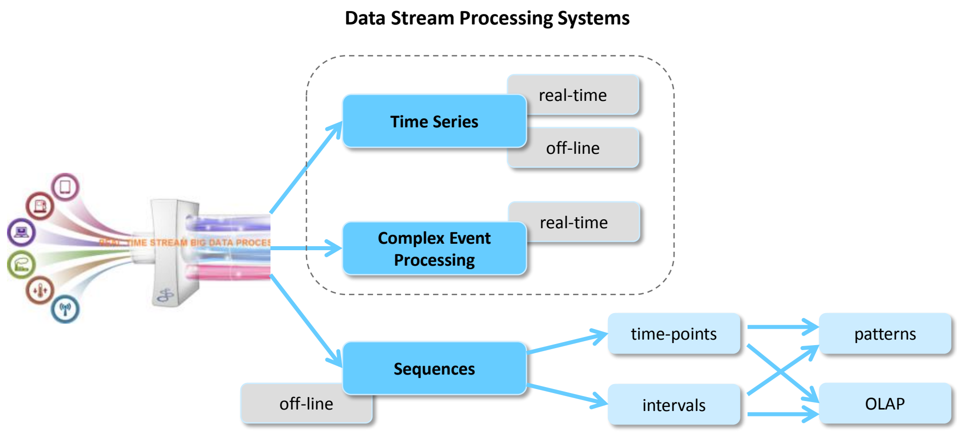
Real-time / Near R-T architecture



Real-time / Near real-time refreshing



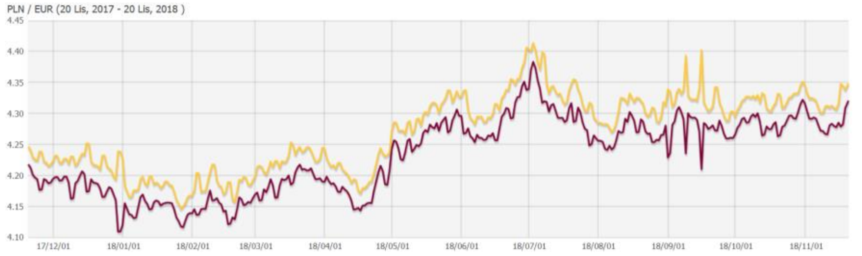
Streaming analytics



Time series

- A **time series** consists of values (elements, events) ordered by time
  - taken at successive **equally spaced points in time**
    - at a given frequency
  - variables of continuous domains
- Examples
  - signals from sensors
  - financial data
  - voice recording

Time series



- Time series analysis
  - Clustering and classification
  - Sequential pattern mining on discrete sequences
  - Searching for patterns
  - Aggregating in a sliding window
  - Trend analysis
  - Trend prediction
  - Finding similarities

Complex Event Processing

- **Analyzing TS to detect**
  - **outliers in a time window**
    - **e.g., a temperature 50% higher than avg in 10 min window**
  - **patterns**
    - **e.g., the 'W' pattern in stock quotes**

pattern: XYXYXY

1 1 9 9 **5 7 5 7 5 7** 0 0 3 **4 3 4 3 4 3** 8 6 7 4

Sequences

- A **sequence** consists of ordered values (elements, events) recorded with or without a notion of time
  - numerical properties (quantify an event)
  - text properties (describe an event)
- Point-based sequences
  - duration → instant
- Interval-based sequences
  - duration → interval
  - sequences of intervals

Sequences

- Sales reps performance interaction with a potential customer
  - pharma business
  - automotive business

meet → present → phone → meet → successful purchase

Sequences

➲ **Commuters' flow in a public transport infrastructure**

pass1: in → S1 → S2 → S3 → S4 → S5 → out

pass2: in → S8 → S9 → out

pass3: in → S3 → S4 → S5 → S6 → S7 → S8 → out

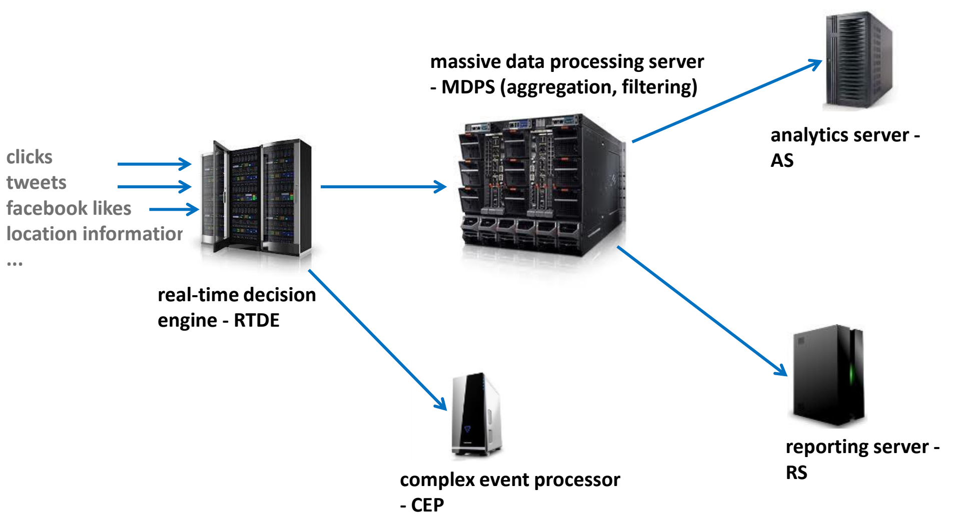
- **the number of passenger round-trips, e.g., S1 → S2 → S2 → S1, and their distributions over all origin-destination station pairs within 1st quarter of 2017**

Sequences

➲ **Other application examples**

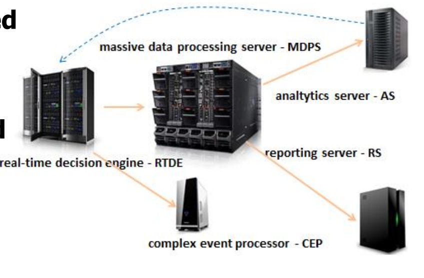
- **WEB logs analysis**
  - **sequence navigation between web pages:** product A page → competing product B page → competing product C page → product A page
- **identification of purchase patterns over time**
- **alarm logs**
- **money laundry scenarios**
- **infrastructure monitoring**
- **workflow management systems**
- **...**

Big Data architecture



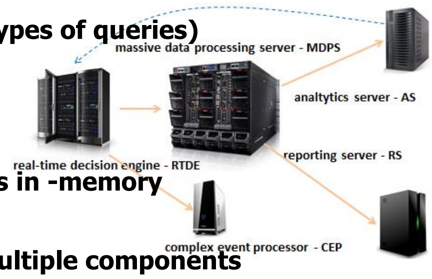
Big Data architecture

- **Scalability**
  - **RTDE - nb of events handled**
  - **MDPS - volume of data**
  - **data processing workload**
  - **AS - complexity of computation, workload of queries**
  - **RS - types of queries, nb of users**
  - **CEP - # events handled**
- **Type of data**
  - **RTDE - unstructured and semistructured (texts, tweets)**
  - **MDPS – semistructured, structured**
  - **AS - structured**
  - **RS - structured**
  - **CEP - unstructured and structured**



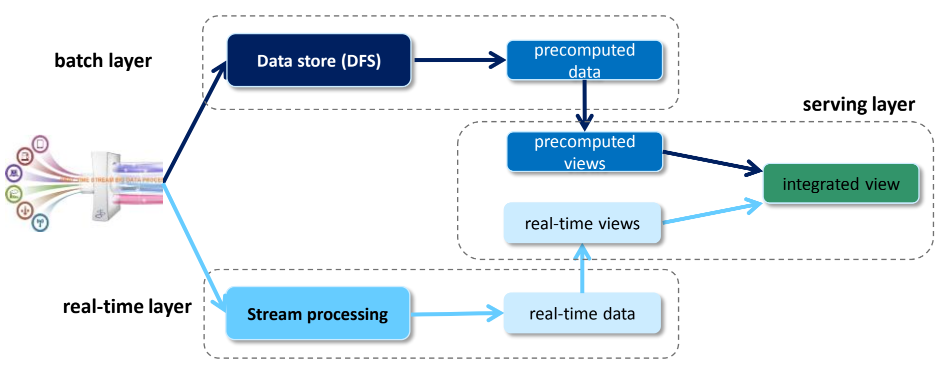
Big Data architecture

- **Workload**
  - **RTDE - high write throughput**
  - **MDPS - long-running data processing (I/O and CPU intensive): data transformations, ...**
  - **AS - I/O and CPU intensive**
  - **RS - compute intensive (various types of queries)**
- **Technologies**
  - **RTDE - key-value, in-memory**
  - **MDPS - Hadoop**
  - **AS, RS - columnar DBs, sometimes in -memory**
- **Conclusion**
  - **very complex architecture with multiple components**
  - **the need of integration**



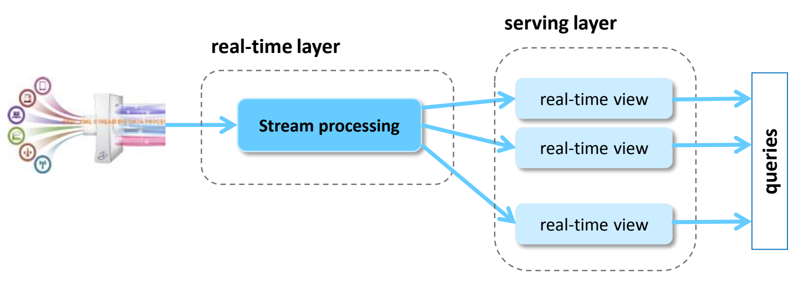
Lambda Architecutre

- **Batch layer → traditional DW processing**
- **Speed layer → stream processing**
- **Serving layer → integration of static and dynamic data by means of views**

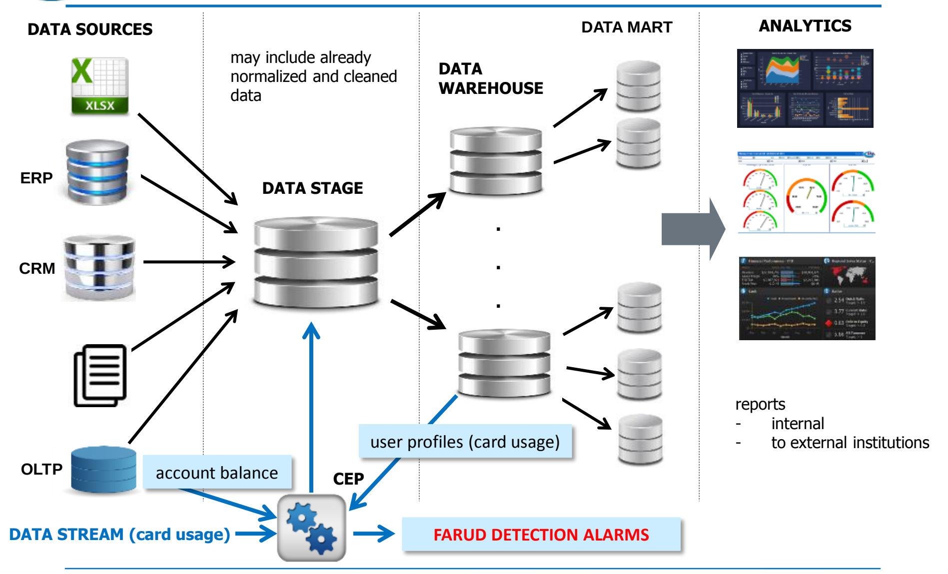


Kappa architecture

- ➲ **Processing streams of data**
- ➲ **Incoming data are streamed through a real-time layer and moved into a serving layer for queries**

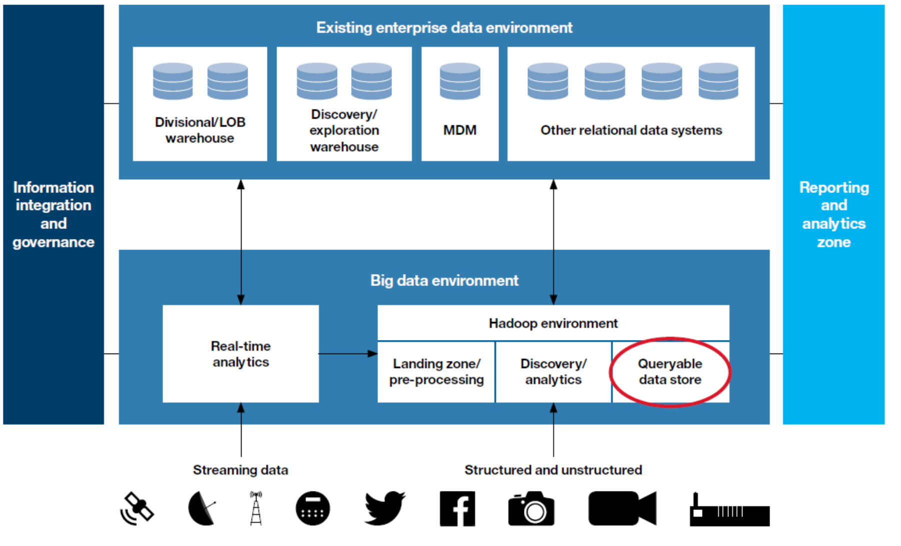


DW in a big bank

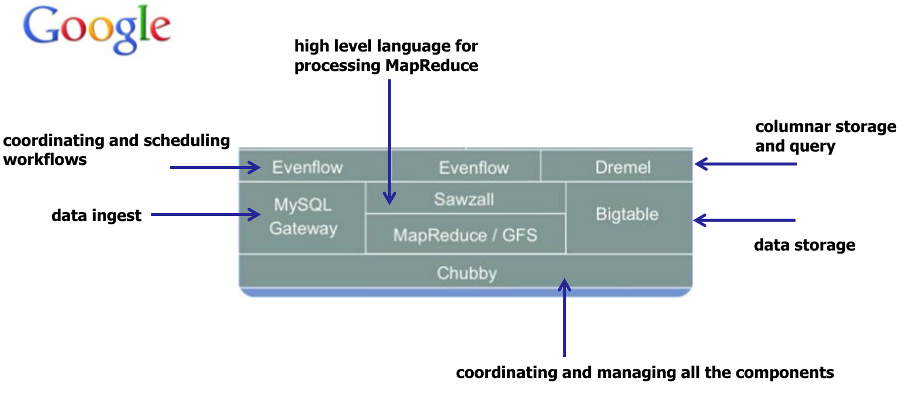


IBM architecture

- Data warehouse augmentation: the queryable data store. IBM software solution brief.



Big Data architecture



http://www.cloudera.com/content/cloudera/en/resources/library/training/apache-hadoop-ecosystem.html

Big Data architecture

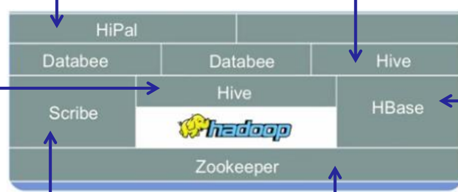
facebook

user interface to Hive

SQL-like language for data analysis (select, join, group by, ...)

high level language for processing MapReduce

columnar storage and query based on BigTable



web log loader (log scrapper), periodical loading into Hadoop aggregating log entries (for offline analysis)

coordinating and managing all the components, load balancing

Big Data architecture

YAHOO!

coordinating and scheduling workflows (Pig, Hive, Java, HDFS tasks)

RDB-like interface to data stored in Hadoop, metadata management

high level languages for processing MapReduce

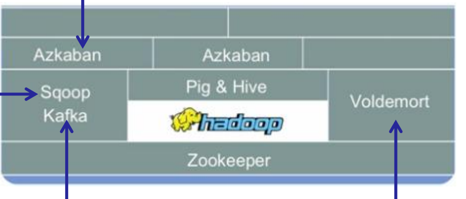
distributed web log loader (log scrapper), periodical loading into Hadoop (for offline analysis)

Big Data architecture

Linked in

workflow (batch job) scheduler
(e.g., data extraction, loading into Hadoop)

data ingest tool



queuing system,
distributed web log loader and
aggregator

table-like data storage + in memory
caching

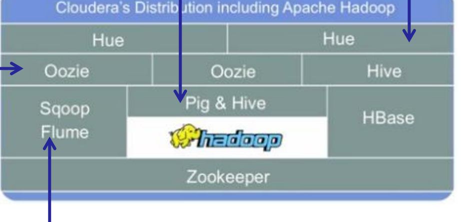
Big Data architecture

cloudera

UI for Hadoop (e.g., HDFS file browser, MapReduce job designer and browser, query interfaces for Hive, Pig, Impala, Oozie, application for creating workflows, Hadoop API)

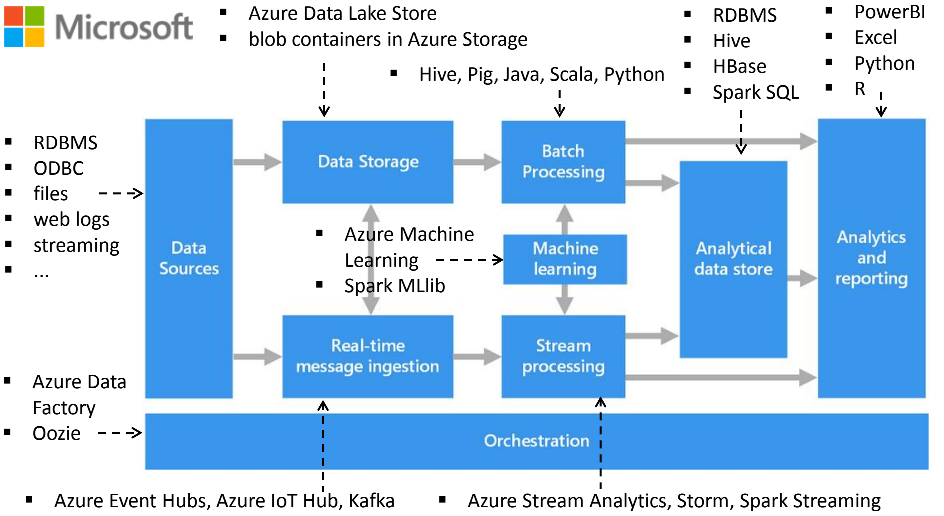
high level languages for
processing MapReduce

workflow coordination
and scheduling



moving data between systems,
distributed web log loader (log scrapper),
periodical loading into Hadoop (for offline
analysis)

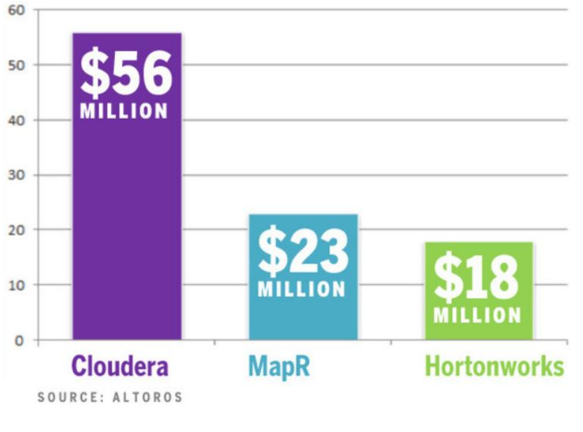
Big Data architecture



- https://docs.microsoft.com/en-us/azure/architecture/data-guide/big-data/index

Hadoop Distributions

➲ **Cloudera, MapR, Hortonworks, IBM, Pivotal Software**

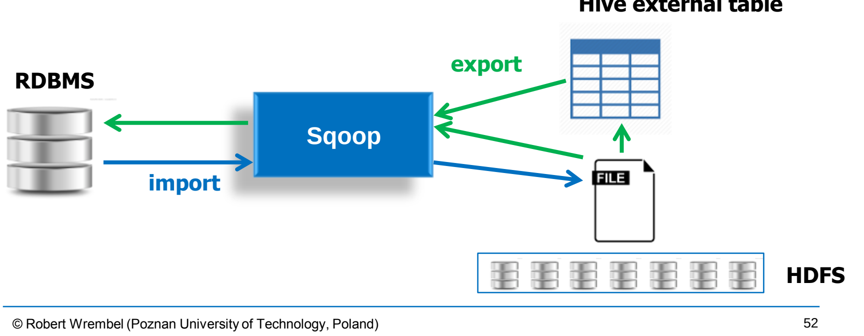


Data Ingest (ETL)

- Apache Sqoop: data movement RDBMS ↔ HDFS
- Apache NiFi: ELT tool
- DataTorrent RTS: ETL tool
- Apache Flume: moving data
- Apache Kafka: queuing system
- ...

Sqoop

- Tool for transferring data between
  - structured (relational DBs)
  - semi-structured (NoSQL)
  - unstructured (HDFS) data sources
- Connectors to DSs
  - FTP, JDBC, HDFS, Kafka, ...

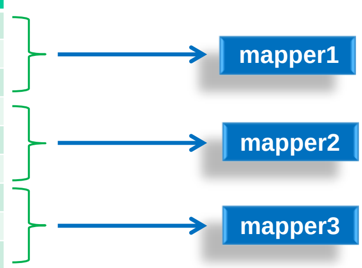


Sqoop

➲ **Parallel processing: n mappers**

```
sqoop import \
--connect jdbc:mysql://hostname/database_name \
--username user1 \
--password pass \
--table accounts \
--num-mappers 3
```

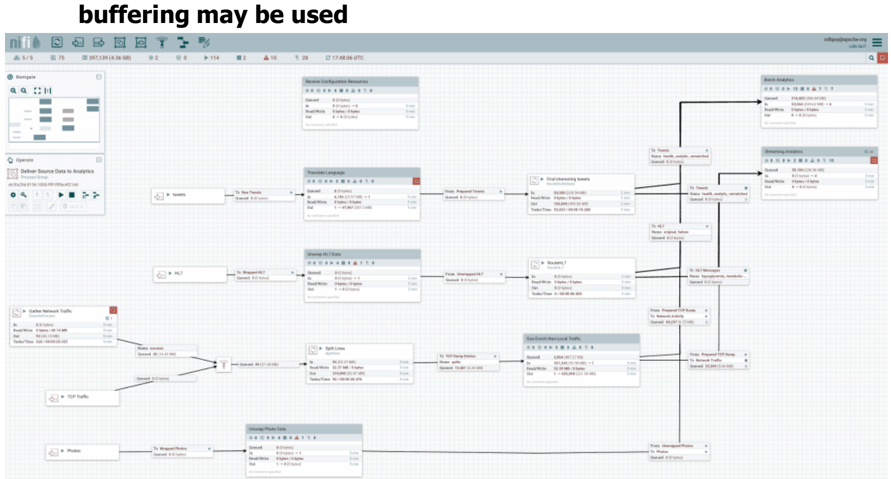
PK	attr1	attr2
1		
...		
100000		
100001		
...		
200000		
200001		
...		
300000		



NiFi

➲ **Purpose: to automate a flow of data between multiple systems → an ETL tool**

➲ **Asynchronous: for very high throughput and slow processing buffering may be used**

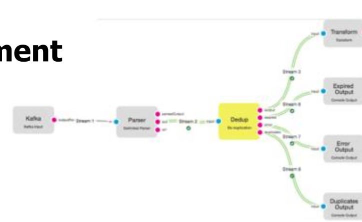


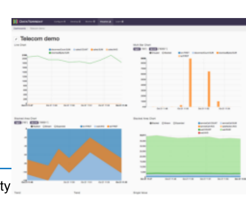
NiFi building blocks

- FlowFile
  - data moved within NiFi, represented as key-value
- Processor
  - processes FlowFiles
- Connection
  - connects processors by means of a queue → buffering
  - different processors may read from a queue at differing rates
- Flow Controller
  - acts as a broker facilitating the exchange of FlowFiles between processors
- Process Group
  - a set of processors and connections, which can receive data via input ports and send data out via output ports

DataTorrent RTS

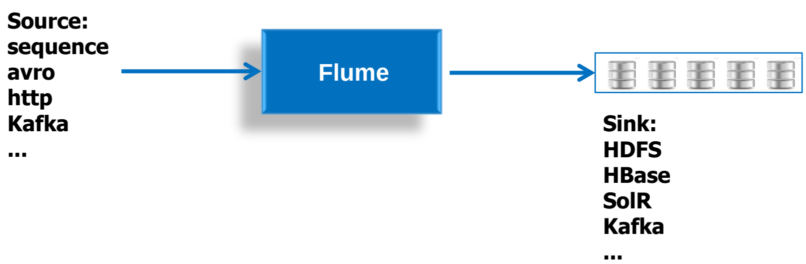
- Ingest from
  - HDFS, Kafka, Flume, flat files, JDBC
- Output into
  - HDFS, Hive, Cassandra, MongoDB, ...
- Data transformation operators
  - dedup, filter, split, sample, ...
  - parsing common formats such as XML, JSON, log files, syslog
- ETL-like development environment
- + data visualization





Flume

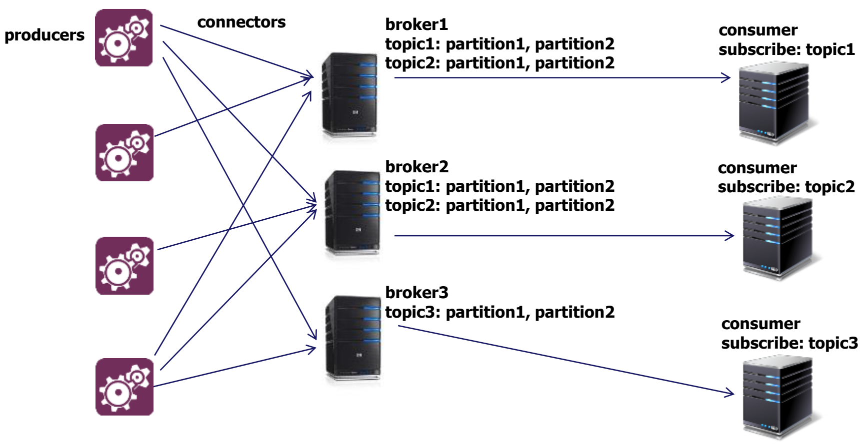
- Movng data between systems
- Ingesting, transforming, and storing



ETL for Big Data - Kafka

- Distributed queuing/messaging
- Processes streams of records: <key, value, timestamp>
- Terms
  - topic: stream of messages of a particular type, divided into partitions
  - producer: publishes a given topic
  - consumer: subscribes to one or more topics
  - broker: stores topics for their distribution to consumers

Kafka



- Multiple brokers (broker cluster) for load balancing

Kafka

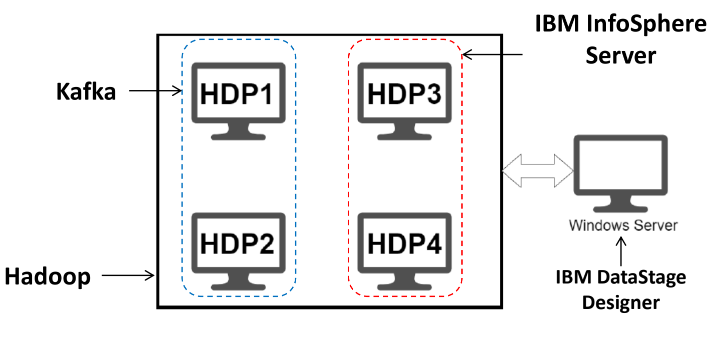
- Producer API: to publish a stream of records to one or more topics
- Consumer API: to subscribe to one or more topics
- Streams API: to process streams (e.g., aggregate)
- Connector API: to connect to input and outpud DSs
  - reads from: JDBC, NoSQL stores, Oracle Golden Gate, IBM Data Replication, Vertica, SolR, Twitter, ...
  - writes to: JDBC, HDFS, Amazon S3, SAP Hana, Vertica, NoSQL, Elasticsearch, SolR, Twitter, ...

Kafka

- Consumer maintains the info about read topic's partitions
- Broker deletes partitions after a given time period (configurable) regardless they have been read by a consumer or not → possible data loss
- At-least-once delivery model in the case of consumer failure
  - after restart a consumer may re-read the last topic's partition → duplicates
- The order of messages in a partition is preserved within a delivery
- The order of inter-partition delivery from different brokers is not preserved (e.g., read partition2 from broker3 then read partition1 from broker2)

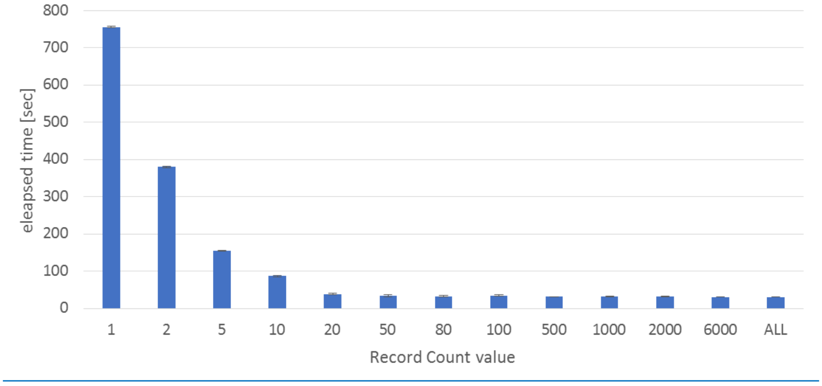
Kafka performance

- Architecture
  - pico-cluster: 4 nodes
  - node: 4-core CPU, 3GHz
  - 16GB RAM, 256GD HDD
  - Linux RedHat

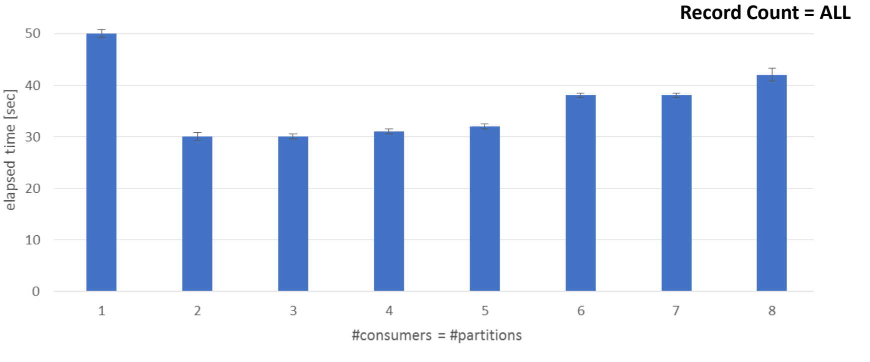


Kafka performance

- **Variable value of Record Count**
  - **parameter of a connector to Kafka from DataStage**
  - **# of rows read from a topic after which data processing in ETL begins**



Kafka performance



Stream processing frameworks

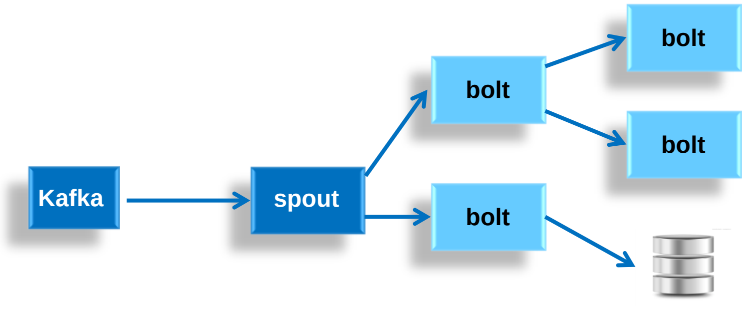
- Apache Storm
- Apache Flink
- Apache Kafka Streams
- Apache Spark Streaming
- Apache Samza: based on Hadoop Yarn and Kafka
- ...

Storm

- Real-time stream processing framework
  - implemented in: Clojure
  - apps in: Java, C#, Python, Scala, Perl, PHP
- Data structure: tuple - a list of coma separated values
- Stream: sequence of tuples
- Software components
  - connectors to: Twitter, queuing systems (e.g., Kafka)
  - spout: a source of a stream
  - bolt: a processing unit, accepts a stream, processes it, and outputs another stream (also to store it in a DB)
- Task: the execution of either a spout or bolt

Storm

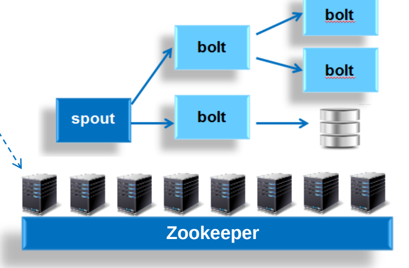
➲ Topology
- DAG composed of spouts and bolts connected by streams
- includes a single spout and an array of bolts



Storm

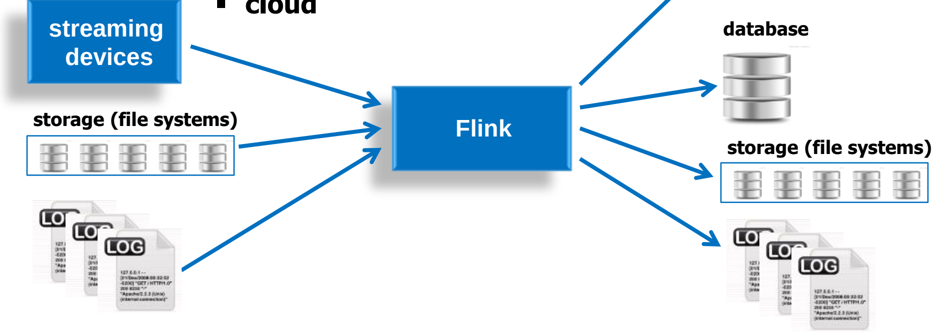
➲ Worker: topology runs in a cluster composed of multiple worker nodes
- Storm divides tasks evenly for all worker nodes
- a topology in a cluster is managed by Zookeeper

➲ Nimbus (master node)
- runs a topology
- distributes tasks to workers



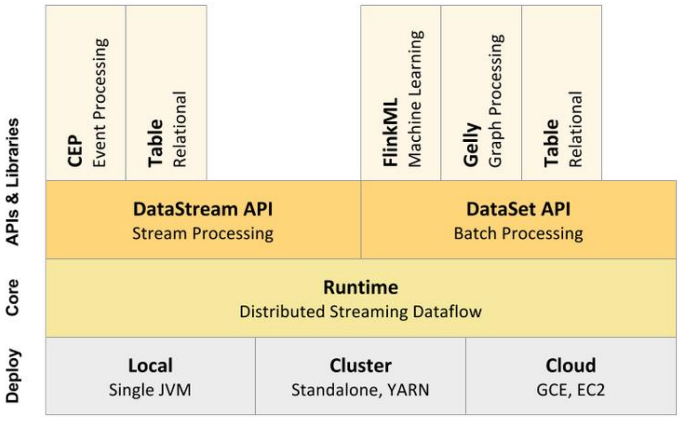
Flink

- Real-time stream processing framework
  - implemened in: Java, Scala
  - apps in: Java, Scala
- Deployment
  - cluster run by Yarn or Mesos
  - cloud



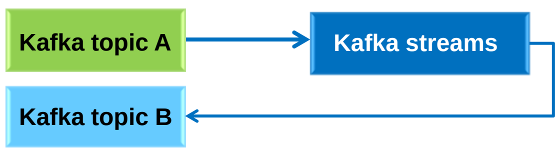
Flink

http://flink.apache.org/introduction.html



Kafka streams

- Java library
- Event-at-a-time processing
- Aggregation in a sliding window
- No built-in stream mining algorithms
- Deployment environment the same as for Kafka



Spark

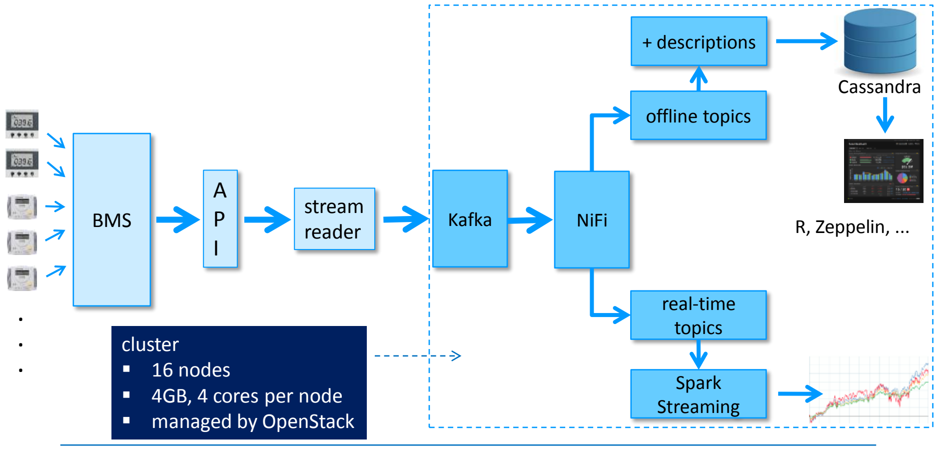
- In-memory data processing framework:
  - implemented in: Scala
  - apps in: Scala, Python, R, Java
- Deployable on
  - Hadoop, Apache Mesos, and EC2
  - independent cluster mode or cloud
- Can access
  - Amazon S3, HDFS, Cassandra, HBase, Hive, JDBC/ODBC, Twitter, ...

Spark

- Apps in: Java, Scala, Python
- Processing of n records at a time (micro-batches)
- Built-in
  - GraphX - a library of graph processing algorithms
  - MLib - a library of machine learning algorithms
  - Spark Streaming - stream processing engine (e.g., window functions)
  - Spark SQL - SQL-like querying structured data within Spark

Project@PUT

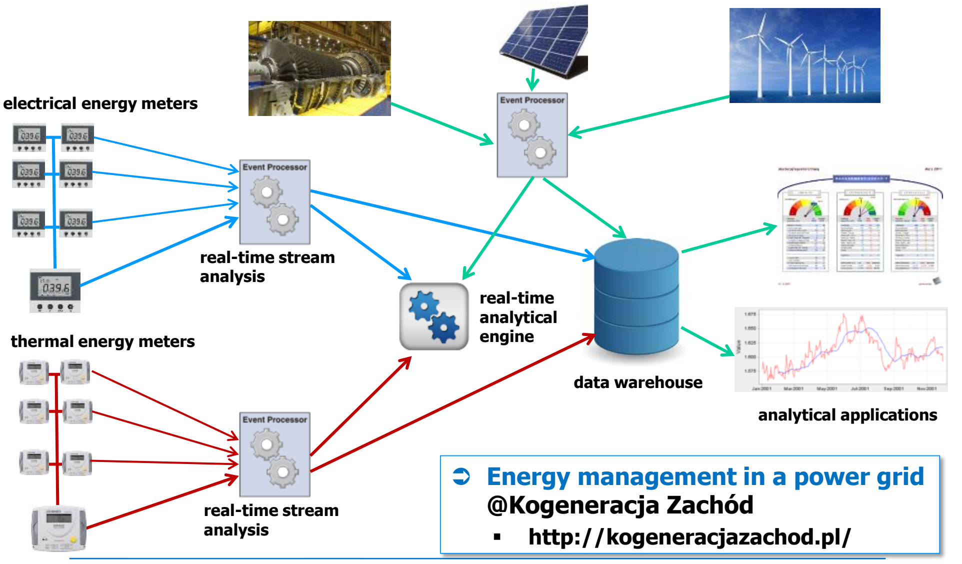
- Analyzing monitoring signals from a data center
  - Poznan Supercomputing and Networking Center (PSNC)
  - http://www.man.poznan.pl/online/en/



Project@PUT

- Processing
  - time series analysis
  - alarm signals → sequences → patterns
- Over 4400 different variables (signals) generated by BMS

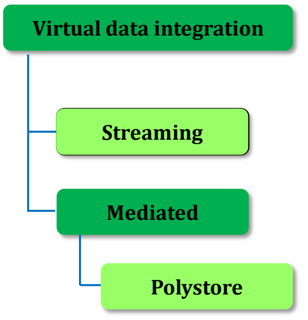
Project@PUT



- **Energy management in a power grid @Kogeneracja Zachód**
  - http://kogeneracjazachod.pl/

Mediated architectures

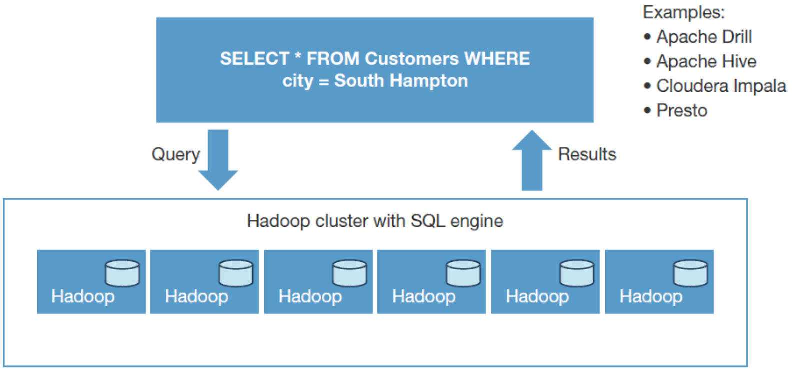
- Integration with Hadoop
- Polystore



Integration with Hadoop

- M.Gualtieri, B. Hopkins: SQL-For-Hadoop: 14 Capable Solutions Reviewed. Forrester, 2015

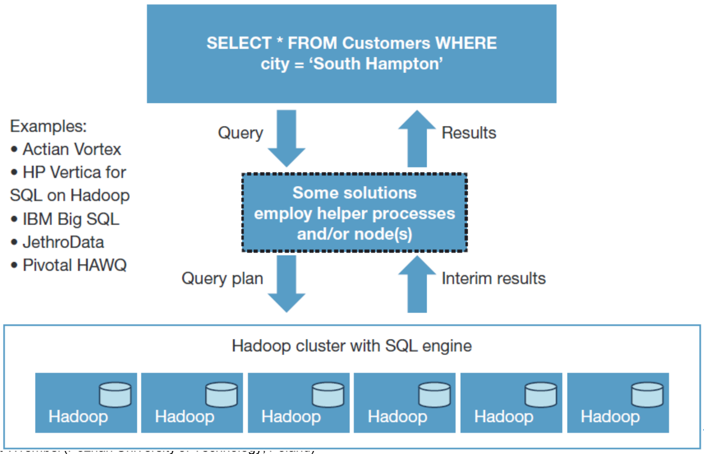
- **Pure SQL for Hadoop**



Integration with Hadoop

➲ Boosted SQL for Hadoop

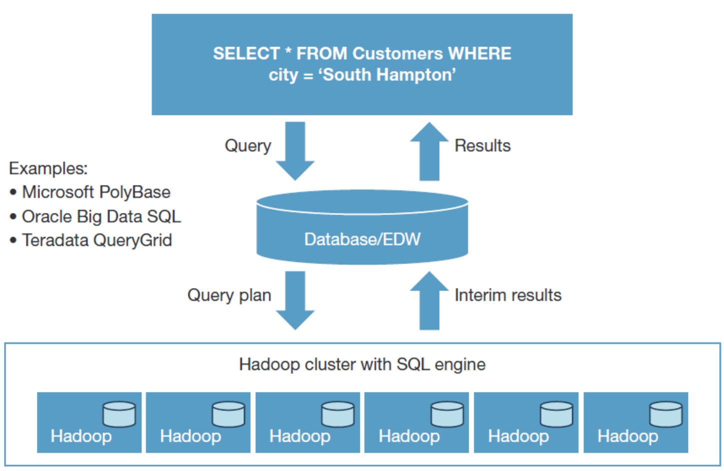
- typically include: query parser and optimizer
- require more strucutred data to exploit the power of SQL



Integration with Hadoop

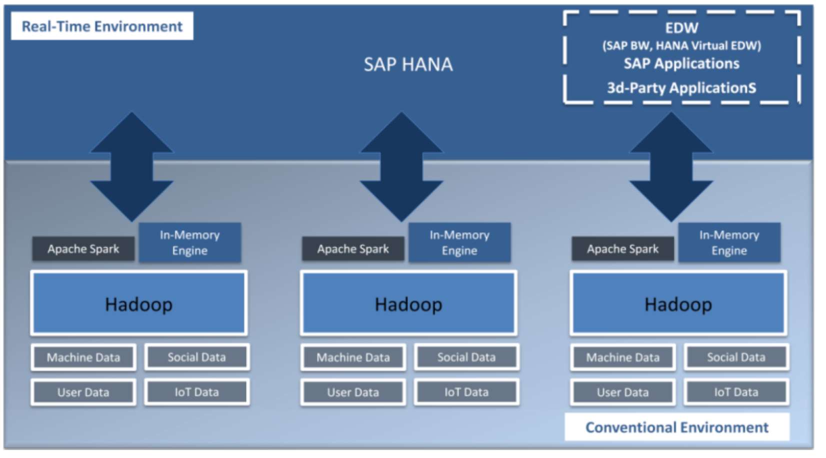
➲ Database + Hadoop

- Hadoop files accessed via external tables from a DB



Integration with Hadoop

➲ SAP Vora: HANA + Spark + Hadoop



➲ The Contextual Data Lake. By SAP, available at: https://tdwi.org/whitepapers/2015/10/the-contextual-data-lake.aspx

Integration with Hadoop

- ➲ **IBM BigInsights ⇨ Cloudera distribution + IBM custom version of Hadoop called GPFS**
- ➲ **Oracle BigData ⇨ appliance based on Cloudera for storing unstructured content**
- ➲ **Informatica HParser ⇨ to launch Informatica process in a MapReduce mode, distributed on the Hadoop servers**
- ➲ **Microsoft ⇨ dedicated Hadoop version for Azure**
- ➲ **EMC Greenplum, HP Vertica, Teradata Aster, SAP Sybase IQ ⇨ provide connectors directly to HDFS**

SQL interface to Hadoop

- Hadapt (currently Teradata)
  - platform for analytics on structured and unstructured data
  - hybrid storage: Hadoop + relational DB
- Teradata SQL-H
  - integrating Aster (sequence proc. engine) and Hadoop
- EMC HAWQ
  - integration of the Greenplum DBMS with Hadoop
- IBM BigSQL
  - part of InfoSphere BigInsights

SQL-like DBs on non-relational DSs

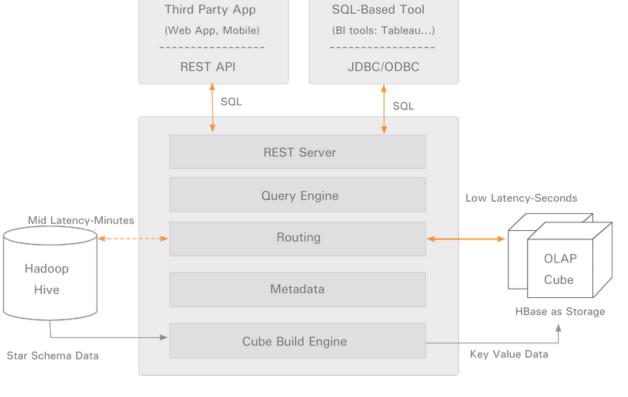
- Splice Machine
  - based on Apache Derby: Java-based ANSI SQL database
  - Derby (redesigned query optimizer to support parallel processing) on HBase (parallel processing) + Hadoop (parallel storage and processing)
- Apache Phoenix
  - relational-like DB on HBase
  - SQL interface
- MarkLogic
  - NoSQL database
- IBM Big SQL
  - a single point to query heterogeneous data stores: RDB, HDFS, NoSQL

SQL-like DBs on non-relational DSs

- **Virtuoso: data management system**
- **Storage engine for**
  - **relational, XML, RDF, text data**
- **Database functionality**
  - **querying (SQL, SPARQL)**
  - **indexing (also full text)**
  - **storage (row, column-store)**
  - **transaction management**
- **Connectors**
  - **ODBC/JDBC, SOAP, REST, HTTP, ...**

Big Data warehousing

- **Apache Kylin**



- http://kylin.apache.org/

Hadoop-based DWs

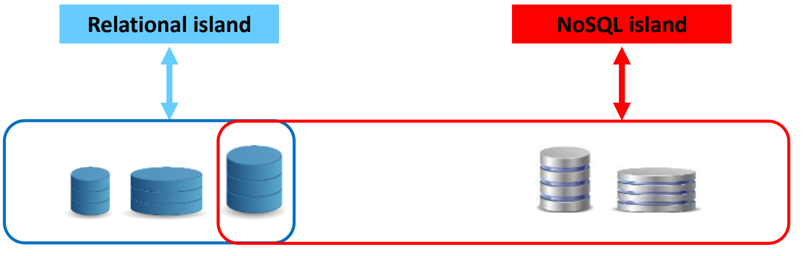
- ➲ **Cloudera Impala**
  - **SQL like query engine that runs on HDFS**
- ➲ **Apache Drill**
  - **SQL like query engine that runs on a data lake**
  - **schema discovery on the fly**

Complementary technologies

- ➲ **Stinger**
  - **in-memory graph analytics engine**
- ➲ **Spark GrapX**
  - **in-memory graph analytics engine**
- ➲ **Shark**
  - **based on Spark**
  - **query accelerator**
  - **uses HiveQL (SQL-like, translated into MapReduce jobs)**
- ➲ **Teradata SQL-H, EMC HAWQ, IBM BigSQL**
- ➲ **...**

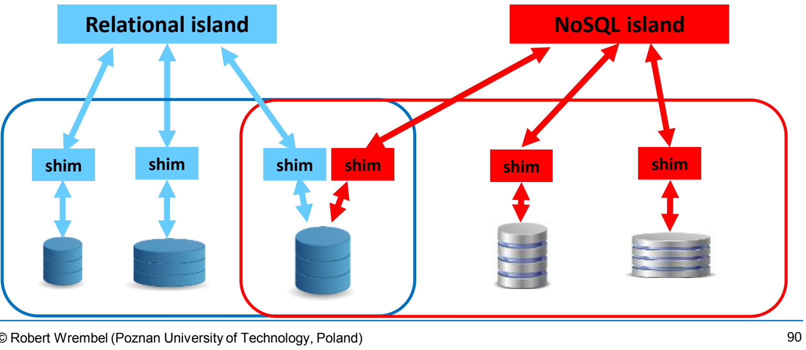
Polystore

- J. Duggan, A.J. Elmore, M. Stonebreaker, et. al.: The BigDAWG Polystore System. SIGMOD Record, Vol. 44, No. 2, 2015
- **Federation of islands of information**
- **Island of information**
  - **collection of storage engines accessed with a single query language**

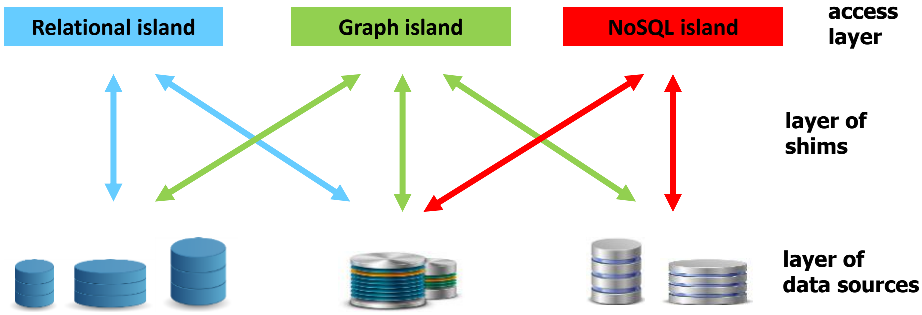


Island of information

- **Specifies a data model (seen by a user) → like mediator**
- **Provides a common query language (for a user)**
- **Includes a set of DMSs to manage data and execute queries**
- **Mapping the island's common language into a local one → shim → like wrapper**



Overall view

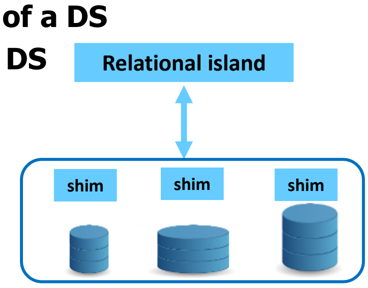


Single-island query

- Island query IQ (expressed in island's native language)
- IQ parsed into abstract syntax tree (AST)
- Decompose AST into partial queries for each DS in the island
- Shim translates the subquery into a query expressed in a language of a DS
- Subquery is executed by DS

user

DS



Multi-island query

- ➲ Scope: specifies in which island to exectue a query
- ➲ Cast: data transformation

query executed in the relational scope

rel. table

NoSQL collection

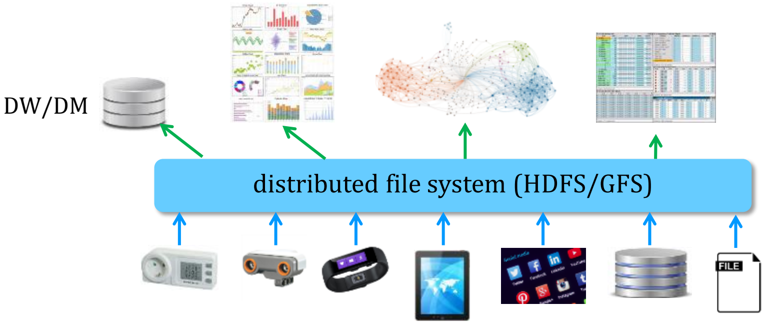
```
RELATIONAL
(select *
 from accounts, CAST(clients, RELATION)
 where accounts.clientID=clients.ID)
```


Query optimization issues

- ➲ Global query optimizer should have available
  - a cost model for each elementary operation in each data store
  - access to metadata (e.g., physical structures, data distribution) in each DS
- ➲ DSs are autonomous: the above information is unavailable
- ➲ GQO must use a "black box" approach and some rules
  - one island queries → move operations to data
  - multi-island queries → gather execution statistics
    - performance of each DS for a given partial query
    - possible to move data to another island for a more efficient execution → data allocation problem

Data Lake

- **Physical integration**
  - **a repository that stores a vast amount of raw data in its native format until it is needed**
  - **typically based on a distributed file system (HDFS, GFS)**



Data Lake

- **Content**
  - **relational tables**
  - **WEB tables**
  - **XML**
  - **texts**
  - **images, sounds, videos**
  - **graphs**
  - **time series**
  - **... any existing format**
- **Each data element in a DL should have assigned a unique identifier and tagged with a set of metadata**

Data Lake

- ➲ **No schema on write**
  - **schemas of data are not defined (considered) while writing into a data lake**
- ➲ **A schema is obtained when data are queried → schema on read**
  - **the need to understand the content → metadata**

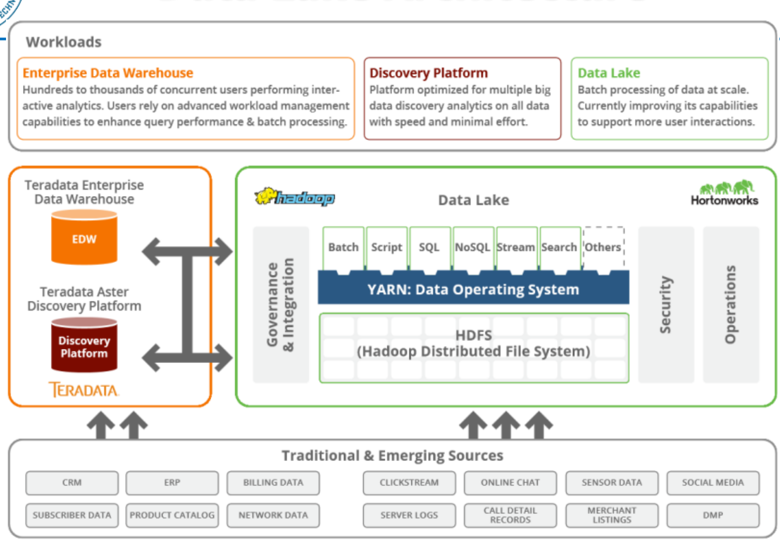
Data Lake

- ➲ **Querying a data lake**
  - **a query language and query engine capable of expressing and processing a query, possibly expressed in a natural language**
  - **finding relevant data sources for a query**
    - **relevant "schema"/structure**
    - **relevant content**
    - **correlating multiple data sources of the same semantics**
    - **selecting the most reliable DS**
    - **managing data quality of DSs**
  - **finding the relevant data sources quickly**
  - **metadata on**
    - **which DS was used to answer a given query**
    - **quality of DSs**

Data Lake

- Querying a data lake
  - efficiently **retrieving subsets of data** for a query
    - data of high quality
  - **transforming data on the fly** (during a query execution) into a common format
  - integrating data on the fly
  - choosing appropriate ways of **visualizing** the results
  - scalability → performance
- Need for refreshing
  - how to detect changes?
  - new algorithms for incremental refreshing?
  - even incremental refreshing uploads large volumes of data

Data Lake Architecture



- Image taken from: Putting the Data Lake to Work - a Guide to Best Practices. CITO Research, April 2014

ETL for Big Data

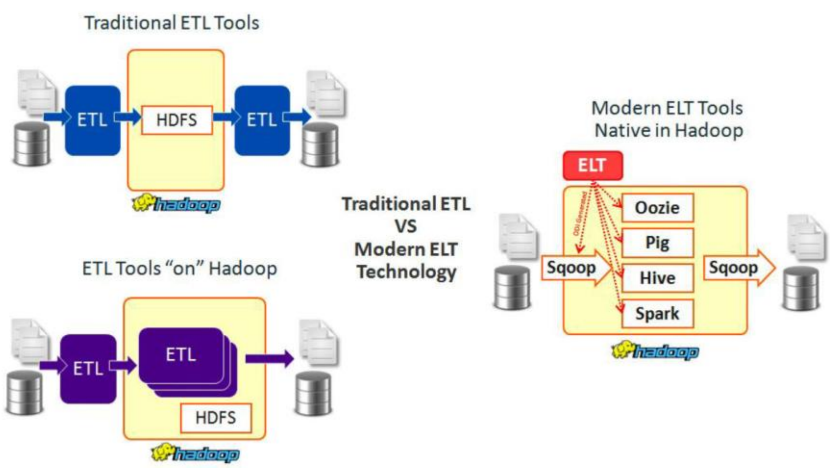
input
- **streams (sequences, time series)**
- **images**
- **sounds**
- **videos**
- **texts (unstruct., semi-struct.)**

periodically uploaded

E(T)L

data lake

ETL



➲ The Five Most Common Big Data Integration Mistakes To Avoid. ORACLE white paper, 2015

Metadata

- **Extensive usage of metadata**
  - **schema/structure**
    - **semantics of properties**
  - **content**
    - **semantics of values**
  - **transformation rules**
  - **visualization**
  - **performance**
- **Data annotation during E(T)L**
- **Data profiling in a data lake**
- **Incremental maintenance of metadata**
- **Metadata standard?**
  - **CWM for relational systems**
  - **? for data lakes**

Big Data integration challenges (1)

- **Panel discussion**
  - Int. Conference on Conceptual Modelling (ER), Gifu, Japan, 2016
  - Big Data and Conceptual models: Are they mutually compatible?
- **How to (semi)-automatically discover data sources?**
  - **DS structure discovery**
  - **DS content understanding**
- **How to dynamically plug-in a DS into a federation?**
- **How to construct an integrated conceptual model?**
- **What integration data model to use?**

Big Data integration challenges (2)

- Global **query processing**?
  - parsing, decomposing, translating into native dialects, and routing
  - global query optimization → cf. polystore
- How to **integrate on the fly** (transform, clean, deduplicate, integrate) data returned by local queries?

Big Data integration challenges (3)

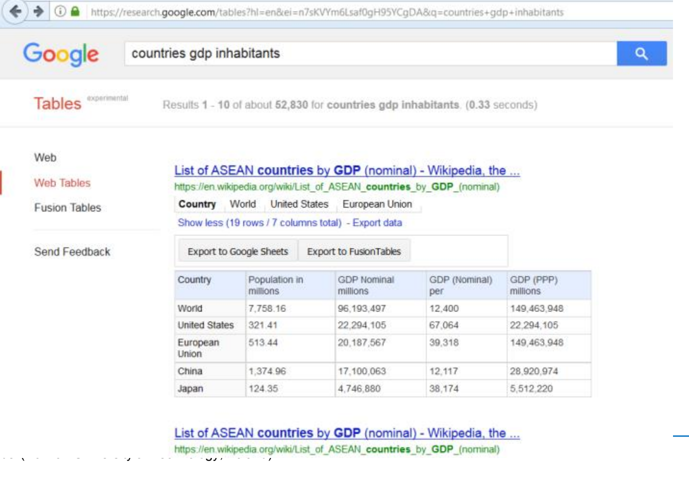
- Performance optimization
  - caching some results
  - what to cache?
  - where to store (RAM only vs. disk)?
  - how to manage the cache (removing/adding data)?
  - from which DSs to cache (slowly changing vs. rapidly changind DSs)
  - when and how to refresh cached data
  - using cached data in queries
  - prefetching
  - fast reads (fast search - indexing, compression)
  - fast writes (storage format, compression)

Big Data integration challenges (4)

- **New ways of querying**
  - **fusion tables**
- **User interface and visualization**
  - **user prefs: graphs, tables, ...**
  - **multiple (different) schemas needed for multiple users → single query language? → natural language?**
- **Conceptual modeling for data warehouses**
  - **facts and dimensions in XML, Graph, NoSQL → already ongoing research**

Querying the Web

- **Web Tables**: https://research.google.com/tables?hl=en
- **Fusion Tables**: to create and populate your own table based on web tables



Country	Population in millions	GDP Nominal millions	GDP (Nominal) per	GDP (PPP) millions
World	7,758.16	96,193,497	12,400	149,463,948
United States	321.41	22,294,105	67,064	22,294,105
European Union	513.44	20,187,567	39,318	149,463,948
China	1,374.96	17,100,063	12,117	28,920,974
Japan	124.35	4,746,880	38,174	5,512,220

Twitter applications

- **Analyzing Twitter posts**
  - **Google flu trend maps**
    - http://www.slate.com/blogs/moneybox/2014/04/04/twitter_economics_university_of_michigan_stanford_researchers_are_using.html
    - "Google tracks flu activity around the world by monitoring how many people are searching flu-related terms in different areas, and with what frequency. "We have found a close relationship between how many people search for flu-related topics and how many people actually have flu symptoms""
  - **tweets on unemployment well correlate with real governmental data**
    - http://www.washingtonpost.com/blogs/wonkblog/wp/2014/05/30/the-weird-google-searches-of-the-unemployed-and-what-they-say-about-the-economy/

RDBMS vs. BData technologies: the Future?

- **TechTarget: Relational database management system guide: RDBMS still on top**
  - http://searchdatamanagement.techtarget.com/essentialguide/Relational-database-management-system-guide-RDBMS-still-on-top
  - "While NoSQL databases are getting a lot of attention, relational database management systems remain the technology of choice for most applications„
- **S. Ghandeharizadeh: SQL, NoSQL, and Next Generation Data Stores**
  - keynote talk at DEXA 2015
  - RDBMS will be important components of IT infrastructures

RDBMS vs. BData technologies: the Future?

➲ **R. Zicari: Big Data Management at American Express**

- Interview with Sastry Durvasula and Kevin Murray. ODBMS Industry Watch. Trends and Information on Big Data, New Data Management Technologies, and Innovation. Oct, 2014, available at: http://www.odbms.org/blog/2014/10/big-data-management-american-express-interview-sastry-durvasula-kevin-murray/
- "The Hadoop platform indeed provides the ability to efficiently process large-scale data at a price point we haven't been able to justify with traditional technology. That said, not every technology process requires Hadoop; therefore, we have to be smart about which processes we deploy on Hadoop and which are a better fit for traditional technology (for example, RDBMS)."–Kevin Murray.

RDBMS vs. BData technologies: the Future?

➲ **The Contextual Data Lake**

- by SAP, https://tdwi.org/whitepapers/2015/10/the-contextual-data-lake.aspx
- "... companies will retain an EDW as part of their overall data architecture ..."

RDBMS

- ➲ Conceptual and logical modeling methodologies and tools
- ➲ Rich SQL functionality
- ➲ Query optimization
- ➲ Concurrency control
- ➲ Data integrity management
- ➲ Backup and recovery
- ➲ Performance optimization
  - buffers' tuning
  - storage tuning
  - advanced indexing
  - in-memory processing
- ➲ Application development tools

NoSQL

- ➲ Flexible "schema" ⇨ suitable for unstructured data
- ➲ Massively parallel processing
- ➲ Cheap hardware + open source software

- ➲ Choosing the right NoSQL database for the job: a quality attribute evaluation. Journal of Big Data; http://www.journalofbigdata.com/content/2/1/18