Recap Spark

- Distributed Resource Management: HDFS, Yarn (difference to HPC schedulers)
- MapReduce Patterns:
  - Summarization
  - Join Patterns: Which join to use when?
- RDD Abstraction for In-Memory Computing - Higher-Level MapReduce inspired API
- Spark Application: Job, Stage, Task
- Distributed Performance:
  - Managing shuffle traffic is important
  - Use of combiners recommended!

Spark Combiners

- groubByKey does not use combiners
- reduceByKey uses combiners by default (Careful: Associativity)
- Custom combiners possible

reduceByKey(*func, numPartitions=None, partitionFunc=<function portable_hash at 0x7fc35dbc8e60>*)

Merge the values for each key using an associative and commutative reduce function.

This will also perform the merging locally on each mapper before sending results to a reducer, similarly to a "combiner" in MapReduce.

Output will be partitioned with **numPartitions** partitions, or the default parallelism level if **numPartitions** is not specified. Default partitioner is hash-partition.

```
>>> from operator import add
>>> rdd = sc.parallelize([("a", 1), ("b", 1), ("a", 1)])
>>> sorted(rdd.reduceByKey(add).collect())
[('a', 2), ('b', 1)]
```

Further Frameworks: Flink

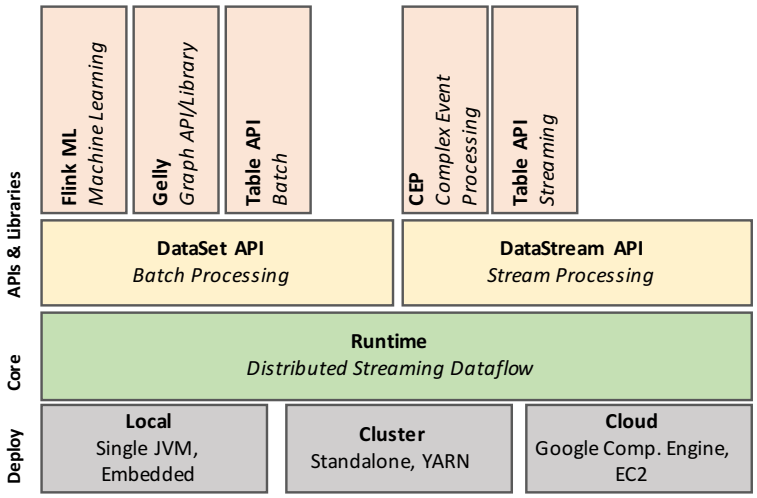


Figure 1: The Flink software stack.

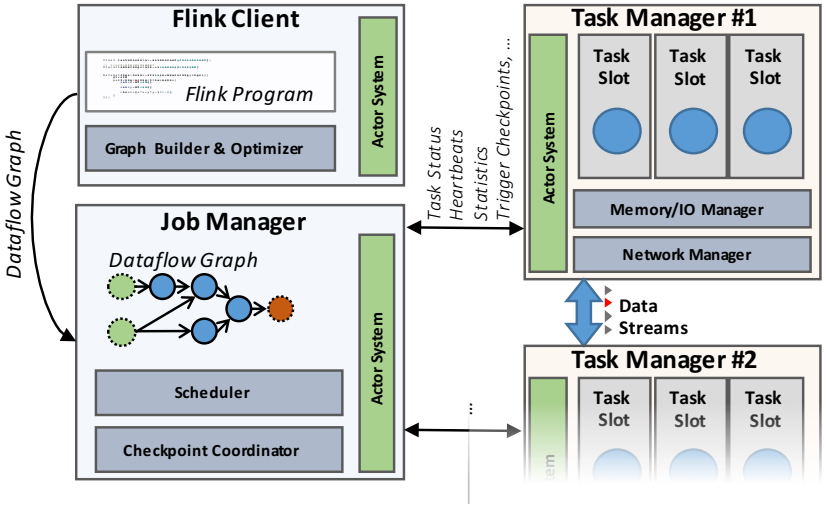
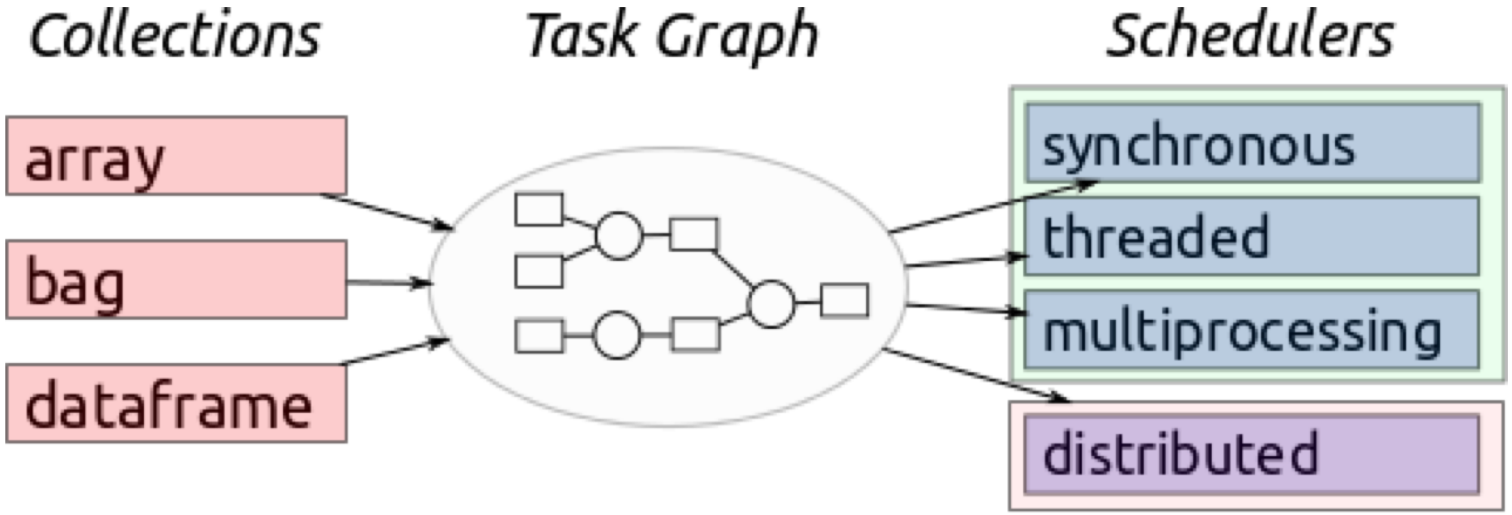


Figure 2: The Flink process model.

Flink: Stateful computations over streams real-time and historic fast, scalable, fault tolerant, in-memory, event time, large state, exactly-once

Further Frameworks: Dask



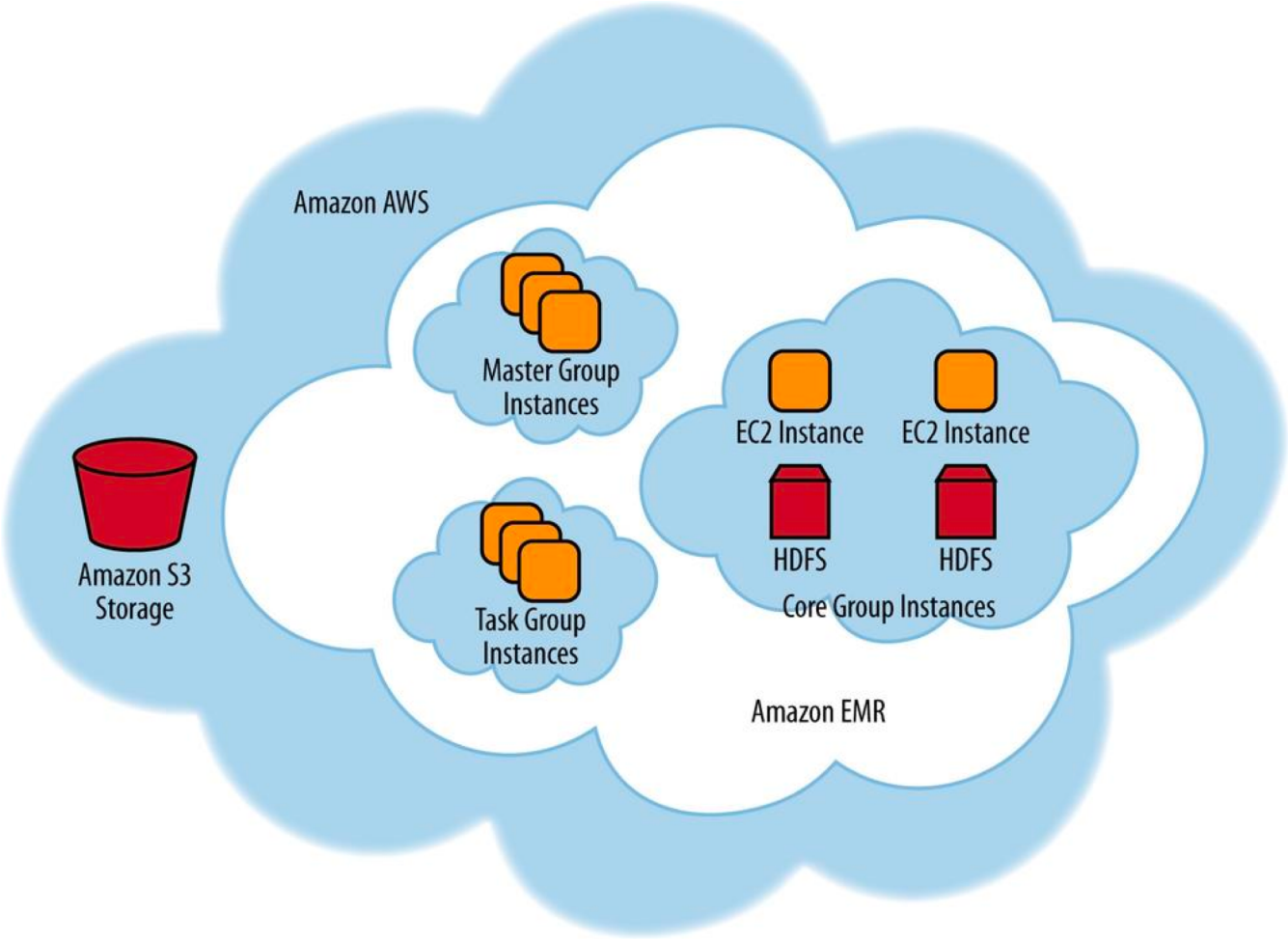
1.Dynamic task scheduling optimized for computation. This is similar to *Airflow, Luigi, Celery, or Make*, but optimized for interactive computational workloads.
2.“Big Data” collections like parallel arrays, dataframes, and lists that extend common interfaces like *NumPy, Pandas, or Python iterators* to larger-than-memory or distributed environments. These parallel collections run on top of the dynamic task schedulers.

https://dask.pydata.org/en/latest/

- Dean et al., MapReduce: Simplified Data Processing in Large Clusters, 2004
- Zaharia et al., Spark: Cluster Computing with Working Sets, Hotcloud 2010.
- Zaharia et al., Resilient Distributed Datasets: A Fault-Tolerant Abstraction forIn-Memory Cluster Computing, https://www.usenix.org/system/files/conference/nsdi12/nsdi12-final138.pdf, NSDI, 2012
- Carbone et al., Apache Flink: Stream and Batch Processing in a Single Engine, IEEE Engineering Bulletins, 2015
- Rocklin, Dask: Parallel Computation with Blocked algorithms and Task Scheduling, Proceedings of the 14th Python in Science Conference, 2015

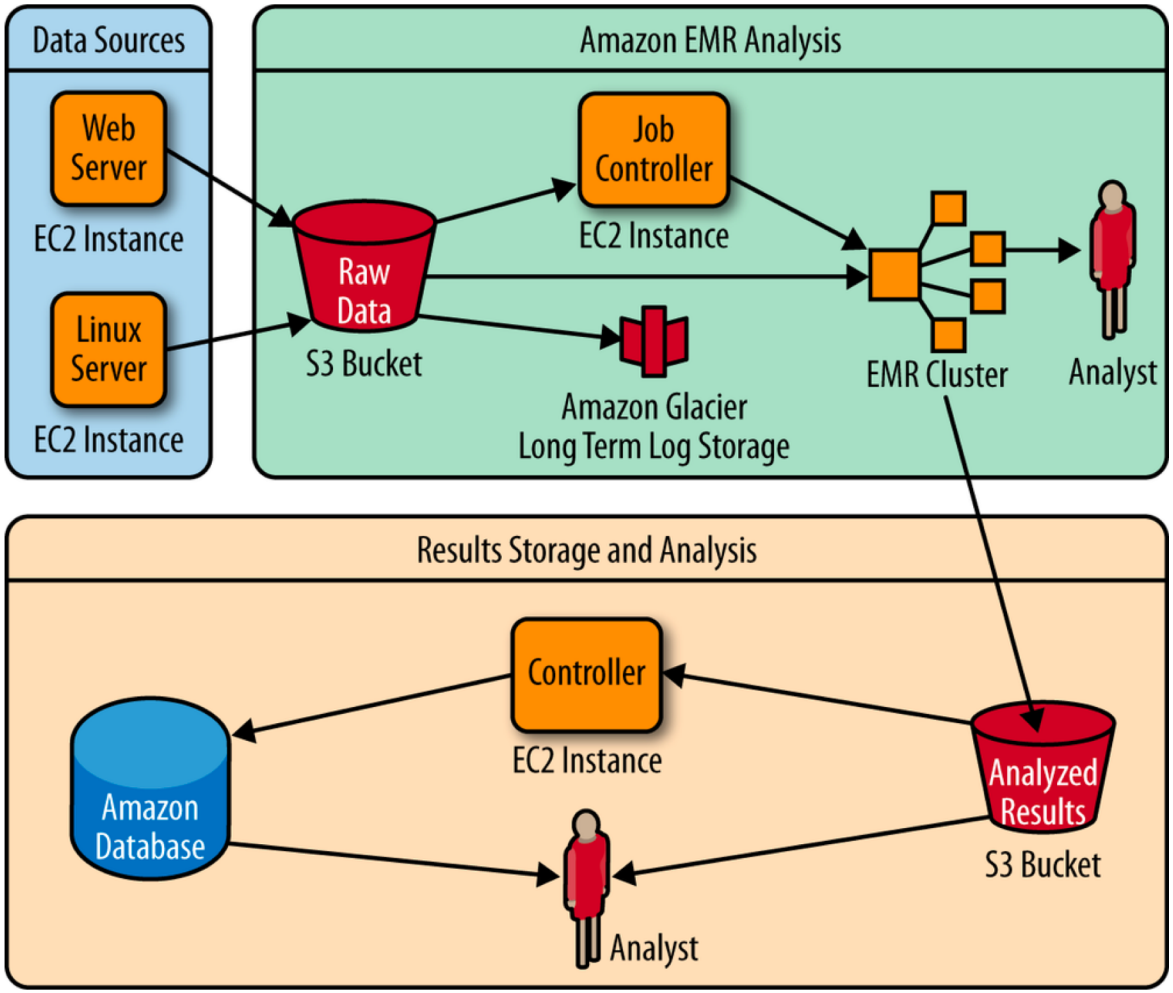
- Hadoop can be installed on most cloud services on their cores infrastructure services:
  - AWS EC2
  - Azure VMs
  - Google Compute Engines
- Hadoop Filesystem Adaptors for various object stores exists:
  - Simple Storage Service (S3)
  - Azure Blob Storage
  - Google Cloud Storage
- All three cloud vendors provided managed Haddop clusters:
  - Amazon Elastic MapReduce
  - Azure HDInsights
  - Google Cloud DataProc

Hadoop in the Cloud: Amazon Elastic MapReduce



Christopher Philips, Kevin Schmidt, Programming Elastic MapReduce, O'Reilly, 2013.

Hadoop in the Cloud: Amazon Elastic MapReduce



Christopher Philips, Kevin Schmidt, Programming Elastic MapReduce, O'Reilly, 2013.

Big Data on Different Infrastructure.

	On-Premise	HPC	Private	Public
Architecture	Dedicated Hardware	Dedicated Hardware managed by HPC scheduler (SLURM, PBS)	Manually setup VM clusters	Fully-managed VM Cluster (manual setup possible)
Maturity	High	Medium	Medium	Medium
Flexibility/ Extensibility/ Elasticity	Low	Low	Low	High (cluster can be dynamically expanded/ shrunk)
Performance	Optimal	Optimal	Good	Good
Security	High	High	High	Medium
Management Tools	Ambari, Cloudera Manager	saga-hadoop, jumpp	Apache Whirr, Cloudbreak	AWS Elastic MapReduce, Azure HDInsights

SQL on Hadoop

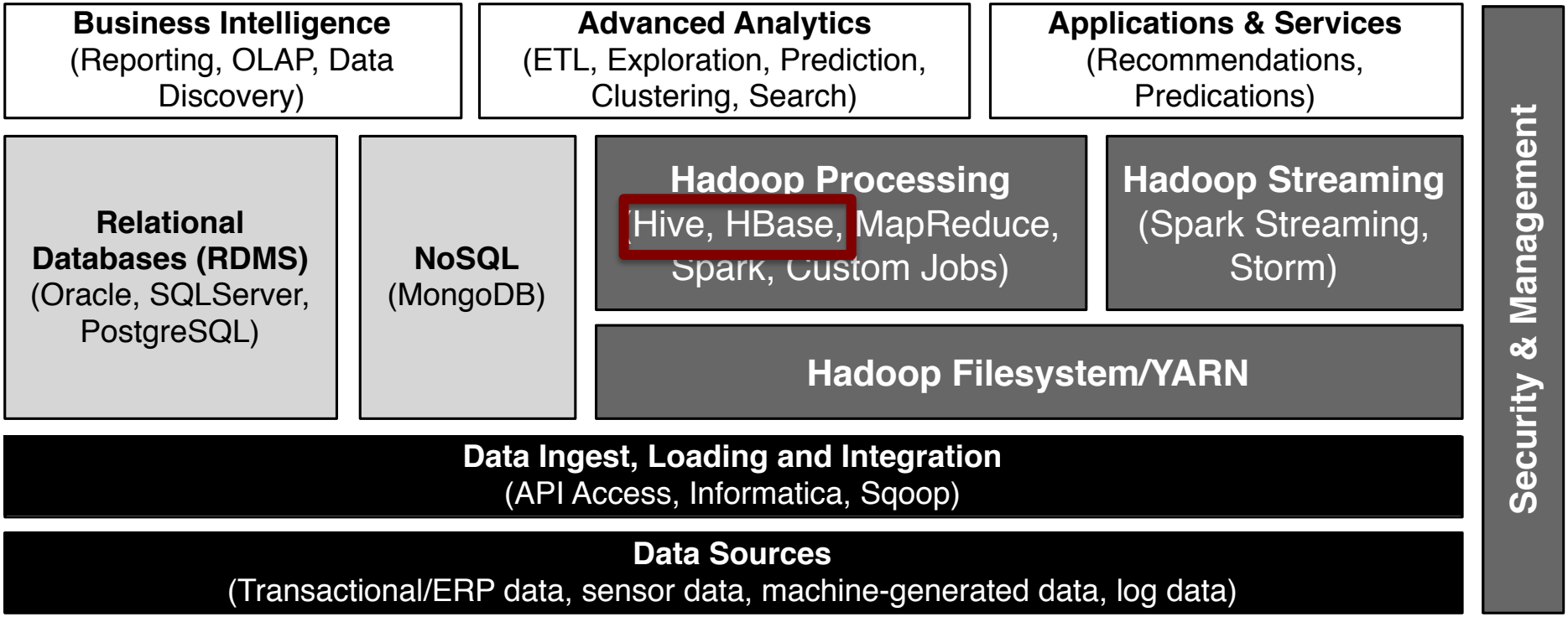
Pramod: Very well said, RDBMS and NoSQL are used for an entirely different class of Applications. Definitely for Transactions its RDBMS. Looking back on your career, what has been the most satisfying part of your technological journey?

Don Chamberlin: It's very gratifying to me that, 43 years after Ray Boyce and I published the first SEQUEL paper, SQL is still the world's most widely used database query language. I'm also very gratified to see that SQL is being used in a vast variety of exciting applications, far more than Ray and I ever anticipated. When we started work on SQL, the internet did not exist, but nevertheless, SQL has turned out to be an essential tool for commerce on the internet. Several open-source SQL systems are available, so anyone who wants to can use SQL for free.

In developing SQL, Ray Boyce and I were standing on the shoulders of giants: Turing Award winners Charlie Bachman, who invented the concept of an integrated database system, and Ted Codd, who invented the relational model of data. What Ray and I were able to do was to couch some of these technical ideas in plain English terms that were easy for everybody to understand. I'm grateful for the work of the System R team at IBM and the Ingres team at U.C. Berkeley, who built the first prototype relational database systems, to Larry Ellison at Oracle, who was the first to establish the market for SQL systems, and to Jim Melton, who dedicated his career to creating and editing the international SQL standard. It has been a privilege for me to work with all these people.

https://www.mappingthejourney.com/single-post/2017/10/12/episode-11-interview-with-don-chamberlin-designer-of-sql-database-language/

Structure Data Processing on Hadoop



References for Material

Material for this lecture has been taken from:

- Grover, Hadoop Application Architecture, O'Reilly, 2014
- Alan Beaulieu, Learning SQL, O'Reilly, 2009
- Jason Rutherglen, Programming Hive, O'Reilly, 2012
- Lars George, HBase – The Definitive Guide, O'Reilly, 2011

Schema on Read vs. Write

- A **schema** is a shared consensus about some universe. Any schema that does emerge will change frequently. Data found in the real world will typically not conform to any schema.
- **Hadoop's Schema-on-Read model does not impose any requirements when loading data into Hadoop.** Data can be simply ingested into HDFS by one of many ways without having the need to associate a schema or pre-process the data before processing it.
- **Schema on-read does not mean we cannot use higher-level abstractions such as SQL.**

Source: Bill Howe, Data Science Curricula, http://de.slideshare.net/billhoweuw/data-science-curricula-at-

Relational Data Modelling

Customer

cust_id	fname	lname
1	George	Blake
2	Sue	Smith

Account

account_id	product_cd	cust_id	balance
103	CHK	1	$75.00
104	SAV	1	$250.00
105	CHK	2	$783.64
106	MM	2	$500.00
107	LOC	2	0

Product

product_cd	name
CHK	Checking
SAV	Savings
MM	Money market
LOC	Line of credit

Transaction

txn_id	txn_type_cd	account_id	amount	date
978	DBT	103	$100.00	2004-01-22
979	CDT	103	$25.00	2004-02-05
980	DBT	104	$250.00	2004-03-09
981	DBT	105	$1000.00	2004-03-25
982	CDT	105	$138.50	2004-04-02
983	CDT	105	$77.86	2004-04-04
984	DBT	106	$500.00	2004-03-27

Alan Beaulieu, Learning SQL, O'Reilly, 2009

- Relational Abstraction:
  - Everything is a table
  - Every row in a table represents a record
  - (in the strict formulation of the relational model it has the same columns)

- Structured Query Language (SQL): domain-specific language for manipulating the data in relational tables:
  - DDL: Data Declaration Language
  - DML: Data Manipulation Language

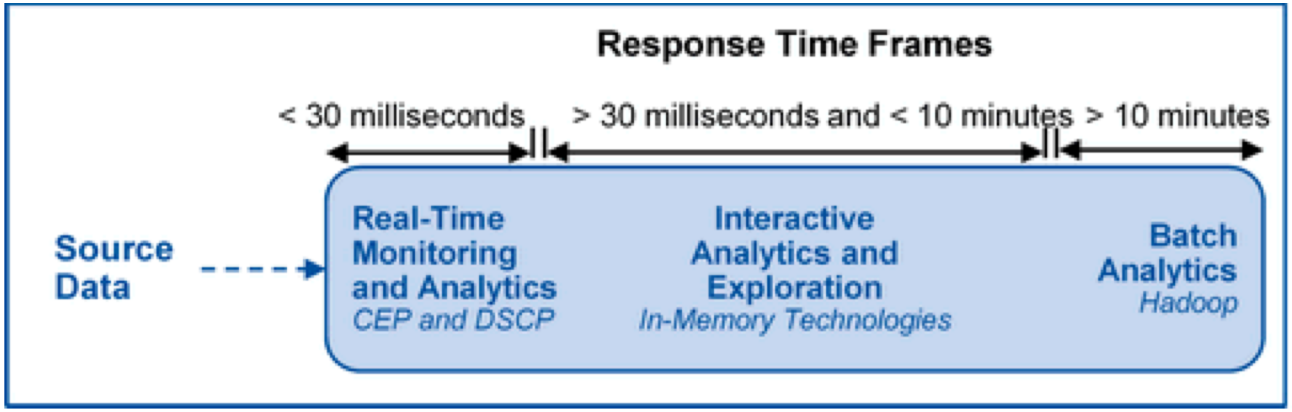
```
CREATE TABLE corporation
 (corp_id SMALLINT,
  name VARCHAR(30),
  CONSTRAINT pk_corporation PRIMARY KEY (corp_id)
 );
```

```
INSERT INTO corporation (corp_id, name) VALUES (27, 'Acme Paper Corporation');
```

```
SELECT name FROM corporation WHERE corp_id= 27;
```

SQL on Hadoop Application Categories.

- SQL widely used in transactional and datawarehouse relational system
- Use of SQL for different kinds of analytic workloads



Heudecker, Adrian, Choosing the right SQL Access Strategy, Gartner 2014

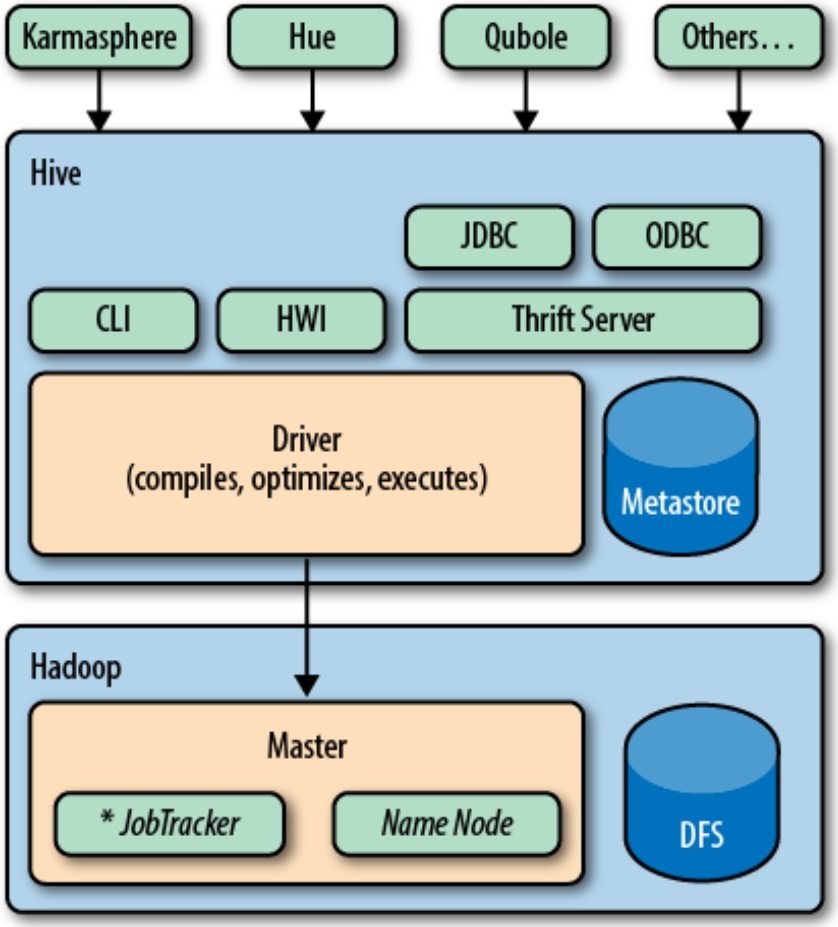
Architectural Choices

- SQL engine built on top **lower-level Hadoop frameworks**:
  - Hive was originally built on top of MapReduce. Now support for TEZ framework and Spark.
- **Direct processing of HDFS data** using agents on nodes:
  - Resource sharing between MR, YARN and SQL framework often statically configured.
  - Better support for long-running and multi-user applications planned.
- **Connector Approach:**
  - HDFS data is exposed at external table in the DBMS. Data needs to be moved before processing.

Hive Overview

- Most-widely SQL engine for Hadoop originally developed by Facebook.
- Compiles **HiveQL** (a subset of SQL) to MapReduce and processes Data in SQL.
- Flexible and extensible:
  - Data storage formats (Text, RC, ORC, Parquet,…)
  - Processing engines (MapReduce, Tez, Spark)
- **Schema on Read:**
  - Unlike traditional relational databases schema constraints are not enforced while writing. Hive assumes no control of the data (can be anywhere in HDFS) and thus schemas are enforced on read.

Hive Architecture



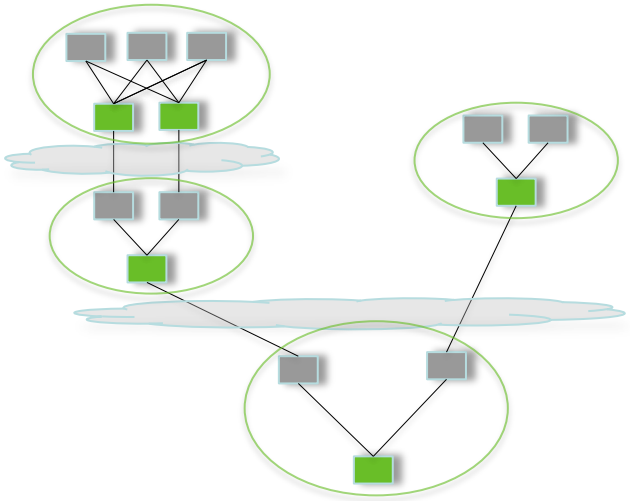
Dean Wampler et al., Programming Hive, O'Reilly, 2012

Hive Processing Engines

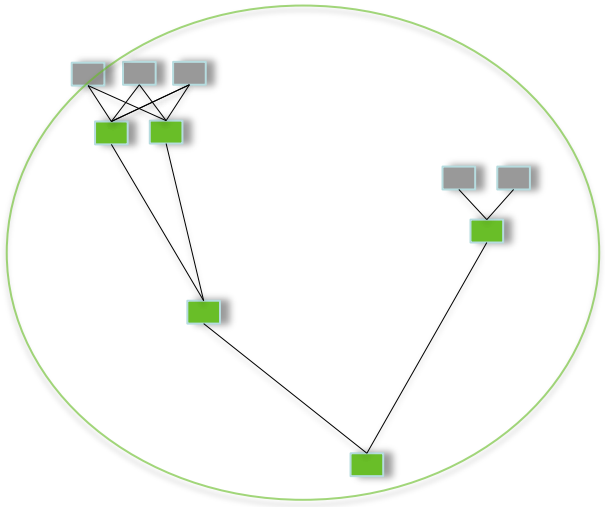
- Traditionally HiveQL is compiled into MapReduce jobs
- Hadoop MapReduce has several limitations:
  - Write to HDFS required between successive computations phases – iterative computations not well supported.
  - Job launch overhead.
- Alternative processing engines enable to retain resources and the re-use resources using long-running containers.
- Activate different Processing Engines:
  - set hive.execution.engine=spark;

Hive Processing Engines: Apache Tez

- YARN-based framework for optimizing the execution of data-flow processing pipelines*
  - API for complex data flow pipelines: Support for common Direct Asymmetric Graph (DAG) patterns
  - Targeted towards data processing applications like Hive/Pig but not limited to it.
  - Performance Optimizations: Container re-use and in-memory caching



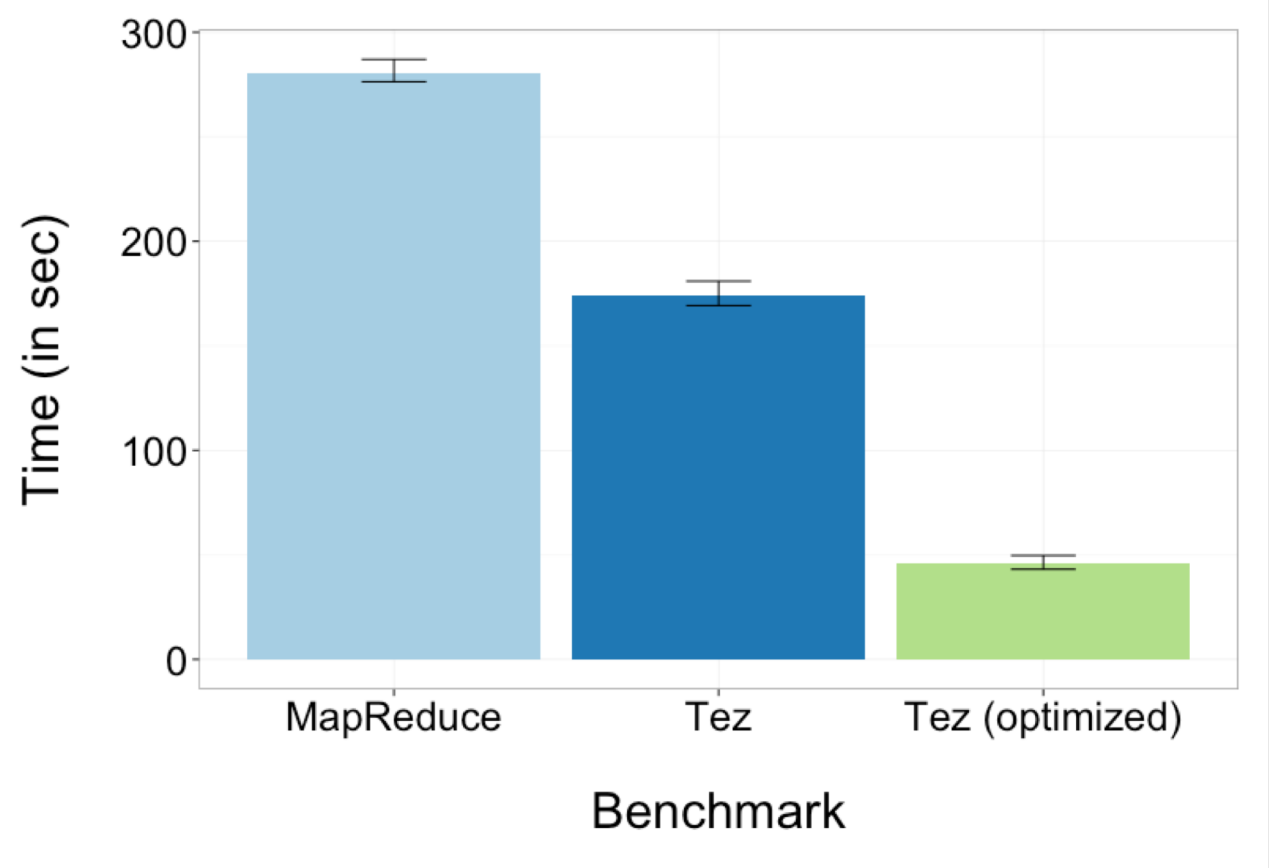
Pig/Hive - MR



Pig/Hive - Tez

Inspired by Microsoft Dryad: Isard et.al. Dryad: Distributed Data-Parallel Programs from Sequential Building Blocks, http://research.microsoft.com/en-us/projects/dryad/

Hive Processing Engines



Runtimes for processing 3.2 TB (raw data) in ORC format using a simple select query.

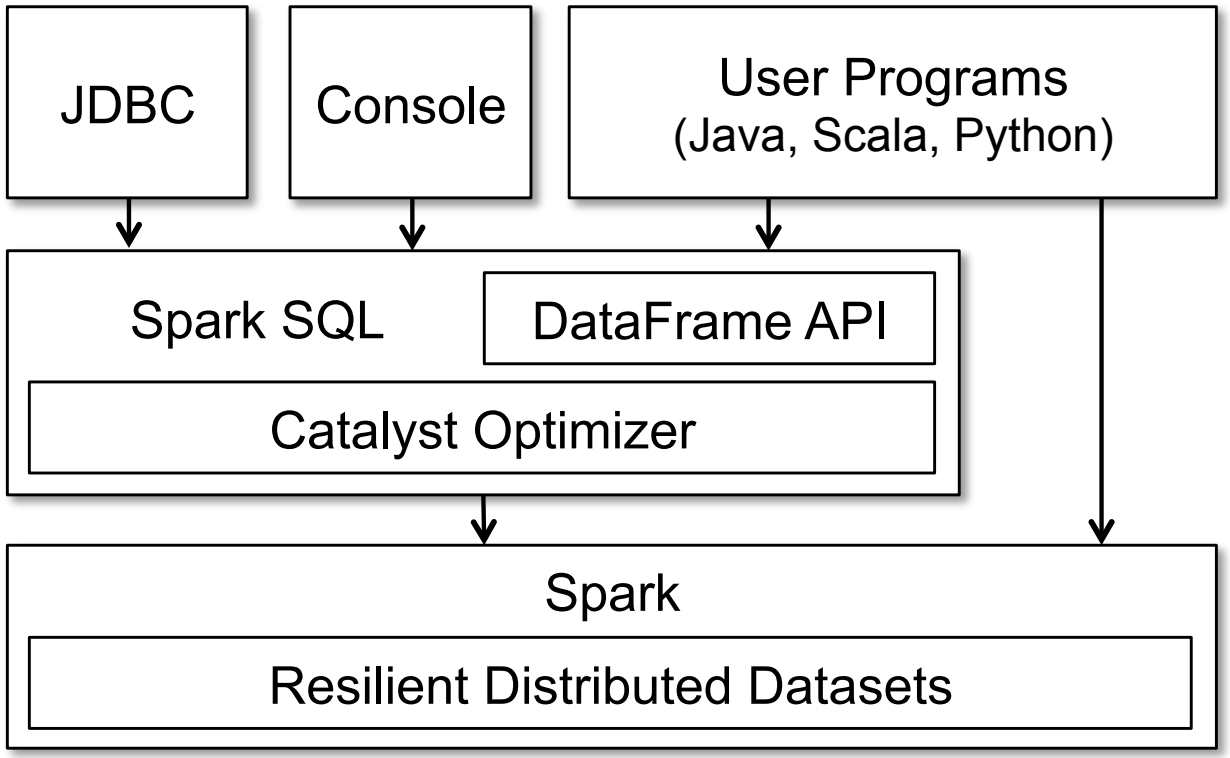


Figure 1: Interfaces to Spark SQL, and interaction with Spark.

Spark-SQL Properties

- Support relational processing both within Spark programs (on native RDDs) and on external data sources using a programmer-friendly API.
- Provide high performance using established DBMS techniques.
- Easily support new data sources, including semi-structured data and external databases amenable to query federation.
- Enable extension with advanced analytics algorithms such as graph processing and machine learning.
- Interoperability with Hive: Retrieve Hive databases/tables from metastore and query them

Armbrust, Spark SQL: Relational Data Processing in Spark,
http://people.csail.mit.edu/matei/papers/2015/sigmod_spark_sql.pdf

Spark-SQL Catalyst

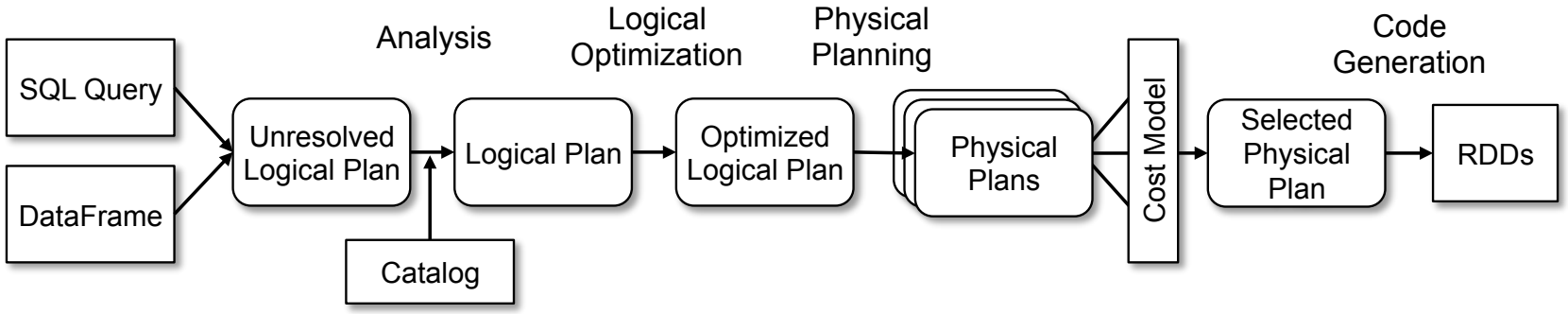
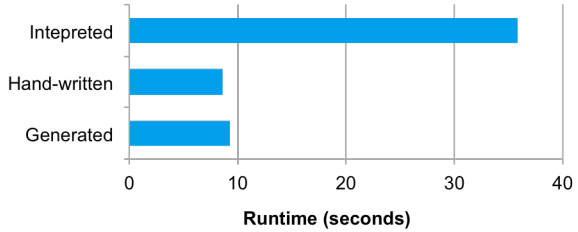


Figure 3: Phases of query planning in Spark SQL. Rounded rectangles represent Catalyst trees.



Armbrust, Spark SQL: Relational Data Processing in Spark,
http://people.csail.mit.edu/matei/papers/2015/sigmod_spark_sql.pdf

Spark-SQL Dataframes.

- A Dataset is a distributed collection of data. Dataset is a new interface added in Spark 1.6 that provides the benefits of RDDs (strong typing, ability to use powerful lambda functions) with the benefits of Spark SQL's optimized execution engine.
- A **DataFrame** is a *Dataset* organized into named columns. It is conceptually equivalent to a table in a relational database or a data frame in R/Python, but with richer optimizations under the hood.
- DataFrames can be constructed from a wide array of sources such as: structured data files, tables in Hive, external databases, or existing RDDs. The DataFrame API is available in Scala, Java, Python, and R.

Spark-SQL Dataframe Example


```
from pyspark.sql.functions import avg
dataRDD = sc.parallelize([("Jim", 20), ("Anne", 31), ("Jim", 30)])

dataDF = dataRDD.toDF(["name", "age"])

# RDD
(dataRDD.map(lambda (x,y): (x, (y,1))) .reduceByKey(lambda x,y: (x[0]
+y[0], x[1] +y[1])) .map(lambda (x, (y, z)): (x, y / z)))

# DataFrame
dataDF.groupBy("name").agg(avg("age"))
```

	SQL	DataFrames	Datasets
Syntax Errors	Runtime	Compile Time	Compile Time
Analysis Errors	Runtime	Runtime	Compile Time

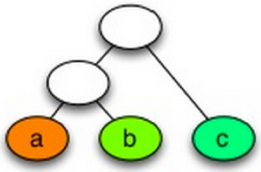
https://databricks.com/blog/2016/07/14/a-tale-of-three-apache-spark-apis-rdds-dataframes-and-datasets.html

Data Storage Formats

- File Format:
  - Text File (CSV)
  - Other serialization formats (Avro, Protocol Buffers)
  - Columnar file format (Parquet versus ORC)
- Compression:
  - Trade-Off CPU, IO vs. Filesize
  - Most common:
    - Gzip
    - Snappy
    - LZO
- Criteria for selection: Interoperability, Space Efficiency and Query Efficiency
- Not all file formats can work with all tools in the Hadoop ecosystem.

Columnar Data Formats (1)

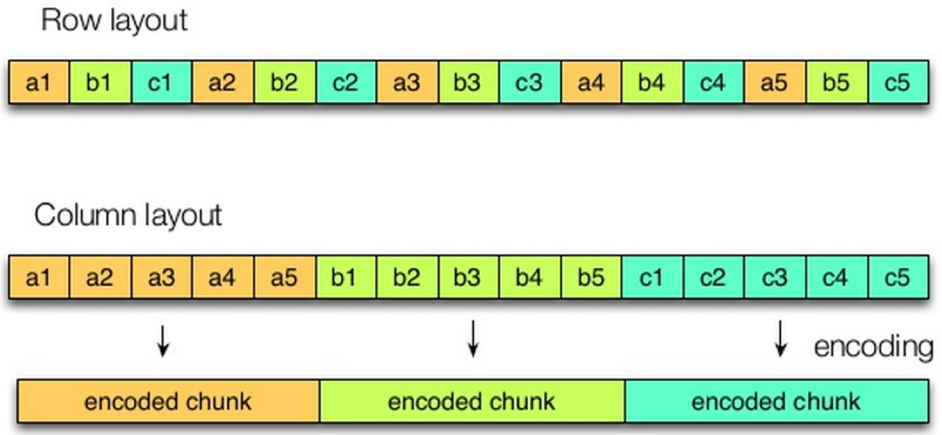
- Traditional database systems store data in a row-oriented fashion
- Only efficient if many columns need to be analyzed



Nested schema

Logical table representation

a	b	c
a1	b1	c1
a2	b2	c2
a3	b3	c3
a4	b4	c4
a5	b5	c5



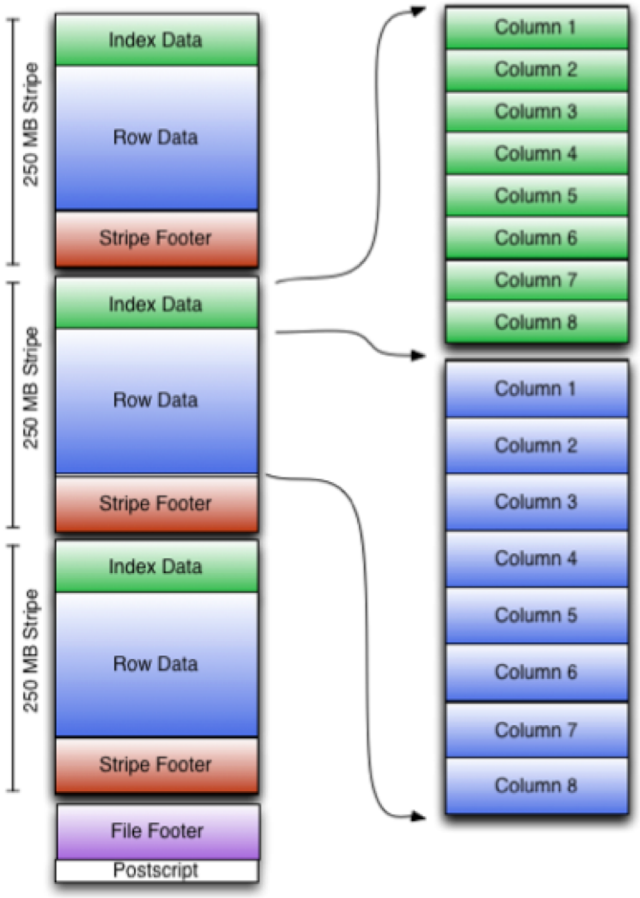
http://www.slideshare.net/cloudera/hadoop-summit-36479635

Columnar Data Formats (2)

- Skips I/O on columns that are not a part of the query.
- Works well for queries that only access a small subset of columns. If many columns are being accessed, then row-oriented is generally preferable.
- **Compression** on columns is generally very efficient, particularly if the column has few distinct values.
- Columnar storage is often well suited for **data-warehousing type** applications where users want to aggregate certain columns over a large collection of records.

ORC File Format

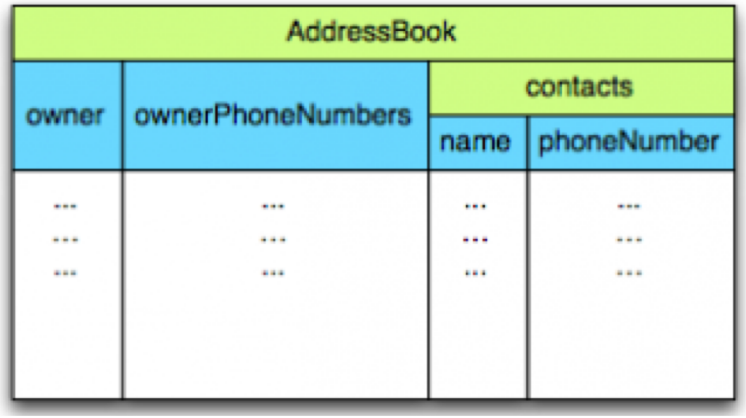
- Successor of the RC (an early columnar format for Hive)
- Always-on compression
- Splitable: data is managed in 256 megabyte chunks of rows
- Optimized for Hive data types
- Index enables row-skipping within a stripe for rapid read
- Stores stats: sum, average,…
- Predicate pushdown avoids IO



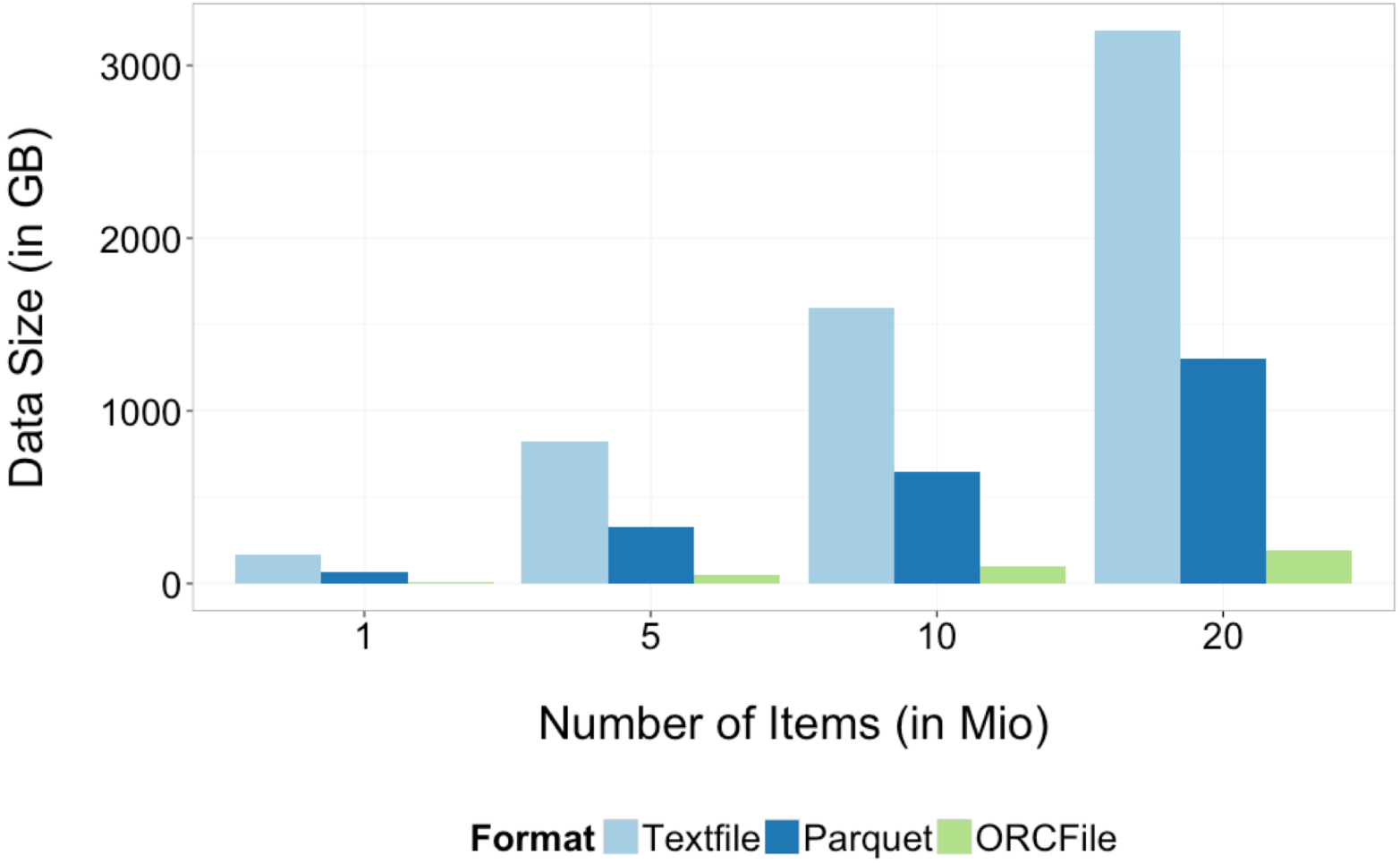
http://docs.hortonworks.com/HDPDocuments/HDP2/HDP-2.0.0.2/ds_Hive/orcfile.html

- Designed as general-purpose format for complex nested data structures and different query engines.
  - Stores nested data structures in a flat columnar format using a technique outlined in Google Dremel Paper (http://research.google.com/pubs/pub36632.html)
  - Written in C and efficient support for different encoding schemes

Column	Type
owner	string
ownerPhoneNumbers	string
contacts.name	string
contacts.phoneNumber	string

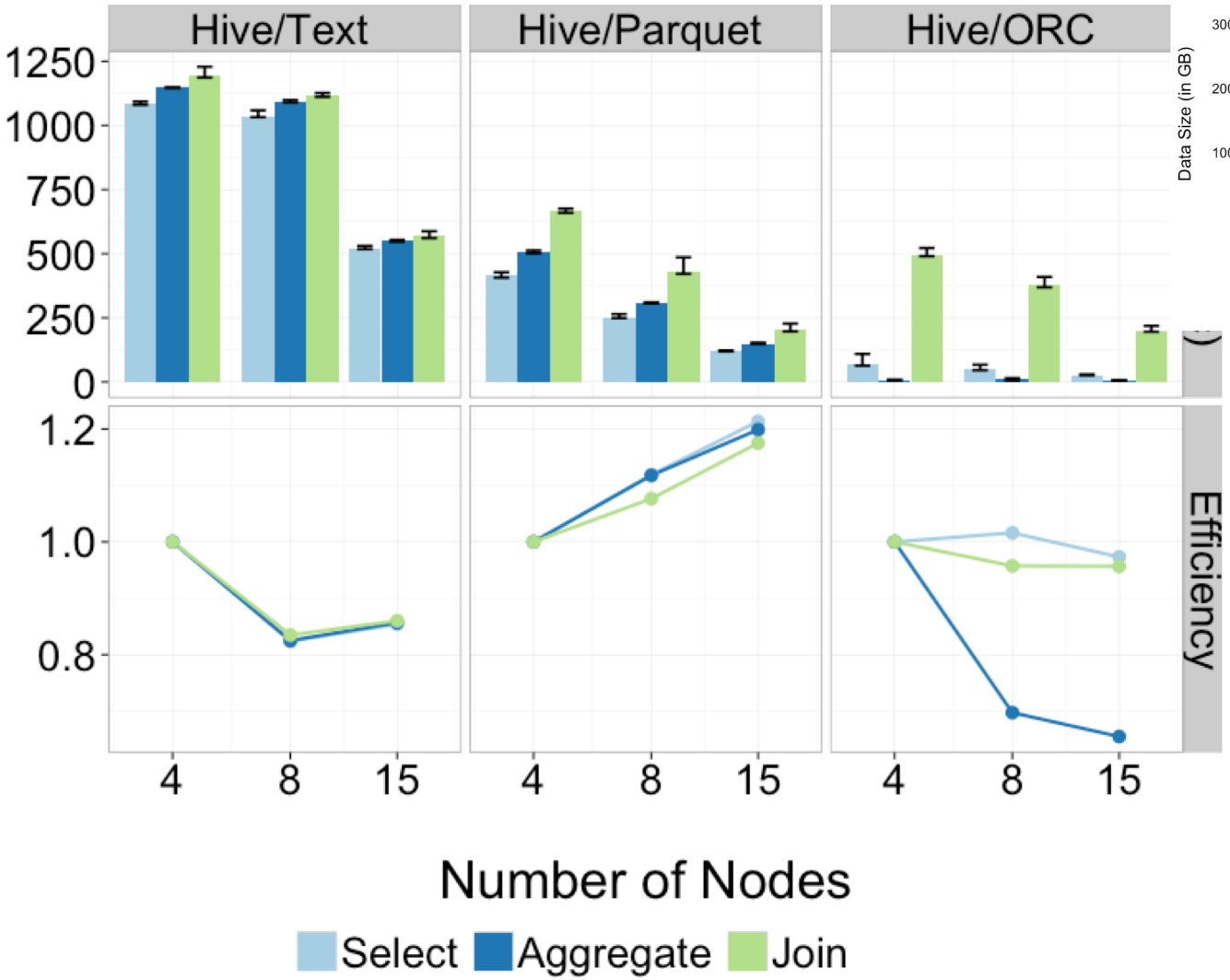


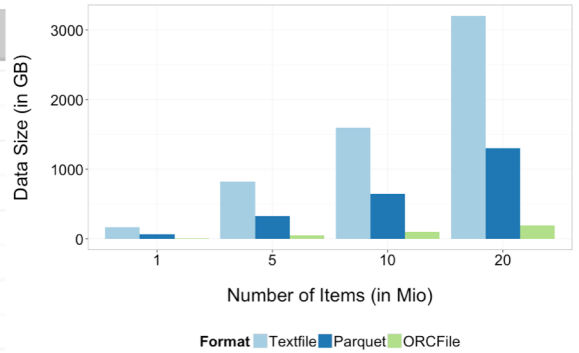
Hive: Data Formats



Luckow et al., Automotive Big Data, IEEE Big Data, 2015.

Hive: Data Formats





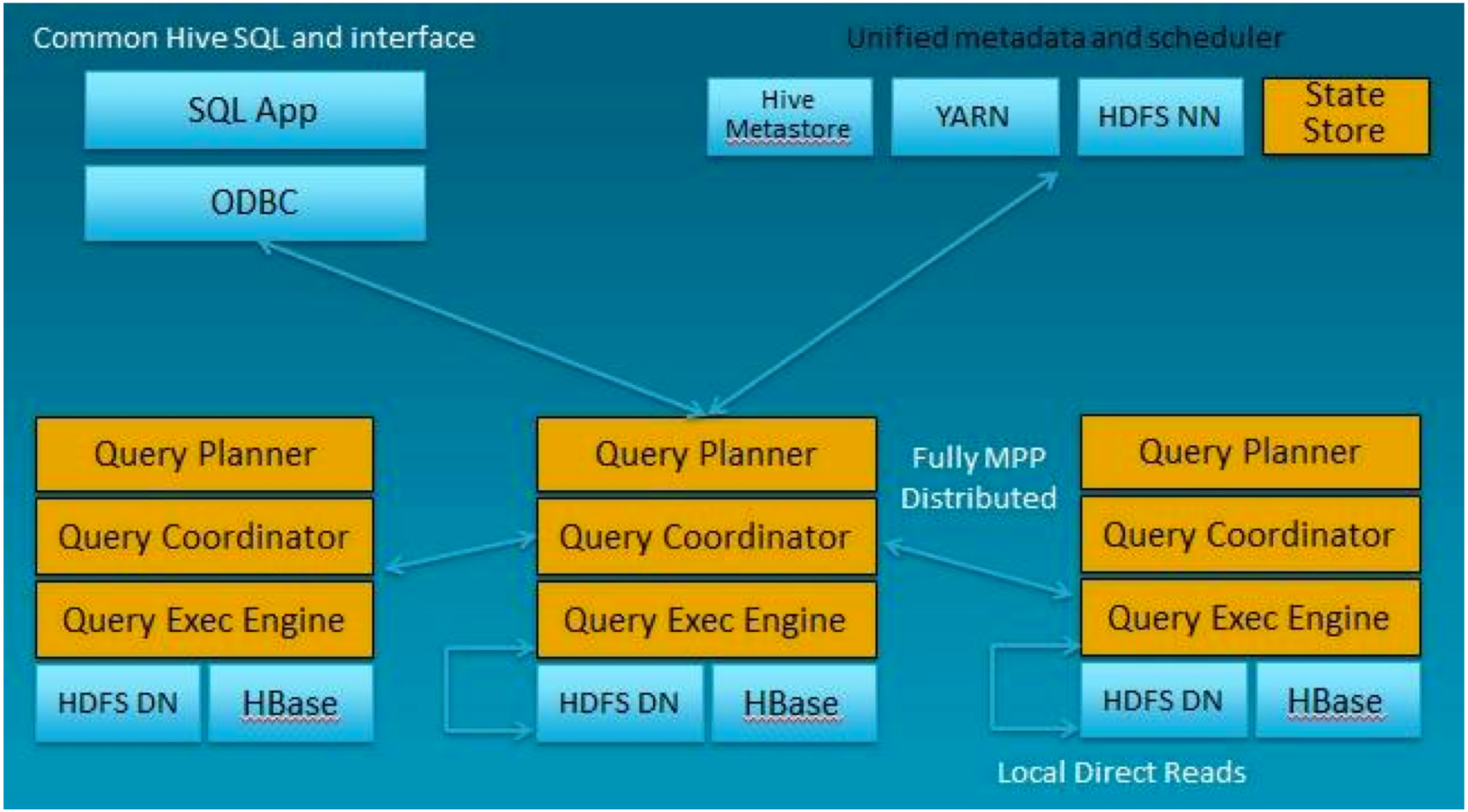
~20 mio items
15 nodes HDP 2.1 cluster

Luckow et al., Automotive Big Data, IEEE Big Data, 2015.

Cloudera Impala

- SQL processing engine that is co-located with Hadoop daemons
- Integrates on several levels with Hadoop ecosystem:
  - Usage of HDFS for storage of data
  - Support for common Hadoop file formats (ORC, Parquet, RC, Text,…)
  - Stores metadata in Hive meta-store
- Architecture
  - Daemons are written in C++
  - In-Memory caching (but does spill to disk)
  - High multi-user concurrency

Impala Architecture

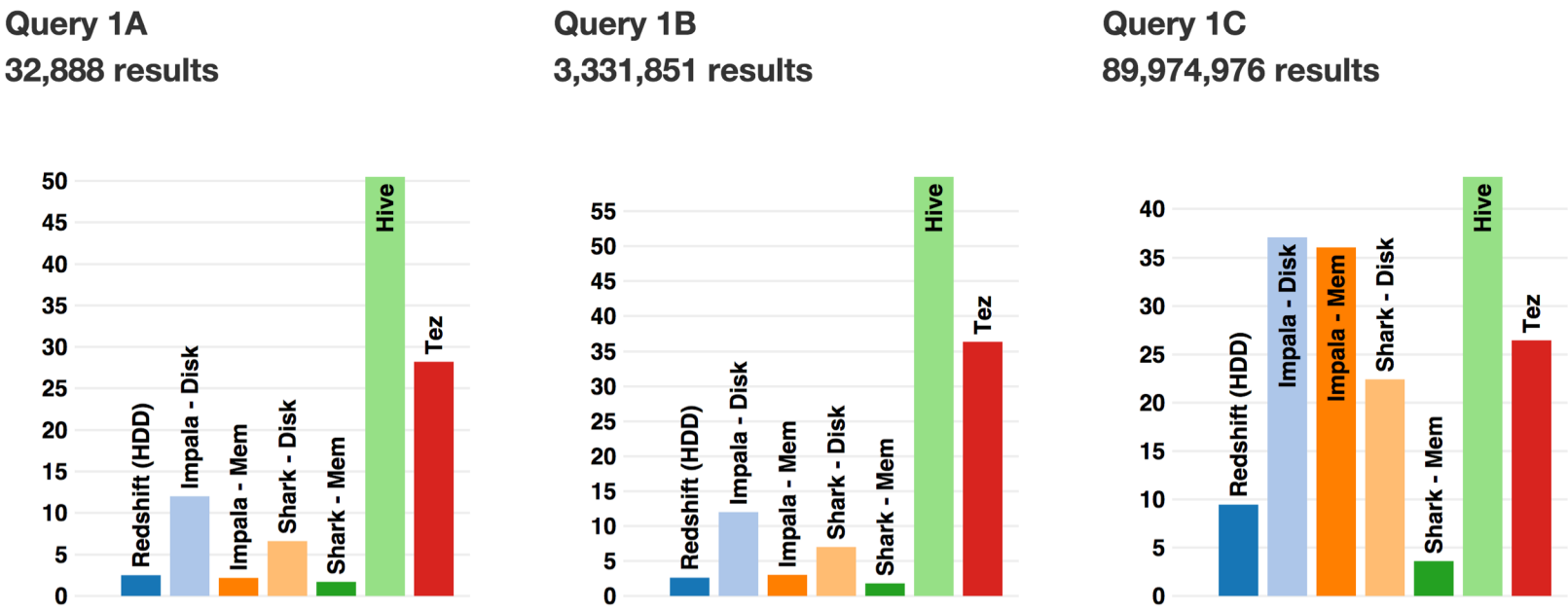


Source: Cloudera

Further Considerations

- De-normalizing:
  - reduces need to join data sets and trades off disk space for query performance
  - Joins are very expansive in Hadoop:
    - Map-side join requires both table to be partitioned by the join key
    - Reduce-side join require to move tables through shuffle phase
- Future performance improvement can be expected
  - Hive Cost-based Optimizer (CBO), improved Hive Task management (Tez/LLAP (live long and process))
- Despite **schema-less nature**: having a good meta-data and file system layout is important

- AMPLab Hadoop SQL Benchmark:
  - `SELECT pageURL, pageRank FROM rankings WHERE pageRank > X`



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

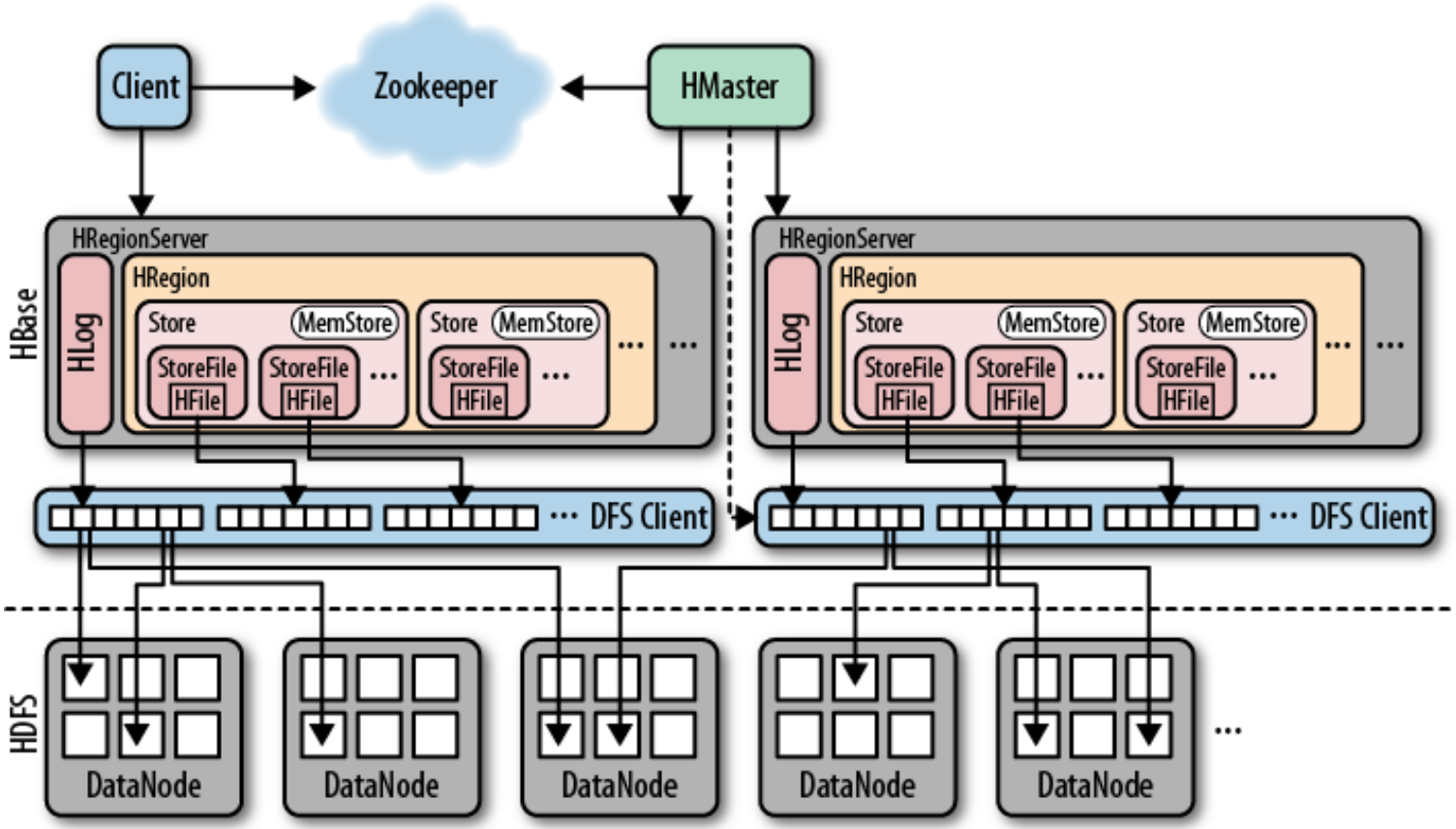
HBase: Processing Engines for Random Data Access.

HBase

- Enables random data access on top of Hadoop*
- NoSQL API for storing and accessing data:
  - A Row Key is used for storing and retrieving data inside of Hbase
  - Operations: Put data in, get data out and iterate through data
  - Row-Key determines partitioning of data and thus, can be used to co-locate data
- HBase uses HFile format. For analytical queries it can be processed with Hive:
  - Trade-off: HBase scan rates are about 8-10x slower then HDFS

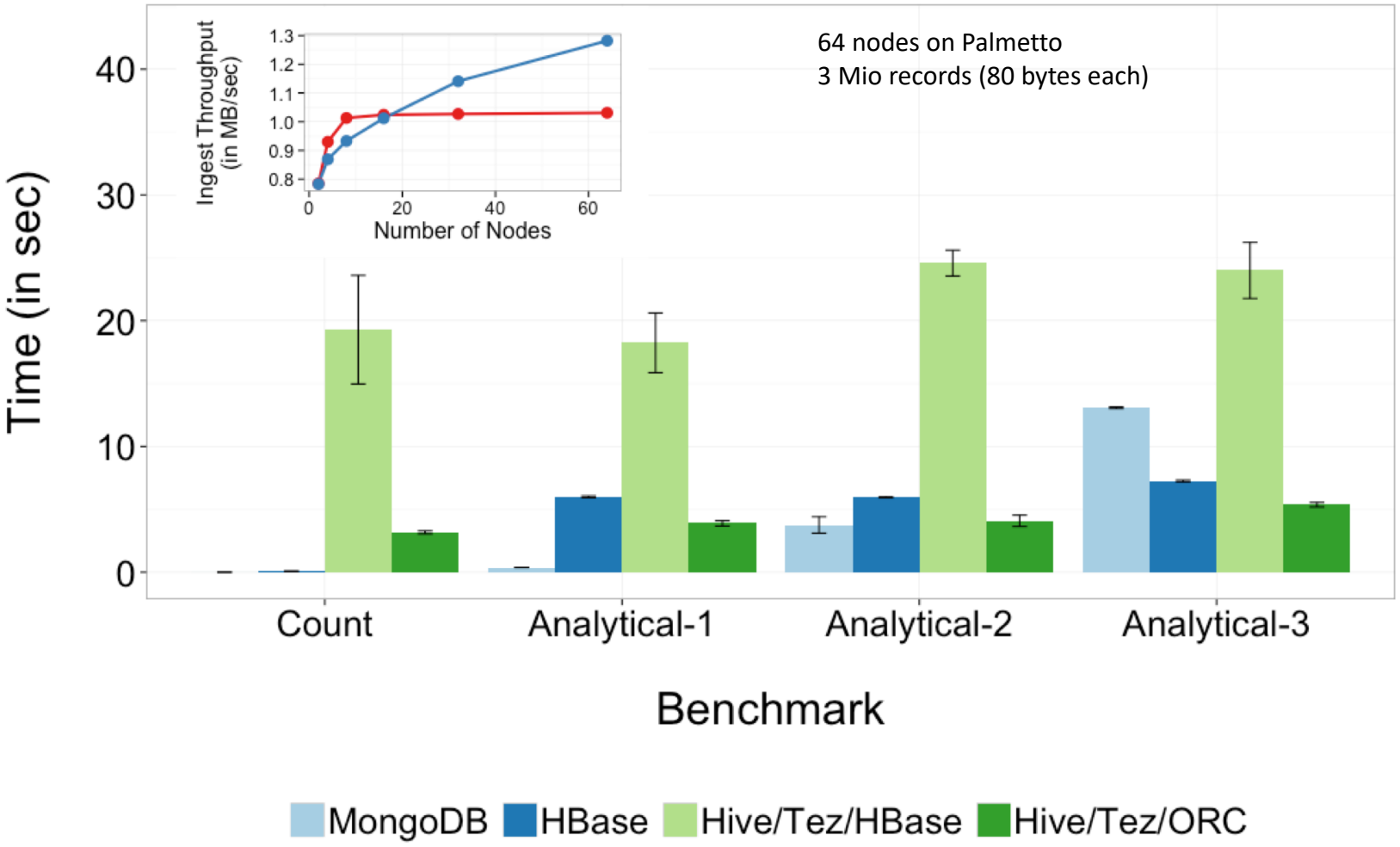
*Inspired by Google BigTable: Dean et al, Bigtable: A Distributed Storage System for Structured Data, http://research.google.com/archive/bigtable.html

HBase



Lars George, HBase – The Definitive Guide, O'Reilly, 2011

NoSQL vs. Analytics Performance



SQL on Hadoop Summary

Framework	Usage
MapReduce	Large batch processes
Hive	Batch Processing with SQL like language. More interactive with **TEZ**.
Impala	Interactive SQL (for medium size data)
Spark-SQL	Interactive SQL (for medium size data)
HBase	Random access to record-level data, sparse datasets

Plumbing required: not all tools work with all tools.

Conclusion

- SQL on Hadoop allows to query data residing on Hadoop infrastructures using SQL language:
  - Data processing is performed at lowest possible level minimizing data movements
  - Maturity and performance of frameworks is increasingly becoming better
- **Schema on Read:**
  - Gold-Standard BI versus Exploratory Analytics
  - Schema changes happen frequently without warning
- Pluggable components for different concerns:
  - data formats
  - compression
  - processing engines (SQL, MapReduce, Spark)

Outlook

- Not everything works with everything ☹
- Community still undecided on several technological issues:
  - Tez vs. Spark
  - Hive on Spark vs. Hive on Tez vs. Hive on MapReduce
  - Many architectural options, but not all make sense!

SQL Engines

- Thusoo et al., Hive: A Warehousing Solution Over a Map-Reduce Framework, 2009.
- Armbrust et al., Spark SQL: Relational Data Processing in Spark, 2015.
- Kornacker et al., Impala: A Modern, Open-Source SQL Engine for Hadoop, 2015

"It is tough to make prediction, especially about the future" Yogi Berra

Data Science and Machine Learning

Sources:

- Jurney, Agile Data Science 2.0
- Google Machine Learning Crash Course
- Gartner
- See sources on slides.

What is Data Science?

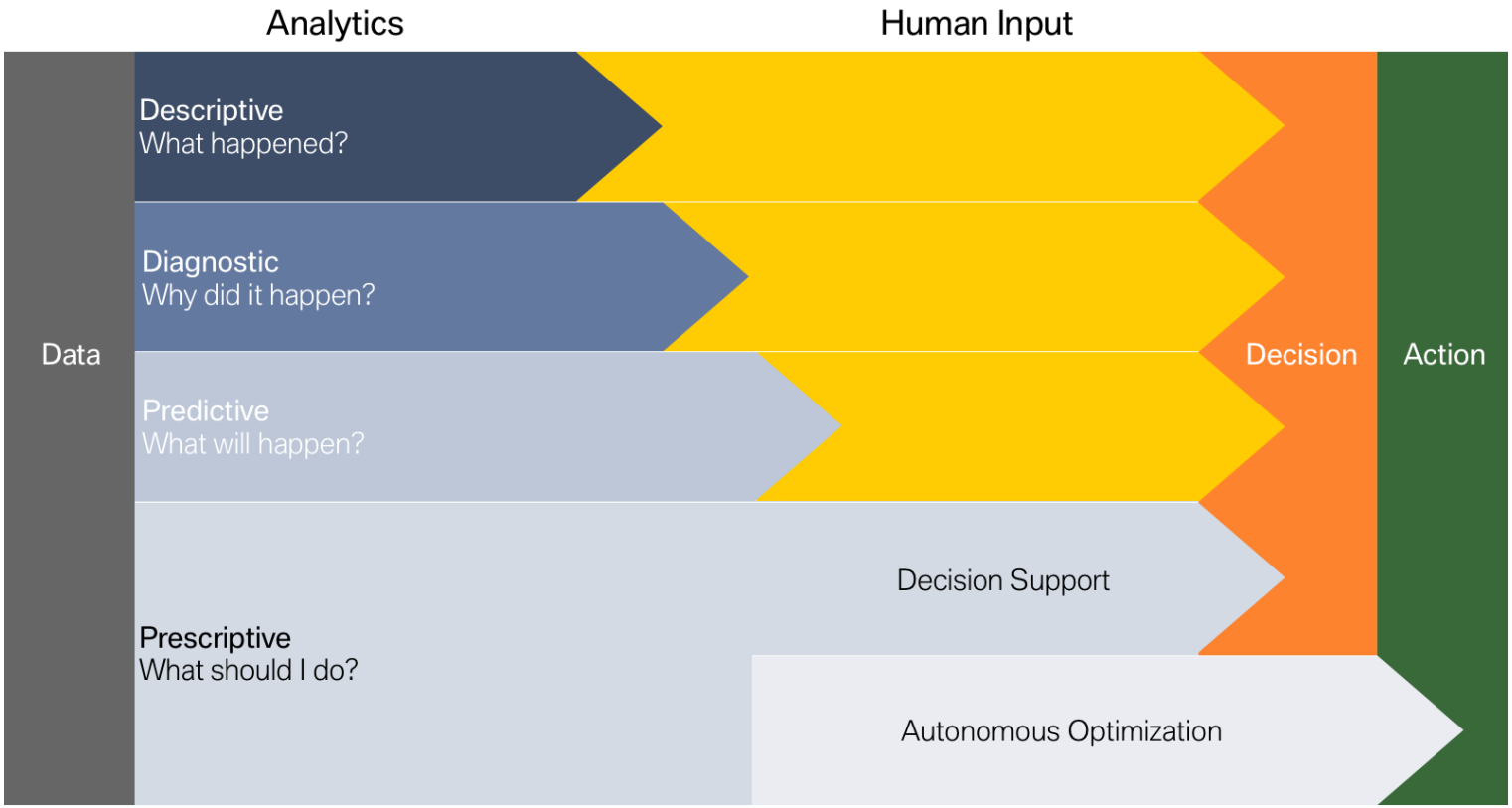
Data Science is the extraction of actionable knowledge directly from data through a process of discovery, hypothesis, and analytical hypothesis analysis.

A **Data Scientist** is a practitioner who has sufficient knowledge of the overlapping regimes of expertise in business needs, domain knowledge, analytical skills and programming expertise to manage the end-to-end scientific method process through each stage in the big data lifecycle.

Examples

- Predict whether a patient, hospitalized due to a heart attack, will have a second heart attack. The prediction is to be based on demo- graphic, diet and clinical measurements for that patient.
- Predict the price of a stock in 6 months from now, on the basis of company performance measures and economic data.
- Identify the numbers in a handwritten ZIP code, from a digitized image.
- Estimate the amount of glucose in the blood of a diabetic person, from the infrared absorption spectrum of that person's blood.
- Identify the risk factors for prostate cancer, based on clinical and demographic variables.

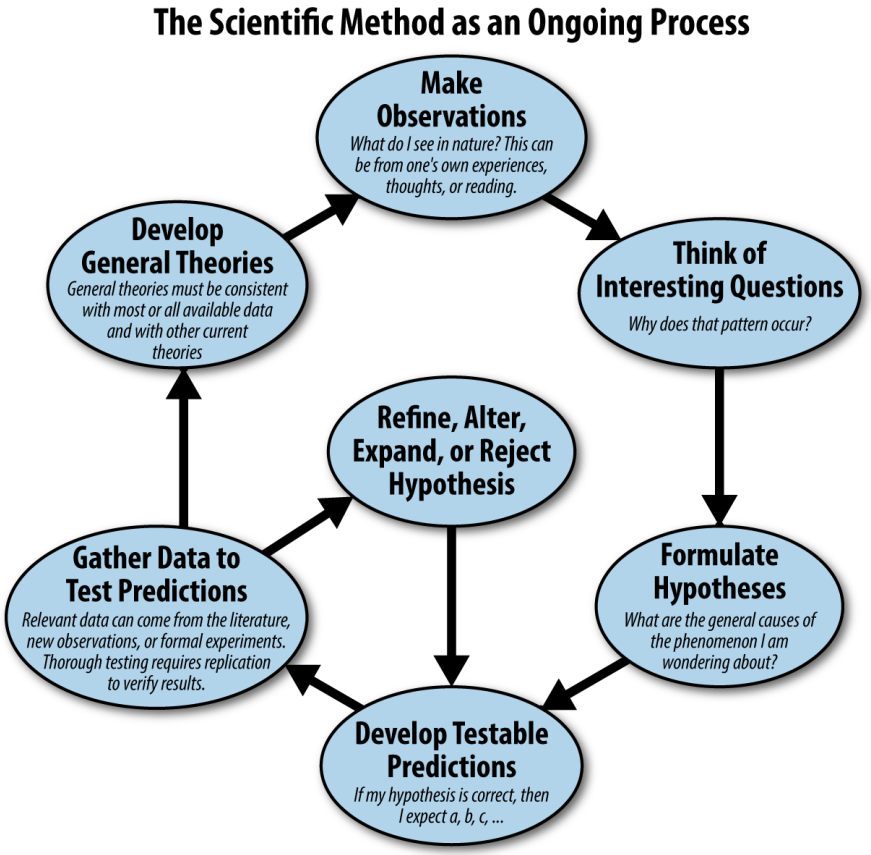
From Descriptive to Predictive and Prescriptive



Source: Gartner

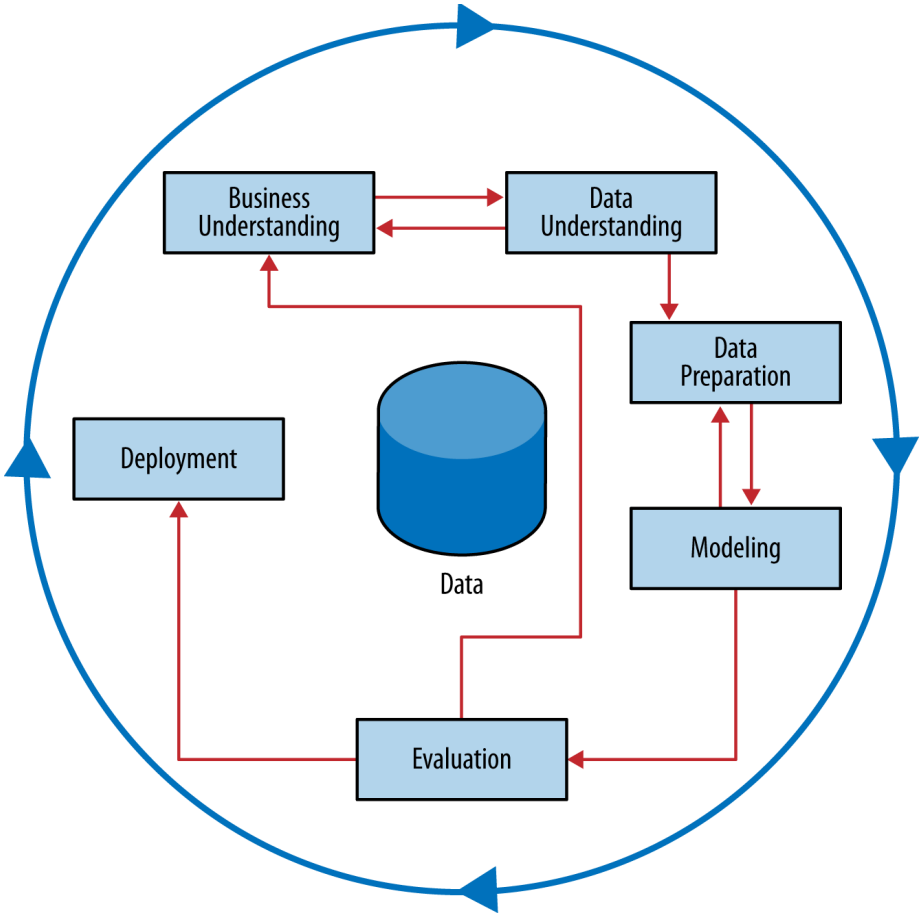
Typical Data Science Tasks

- Define question
- Data acquisition
- Exploratory data analysis using plots, summary statistics
- Data Engineering (logs → MapReduce/Spark/Dataframes/SQL → dataset → join with other data → MapReduce/Spark/Dataframes/SQL → external data → join)
- Build data models: Unsupervised and supervised learning
- Validate Model
- Building data products and services
- Communication & Visualization



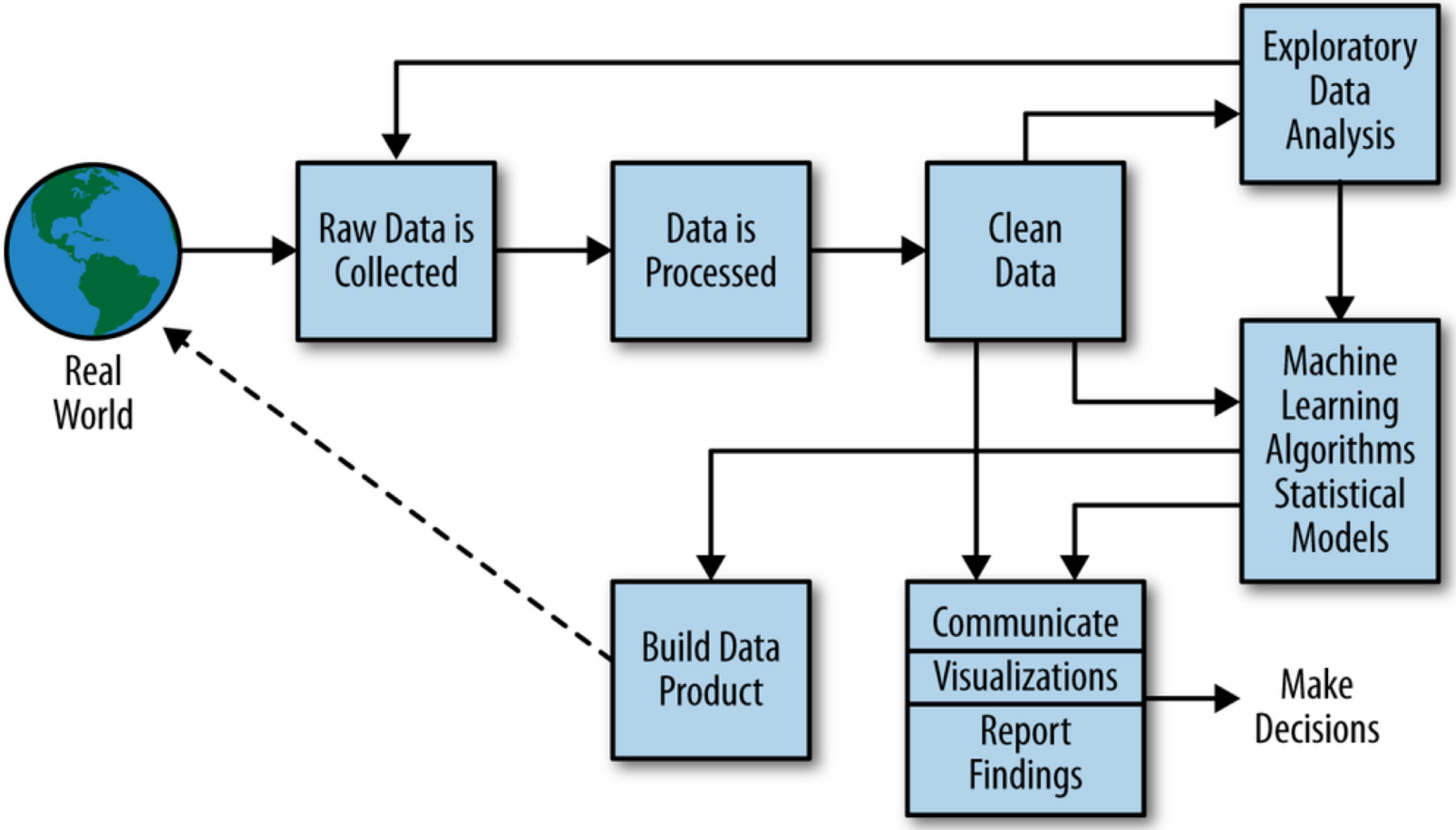
Russell Jurney, Agile Data Science 2.0, 2017

Crisp DM Process



Provost, Fawcett, Data Science for Business, O'Reily 2013.
https://www.the-modeling-agency.com/crisp-dm.pdf

Data Science Process



Cathy O'Neil, Rachel Schutt, Doing Data Science, O'Reilly, 2013

Types of Data and Representation

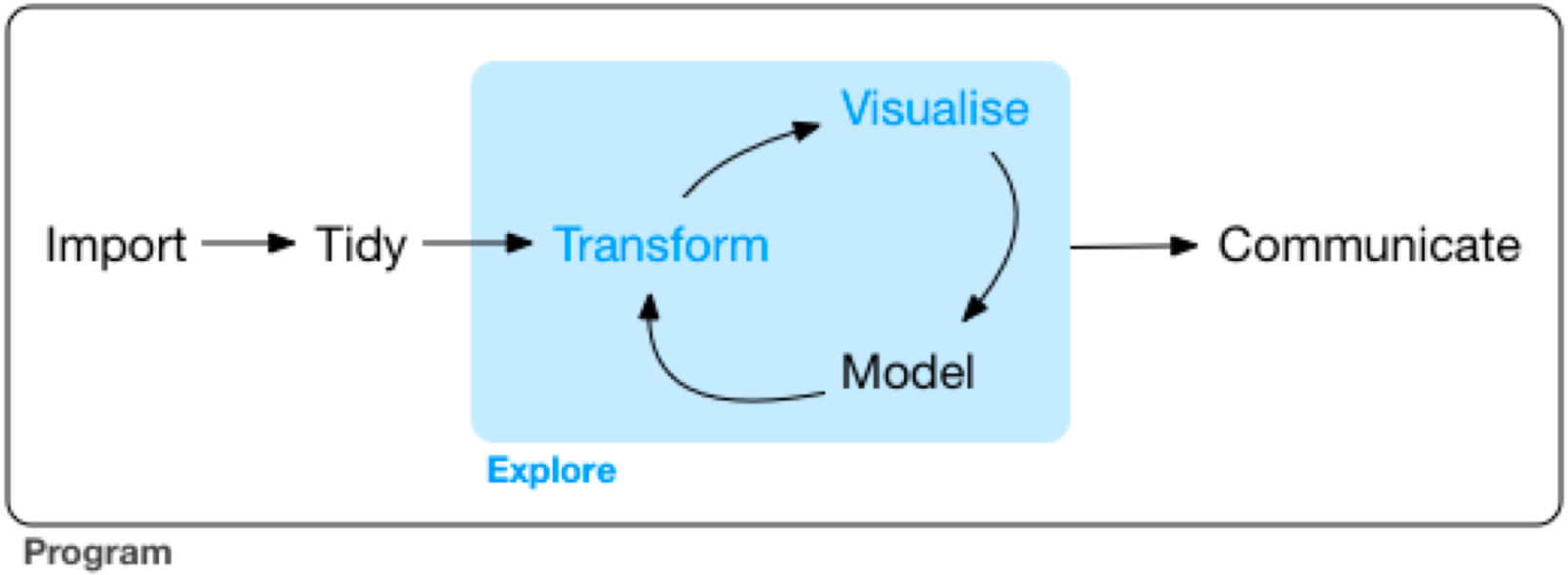
- **Quantitative:** Measurements of this type of variable or attribute are represented by continuous real-valued numbers.
- **Ordinal:** The values of this type of variable are often represented as contiguous integers, and the realizable values are considered to be an ordered set.
- **Qualitative, Categorical, Discrete, Nominal:** No explicit ordering in the classes. Often, descriptive labels rather than numbers are used to denote the classes.

Feature	Demographic	# Values	Type
1	Sex	2	Categorical
2	Marital status	5	Categorical
3	Age	7	Ordinal
4	Education	6	Ordinal
5	Occupation	9	Categorical
6	Income	9	Ordinal
7	Years in Bay Area	5	Ordinal
8	Dual incomes	3	Categorical
9	Number in household	9	Ordinal
10	Number of children	9	Ordinal
11	Householder status	3	Categorical
12	Type of home	5	Categorical
13	Ethnic classification	8	Categorical
14	Language in home	3	Categorical

Hastie, Tibshirani, Friedman, Elements of Statistical Learning, 2008

- 80% of data analysis is spent on the process of cleaning and preparing the data (Dasu and Johnson 2003).
- Data Preparation is the process of preparing and providing data for data discovery and advanced analytics:
  - Access: Pulling Data from Source System
  - Profiling: Generating and using descriptive statistics to understand what is in the data
  - Transformation: Functional change of data to facilitate the needs of later analysis stages:
    - Structuring refers to actions that manipulate the schema
    - Enrichment refers to the additions of columns/fields that add new information to dataset
    - Cleaning: Ensure that records provide valid values

Data Wrangling



"Happy families are all alike; every unhappy family is unhappy in its own way." — Leo Tolstoy

"Tidy datasets are all alike, but every messy dataset is messy in its own way." — Hadley Wickham

Data Organization: Tidy Data (1)

- Messy Data:
  - Column headers are values, not variable names.
  - Multiple variables are stored in one column.
  - Variables are stored in both rows and columns.
  - Multiple types of observational units are stored in the same table.
  - A single observational unit is stored in multiple tables.

Dasu T, Johnson T (2003). Exploratory Data Mining and Data Cleaning. John Wiley & Sons.
H. Wickham, Tidy Data, https://www.jstatsoft.org/article/view/v059i10/v59i10.pdf

Data Organization: Tidy Data (2)

- Tidy data is a standard way of mapping the meaning of a dataset to its structure:
    1. Each variable forms a column.
    2. Each observation forms a row.
    3. Each type of observational unit forms a table.

H. Wickham, Tidy Data, https://www.jstatsoft.org/article/view/v059i10/v59i10.pdf

Data Organization: Tidy Data (2)

religion	<$10k	$10–20k	$20–30k	$30–40k	$40–50k	$50–75k
Agnostic	27	34	60	81	76	137
Atheist	12	27	37	52	35	70
Buddhist	27	21	30	34	33	58
Catholic	418	617	732	670	638	1116
Don't know/refused	15	14	15	11	10	35
Evangelical Prot	575	869	1064	982	881	1486
Hindu	1	9	7	9	11	34
Historically Black Prot	228	244	236	238	197	223
Jehovah's Witness	20	27	24	24	21	30
Jewish	19	19	25	25	30	95

Table 4: The first ten rows of data on income and religion from the Pew Forum. Three columns, $75–100k, $100–150k and >150k, have been omitted.

Data Organization: Tidy Data (3)

row	a	b	c
A	1	4	7
B	2	5	8
C	3	6	9

(a) Raw data

row	column	value
A	a	1
B	a	2
C	a	3
A	b	4
B	b	5
C	b	6
A	c	7
B	c	8
C	c	9

(b) Molten data

(b) Contains exactly the same data as (a)

religion	income	freq
Agnostic	<$10k	27
Agnostic	$10–20k	34
Agnostic	$20–30k	60
Agnostic	$30–40k	81
Agnostic	$40–50k	76
Agnostic	$50–75k	137
Agnostic	$75–100k	122
Agnostic	$100–150k	109
Agnostic	>150k	84
Agnostic	Don't know/refused	96

Tidy Data of Religion/Income Dataset

"The simple graph has brought more information to the data analyst's mind than any other device." — John Tukey

Data Visualization.

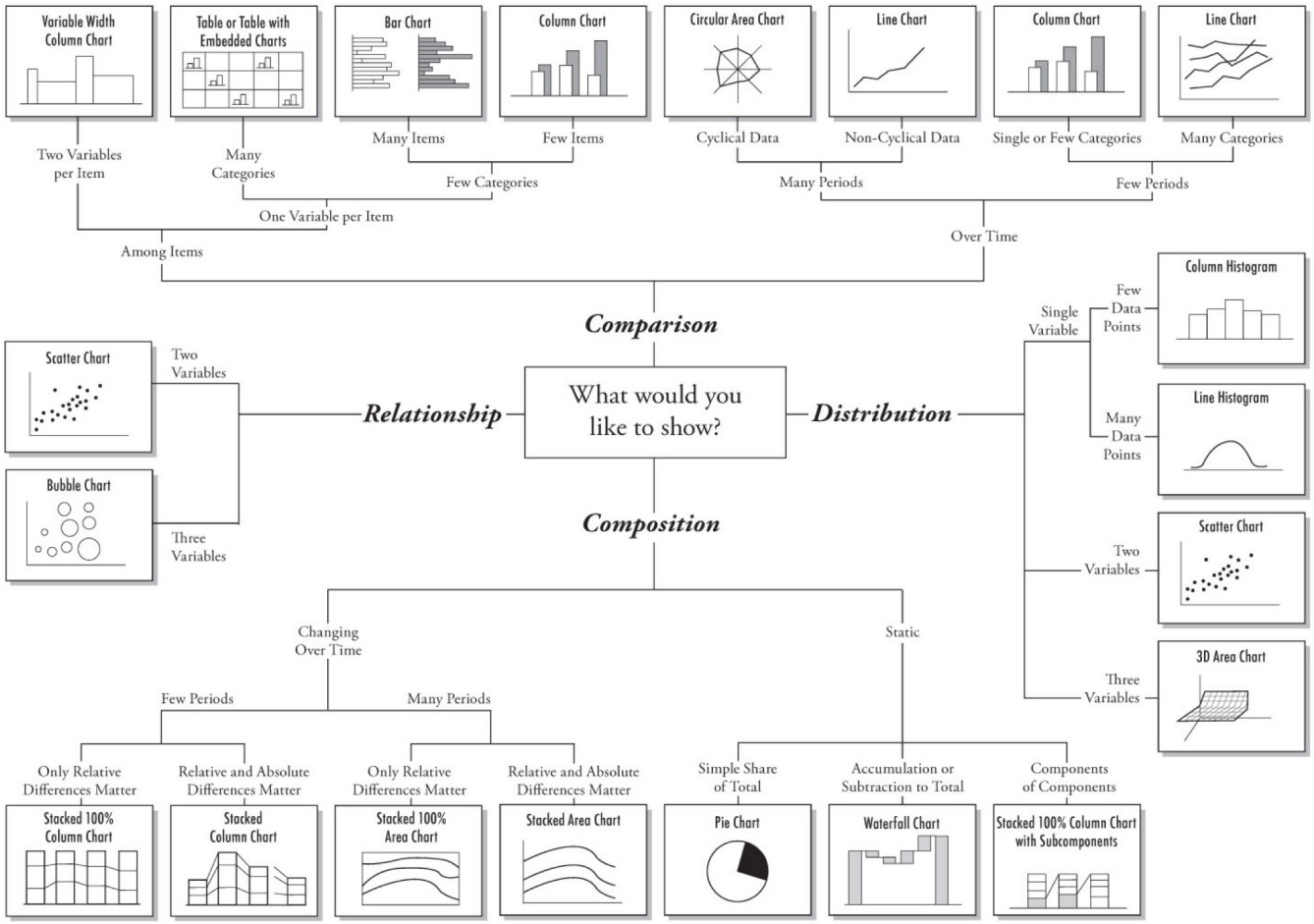
Visualizations

- Why?
  - Move huge amount of information into the brain quickly
  - Visualizationtakesadvantageofourbrains'built-in"software"toidentifypatterns and communicate relationships and meaning.
  - Visualization can inspire new questions and further exploration.
  - Visualization helps identify sub-problems.
  - Visualization is really good for identifying trends and outliers, discovering or searching for interesting or specific data points in a larger field, etc.
- Different objectives: exploration and explanation:
  - Exploratory data visualizations are appropriate when you have a whole bunch of data and you're not sure what's in it.
  - Explanatory data visualization is appropriate when you already know what the data has to say, and you are trying to tell that story to somebody else.

Visual Encodings

Example	Encoding	Ordered	Useful values	Quantitative	Ordinal	Categorical	Relational
	position, placement	yes	infinite	Good	Good	Good	Good
1, 2, 3; A, B, C	text labels	optional alpha or num	infinite	Good	Good	Good	Good
	length	yes	many	Good	Good		
	size, area	yes	many	Good	Good		
	angle	yes	medium	Good	Good		
	pattern density	yes	few	Good	Good		
	weight, boldness	yes	few		Good		
	saturation, brightness	yes	few		Good		
	color	no	few (<20)			Good	
	shape, icon	no	medium			Good	
	pattern texture	no	medium			Good	
	enclosure, connection	no	infinite			Good	Good
	line pattern	no	few				Good
	line endings	no	few				Good
	line weight	yes	few		Good		

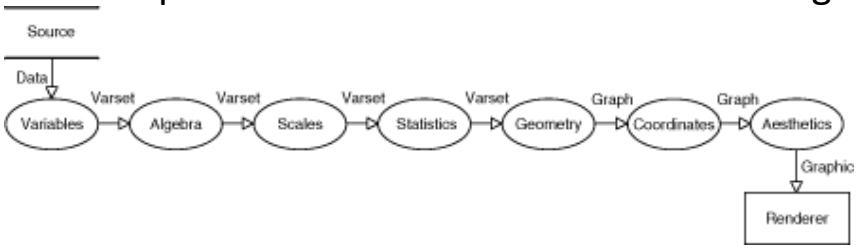
Chart Suggestions—A Thought-Starter



A. Abela, Chart Suggestions, http://extremepresentation.typepad.com/blog/2006/09/choosing_a_good.html, 2009

Grammar of Graphics

- The data that you want to visualize and a set of aesthetic mappings describing how variables in the data are mapped to aesthetic attributes that you can perceive.
- Geometric objects, geoms for short, represent what you actually see on the plot: points, lines, polygons, etc.
- Statistical transformations, stats for short, summarize data in many useful ways. For example, binning and counting observations to create a histogram, or summarizing a 2d relationship with a linear model. Stats are optional, but very useful.
- The scales map values in the data space to values in an aesthetic space, whether it be color, or size, or shape. Scales draw a legend or axes, which provide an inverse mapping to make it possible to read the original data values from the graph.
- A coordinate system, coord for short, describes how data coordinates are mapped to the plane of the graphic. It also provides axes and gridlines to make it possible to read the graph.
- A faceting specification describes how to break up the data into subsets and how to display those subsets as small multiples. This is also known as conditioning or latticing/trellising.



Business Intelligence & Exploratory Analytics

- Traditionally focused on providing dashboards for KPI performance indicators
- Increasing number of tools expand from static reports to exploratory dashboards
- Support for descriptive modeling, where the primary purpose of the model is not to estimate a value but instead to gain insight into the underlying phenomenon or process
- Examples:
  - Tableau
  - QlikView

Tableau.

- Business Intelligence Tools developed as commercialized product from the Polaris Project
- Based on Grammar of Graphics (Wilkinson, 2005)
- Objectives:
  - **Data-dense displays:** The databases typically contain a large number of records and dimensions. Analysts need to be able to create visualizations that will simultaneously display many dimensions of large subsets of the data.
  - **Multiple display types:** Analysis consists of many different tasks such as discovering correlations between variables, finding patterns in the data, locating outliers and uncovering structure. An analysis tool must be able to generate displays suited to each of these tasks.
  - **Exploratory interface:** The analysis process is often an unpredictable exploration of the data. Analysts must be able to rapidly change what data they are viewing and how they are viewing that data.
- Three graph families by the type of fields assigned to their axes:
  - Ordinal-Ordinal
  - Ordinal-Quantitative
  - Quantitative-Quantitative

Polaris: A System for Query, Analysis and Visualization of Multidimensional Relational Databases

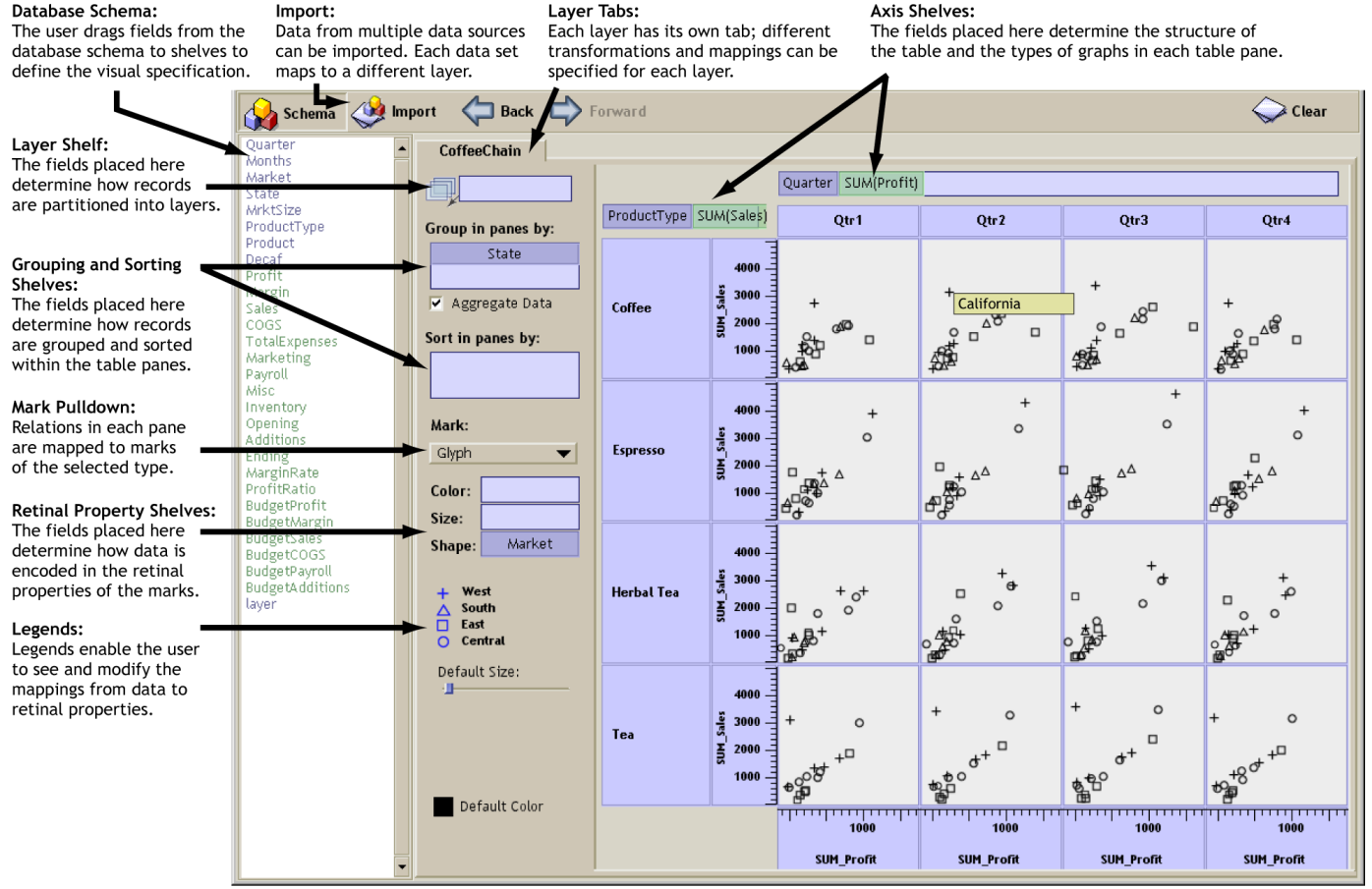


Figure 1: *The Polaris user interface. Analysts construct table-based displays of relational data by dragging fields from the database schema onto shelves throughout the display. A given configuration of fields on shelves is called a visual specification. The specification unambiguously defines the analysis and visualization operations to be performed by the system to generate the display.*

property	marks	ordinal/nominal mapping	quantitative mapping
shape	glyph	○ □ + △ S U	
size	rectangle, circle, glyph, text		
orientation	rectangle, line, text		
color	rectangle, circle, line, glyph, y-bar, x-bar, text, gantt bar		min max

Figure 4: *The different retinal properties that can be used to encode fields of the data and examples of the default mappings that are generated when a given type of data field is encoded in each of the retinal properties.*

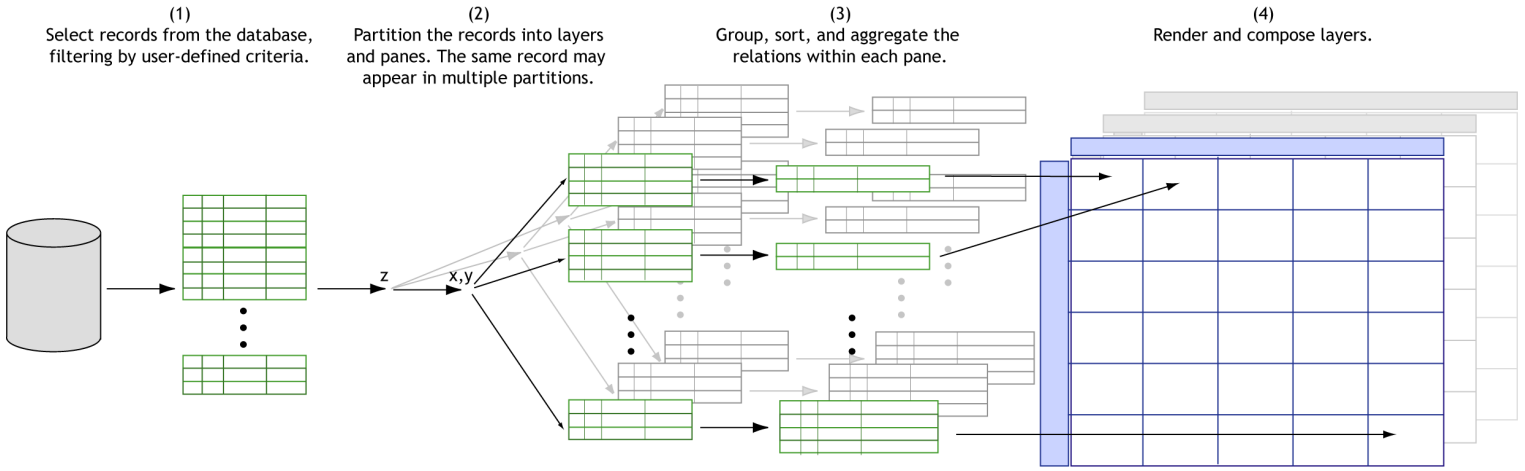
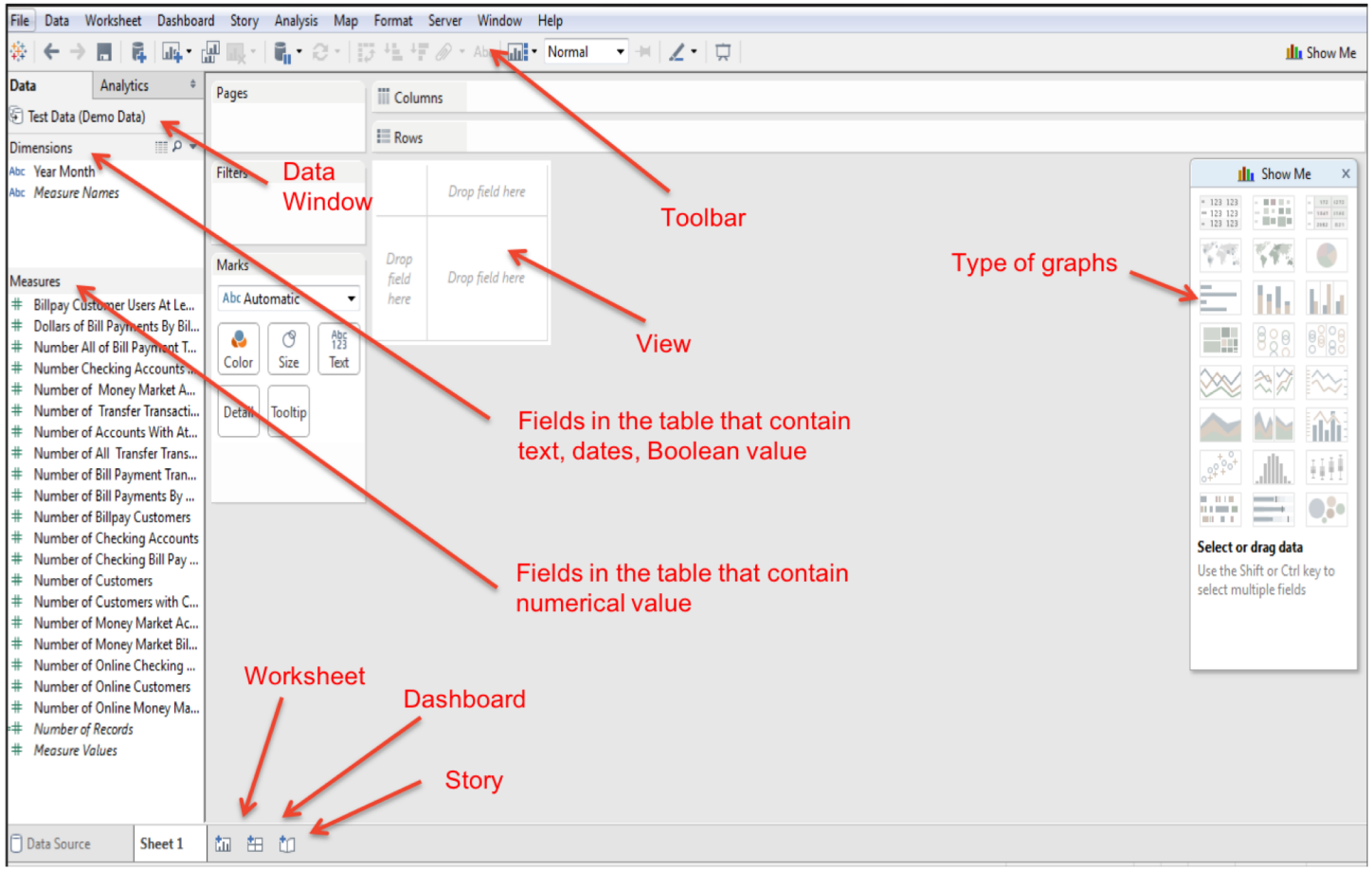


Figure 5: *The transformations and data flow within Polaris. The visual specification generates queries to the database to select subsets of the data for analysis, then to filter, sort, and group the results into panes, and then finally to group, sort and aggregate the data within panes.*



https://community.tableau.com/docs/DOC-9644

Python Matplotlib

matplotlib Version 2.1.2

home | examples | tutorials | pyplot | docs »

Introduction

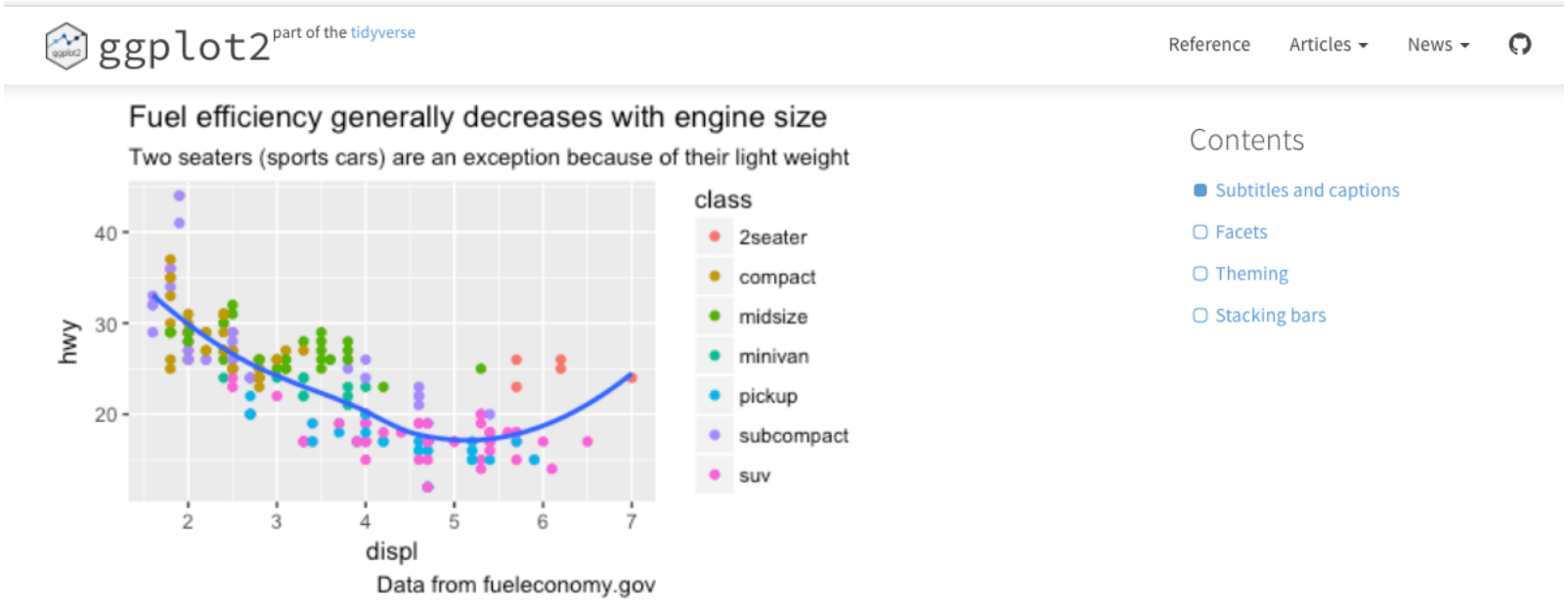
Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shell, the jupyter notebook, web application servers, and four graphical user interface toolkits.

Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, errorcharts, scatterplots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the `pyplot` module provides a MATLAB-like interface, particularly when combined with `IPython`. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

https://matplotlib.org/

R GGPlot



http://ggplot2.tidyverse.org/articles/releases/ggplot2-2.2.0.html

Wickham: ggplot2: Elegant Graphics for Data Analytics, Springer, 2009