UNIVERSITY OF CALIFORNIA
UCRIVERSIDE

UNIVERSITY OF CALIFORNIA
UCRIVERSIDE

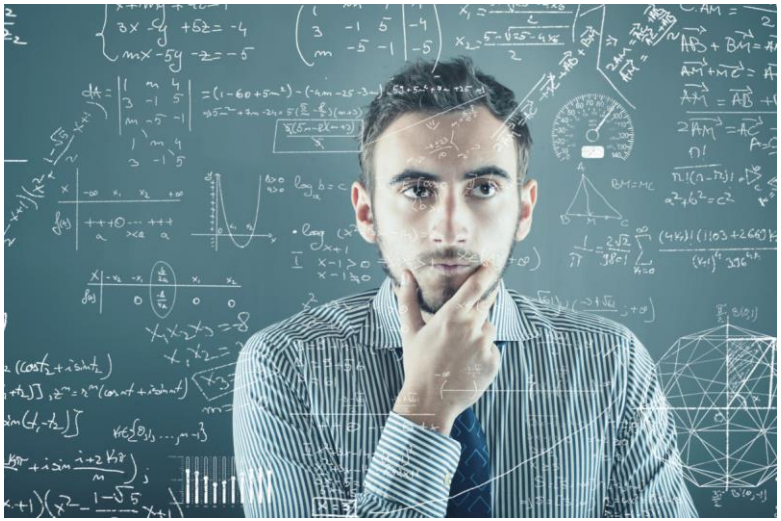
Data Science and Exploration using Big-data Tools

Ahmed Eldawy

Computer Science and Engineering

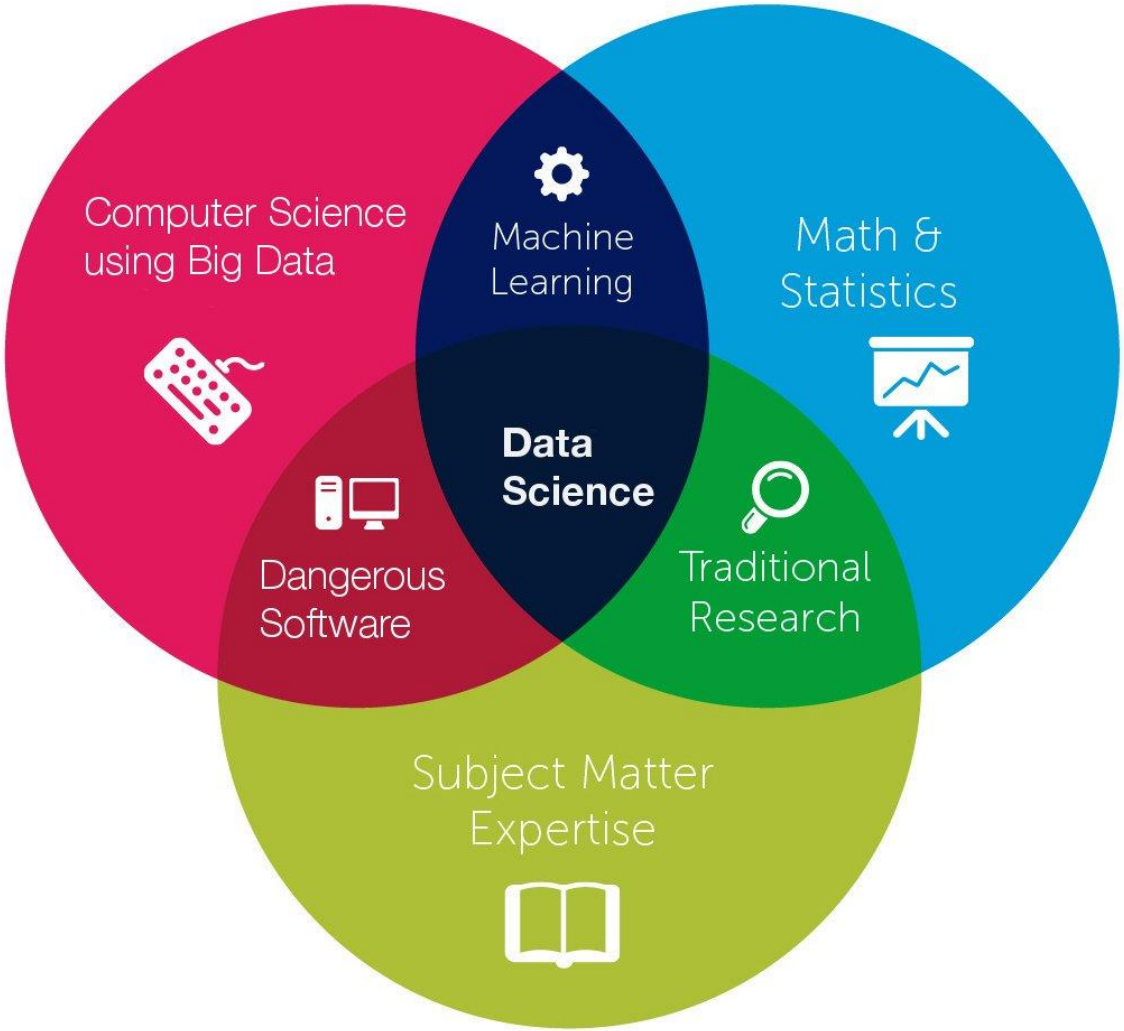
Data Science

- Data is the new gold
- Data science is the power of extracting meaningful knowledge from data
- Becoming increasingly important due to the abundance of data



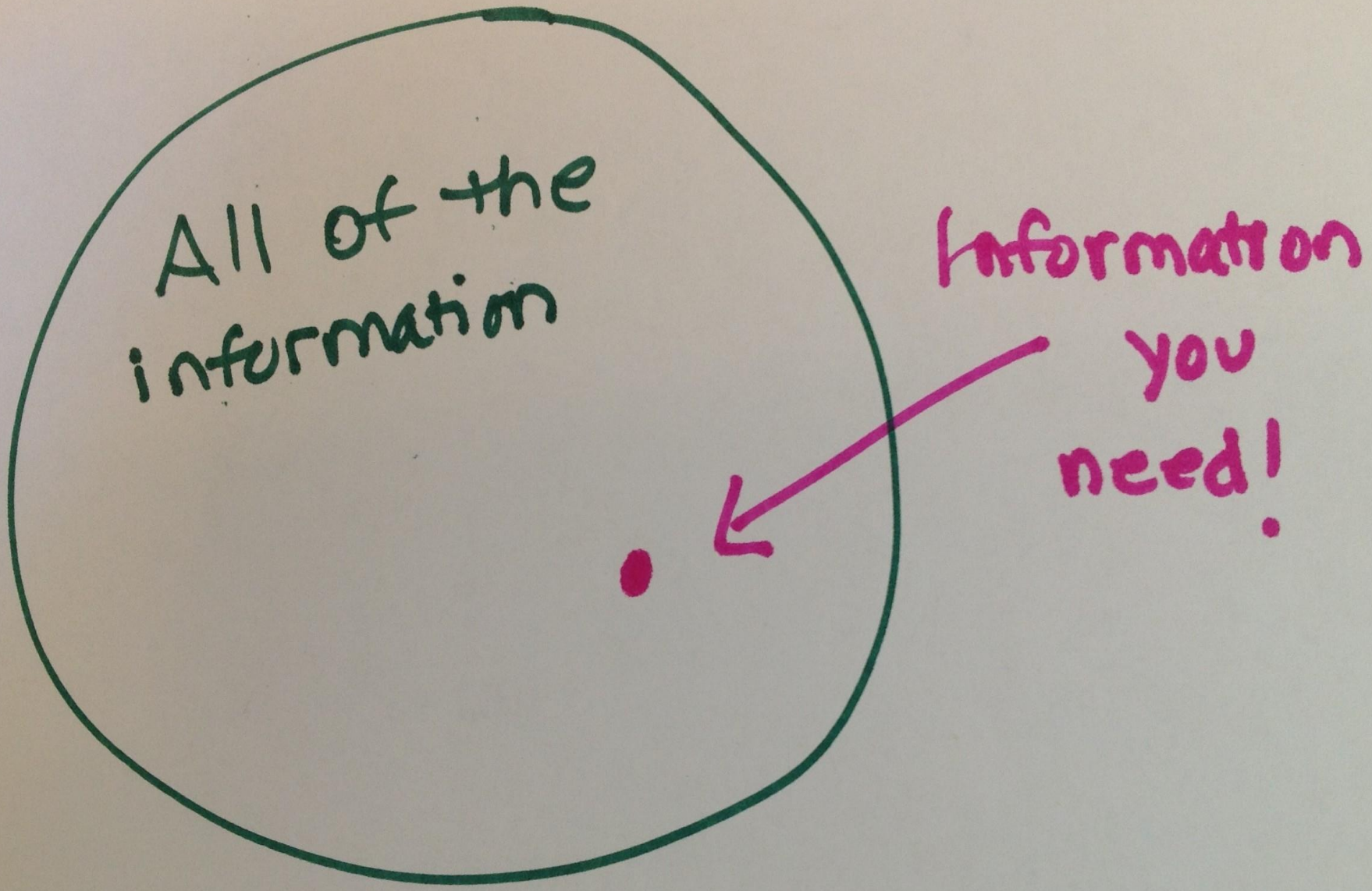
Data Collection

Data Science

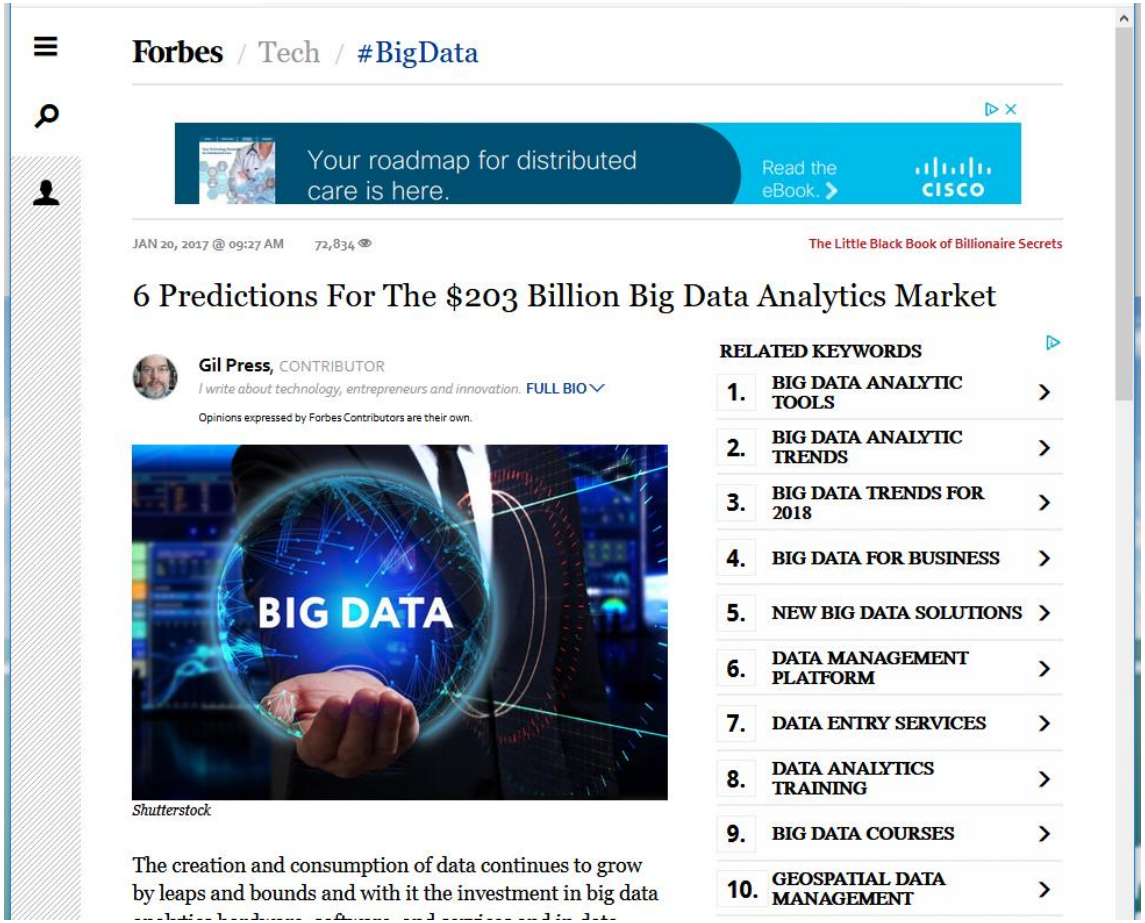


https://mashimo.wordpress.com/2016/05/28/big-data-data-science-and-machine-learning-explained/

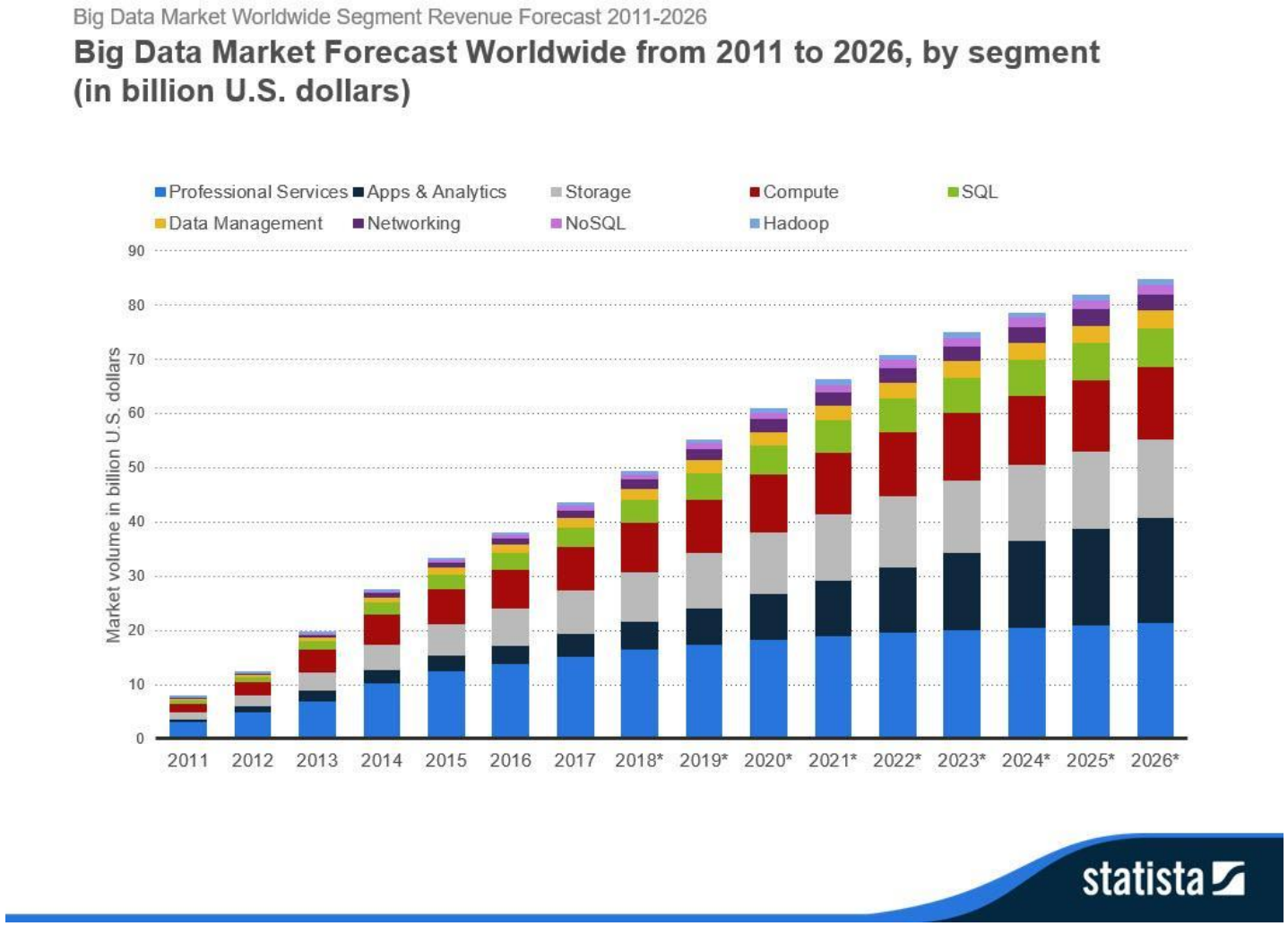

Big Data
Straight Ahead


All of the information
Information you need!

The Market of Big Data

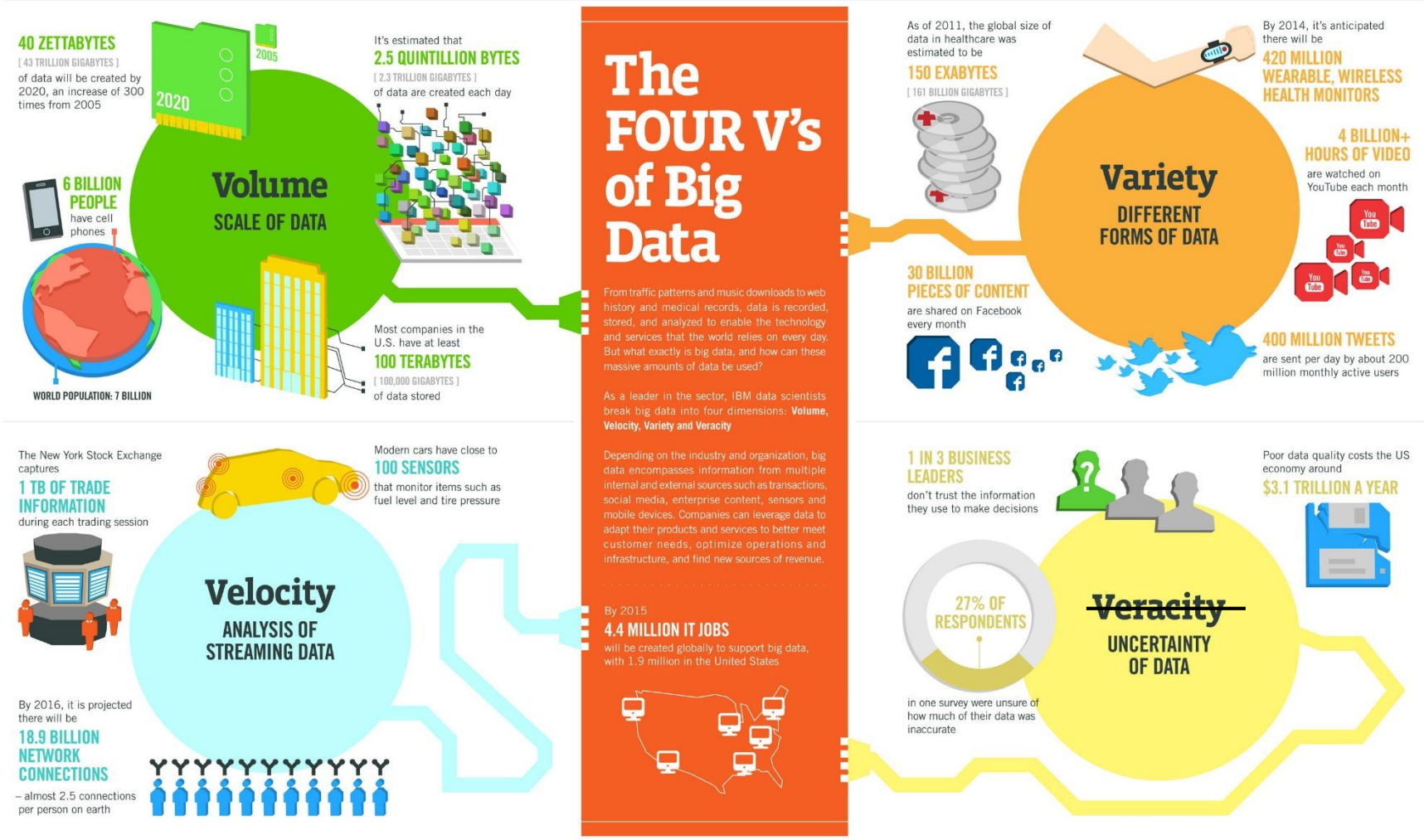


Job Market



https://www.techicy.com/5-best-programming-languages-to-watch-out-in-2019-for-data-science.html

~~Four~~ Three V's of Big Data



Big Data Vs Big Computation

- Full scans (e.g., log processing)
- Range scans
- Point lookups
- Iterations
- Joins (self, binary, or multiway)
- Proximity queries
- Closures and graph traversals
- **Big Data:** The computation is too expensive to be processed using the common tools

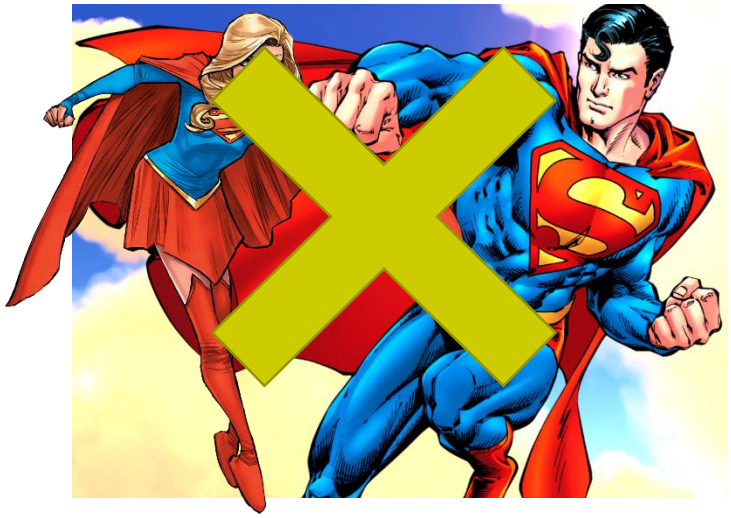
Landscape of Big Data

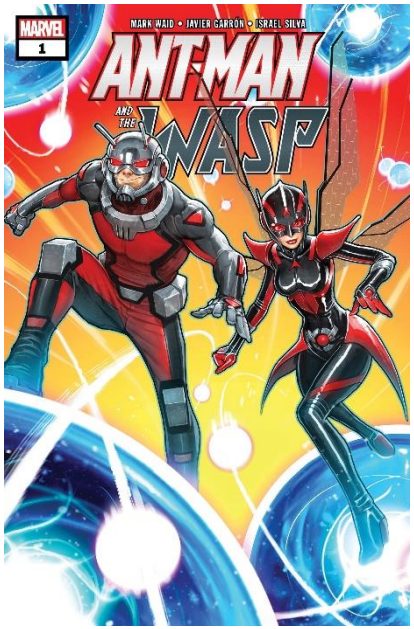
- What is currently known as big data?
  - Distributed computing
  - Horizontal scaling
  - Fault tolerance
  - In-situ computation.
- No one-size fits all
  - There are many solutions and your choice depends on what you need
- You need to have basic knowledge that allows you to explore new big data systems

Big-data Expert

- Understand how the big-data platforms really work
- Control those thousands of processors efficiently to carry out your task

Superhero



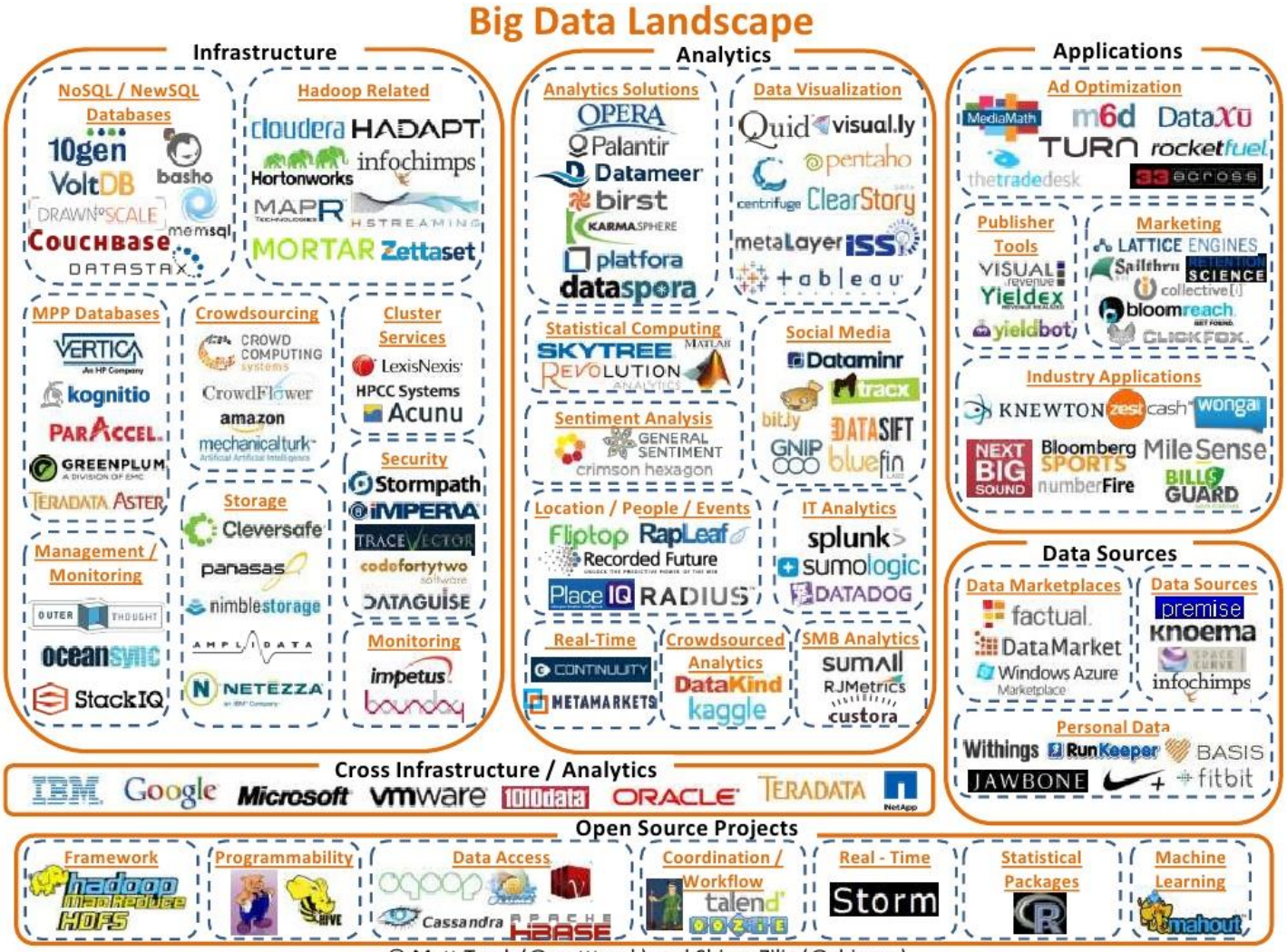

MARVEL
1
MARK WAID • JAVIER GARRÓN • ISRAEL SILVA
ANT-MAN
AND THE
WASP

Ant-Man/Wasp

Get smaller to understand how ants work and what they are capable of.

Use this knowledge to control thousands of ants and do amazing things!

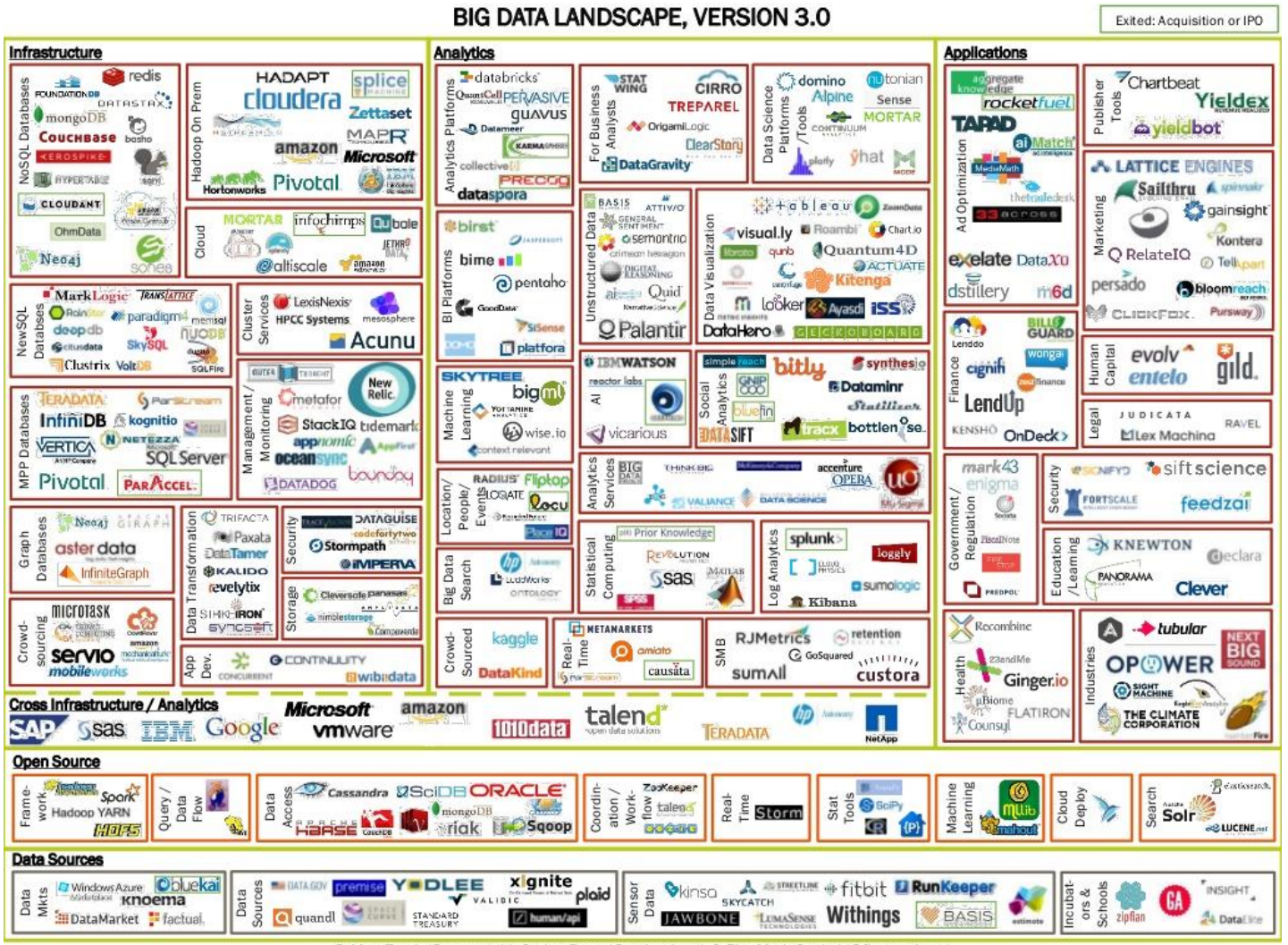
Big Data Landscape 2012



© Matt Turck (@mattturck) and ShivonZilis (@shivonz)

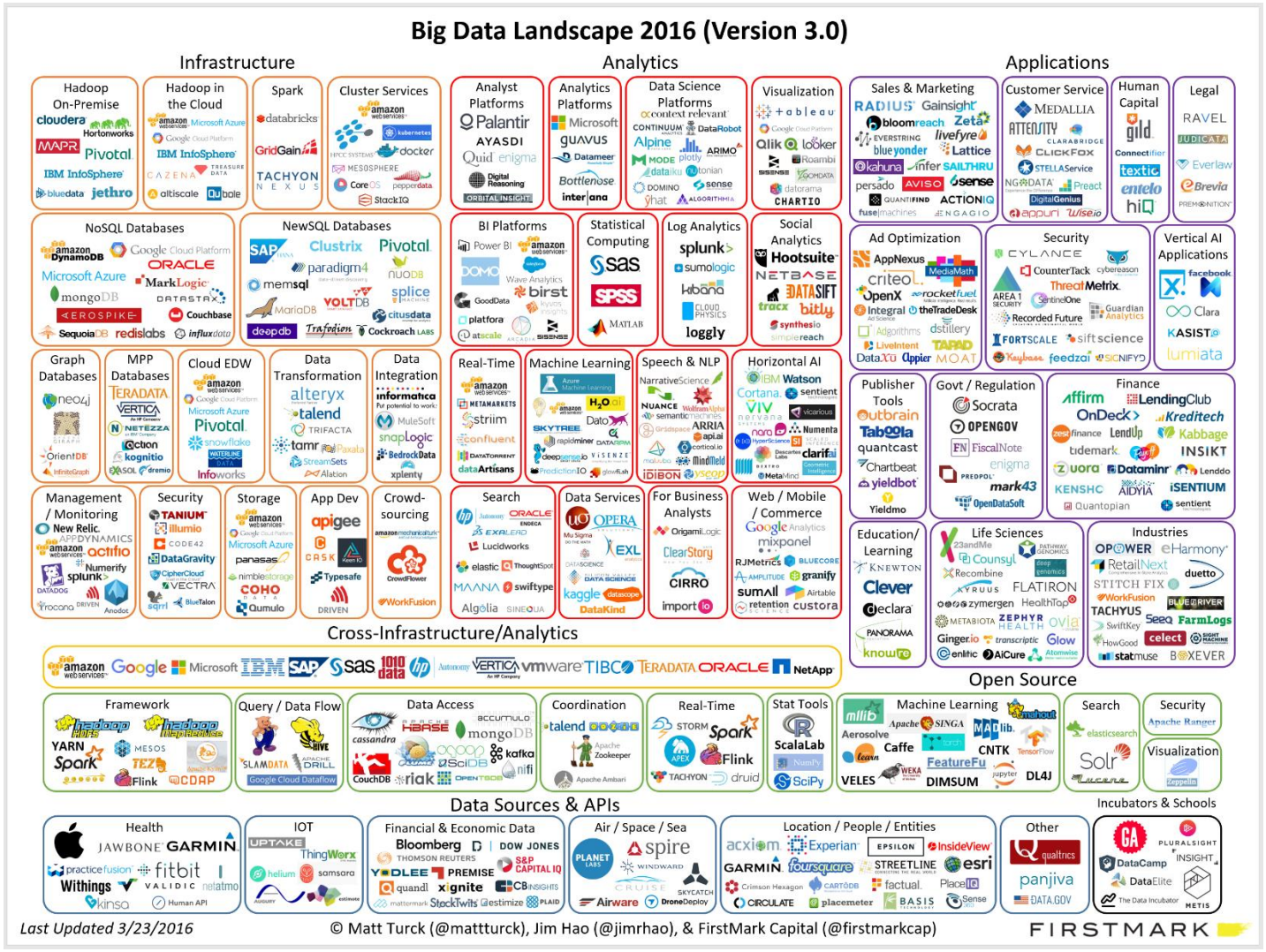
http://mattturck.com/2012/06/29/a-chart-of-the-big-data-ecosystem/

Big Data Landscape 2014



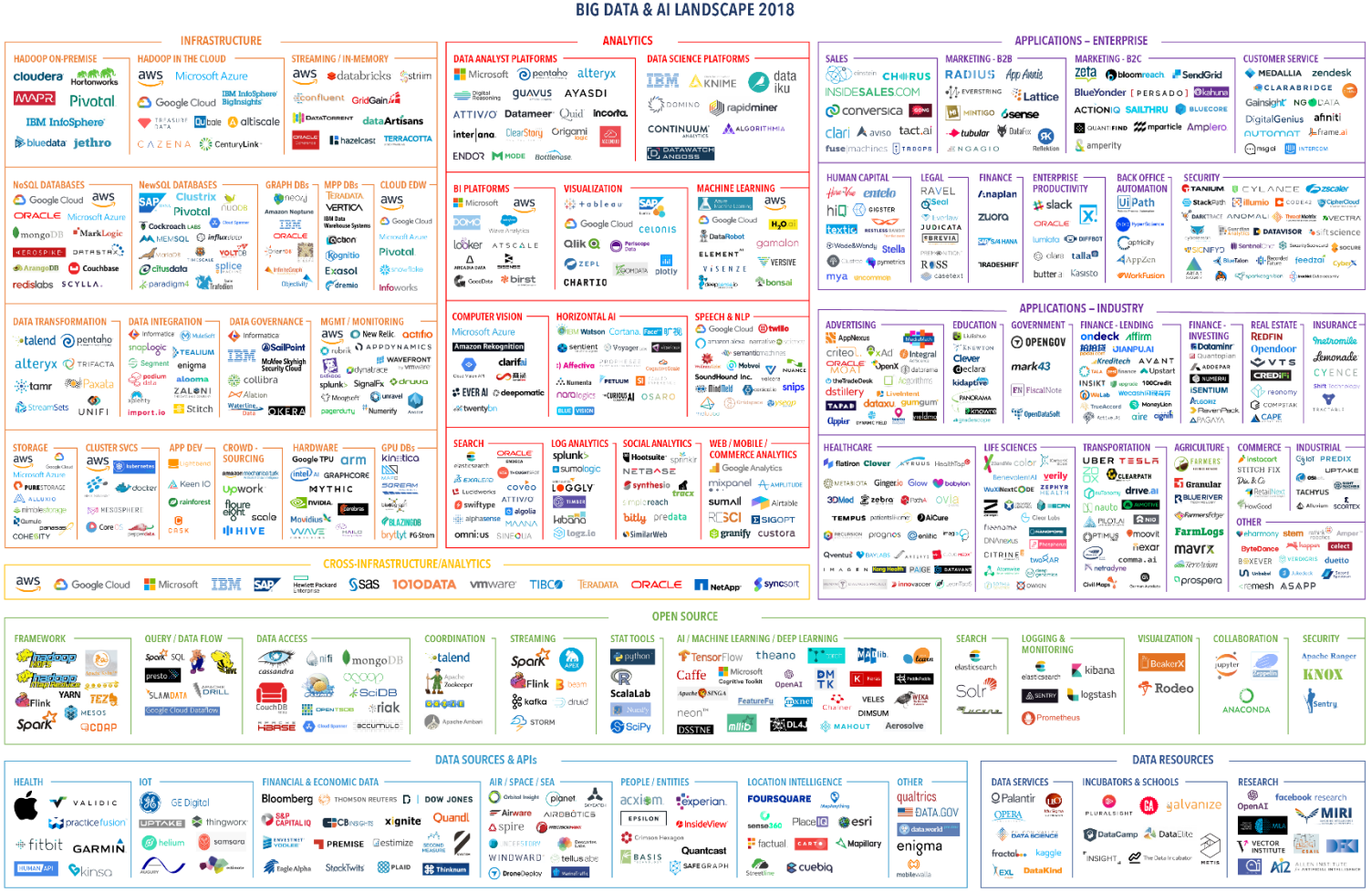
http://mattturck.com/2014/05/11/the-state-of-big-data-in-2014-a-chart/

Big Data Landscape 2016



http://mattturck.com/2016/02/01/big-data-landscape/

Big Data Landscape 2018



V1 – Last updated 6/19/2018

FIRSTMARK
EARLY STAGE VENTURE CAPITAL

Components of Big Data

Big-data Libraries
MLlib (Machine Learning), GraphX

High-level Languages
SparkSQL, Pig, SQL++, HiveQL

Distributed Computing
MapReduce (Hadoop and Google), Resilient Distributed Dataset (Spark), Hyracks (AsterixDB)

Big Data Distributed Storage
Hadoop Distributed File System, Cloud storage systems (Amazon S3 and Google File System), Key-value stores

Cloud Services
Amazon Web Services, Microsoft Azure, and Google Cloud Platform

Coordination/Cluster Management
Oozie, Yarn, Kubernetes

Big-data Storage

Big-data Libraries

High-level Languages

Distributed Computing

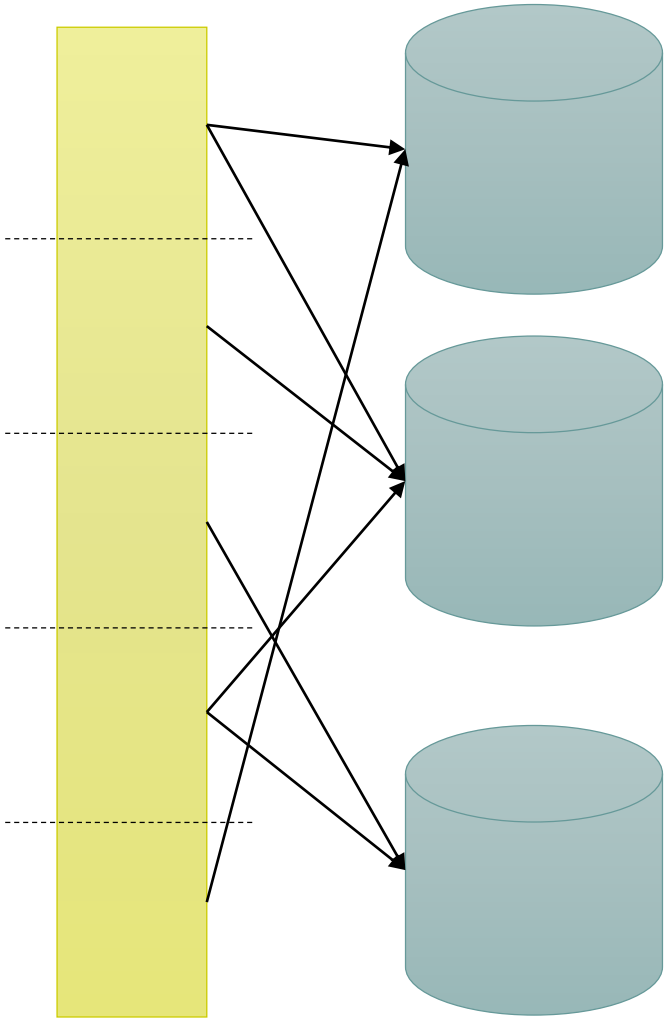
Big Data Storage

Cloud Services

Coordination/
Cluster
Management

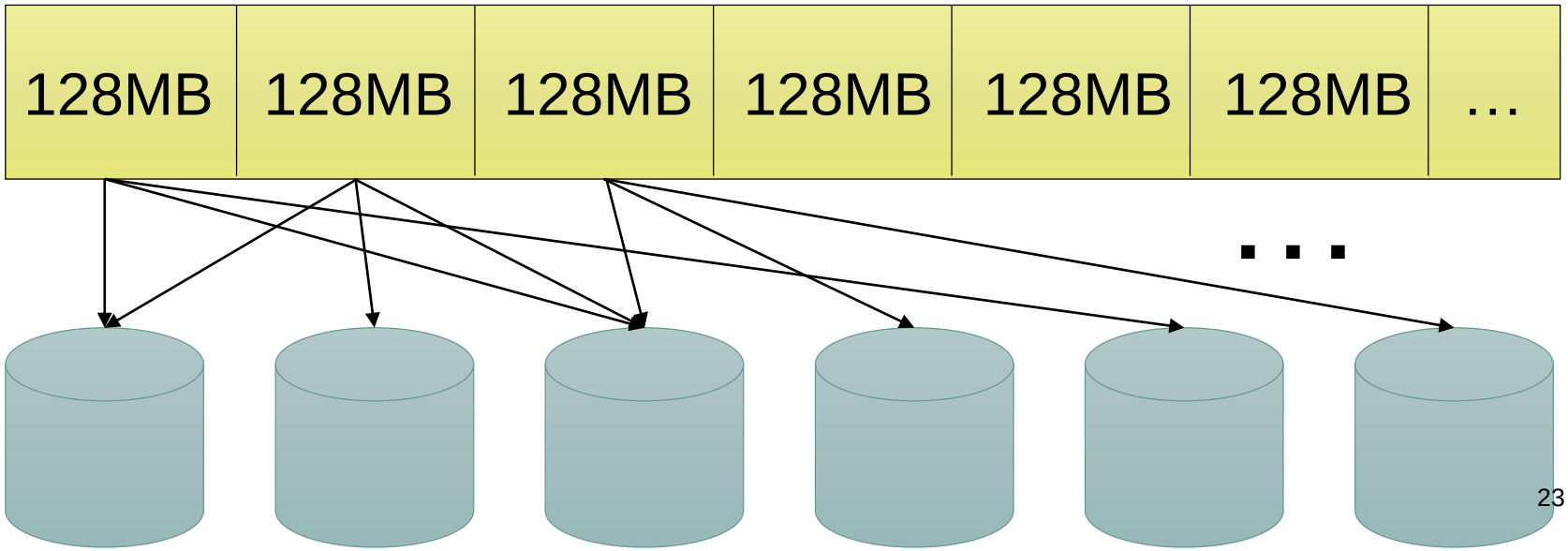
Storage of Big Data

- Data is growing faster than Moore's Law
- Too much data to fit on a single machine
- Partitioning
- Replication
- Fault-tolerance



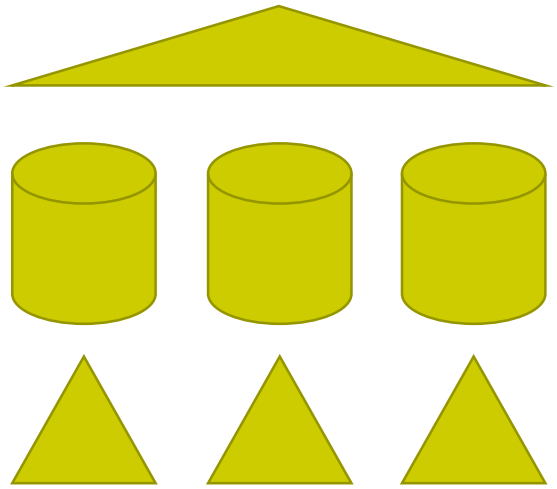
Hadoop Distributed File System (HDFS)

- The most widely used distributed file system
- Fixed-sized partitioning
- 3-way replication
- Write-once read-many
- See also: GFS, Amazon S3, Azure Blob Store



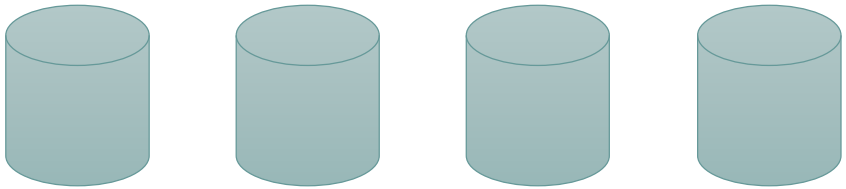
Indexing

- Data-aware organization
- Global Index **partitions** the records into blocks
- Local Indexes organize the records in a partition
- Challenges:
  - Big volume
  - HDFS limitation
  - New programming paradigms
  - Ad-hoc indexes

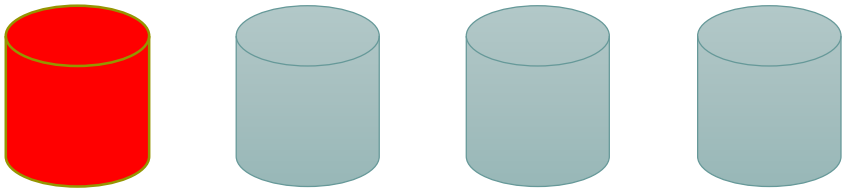


Fault Tolerance

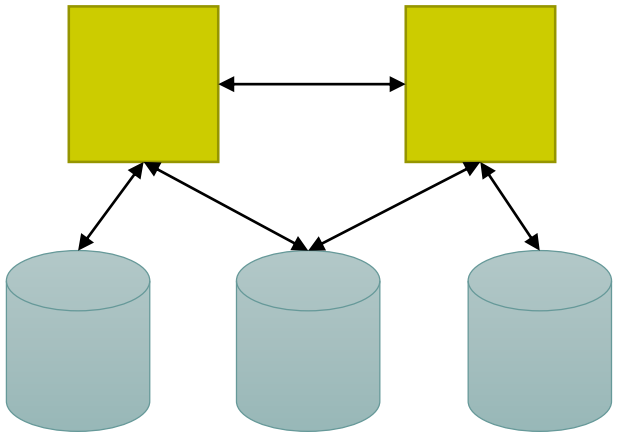
- Replication



- Redundancy

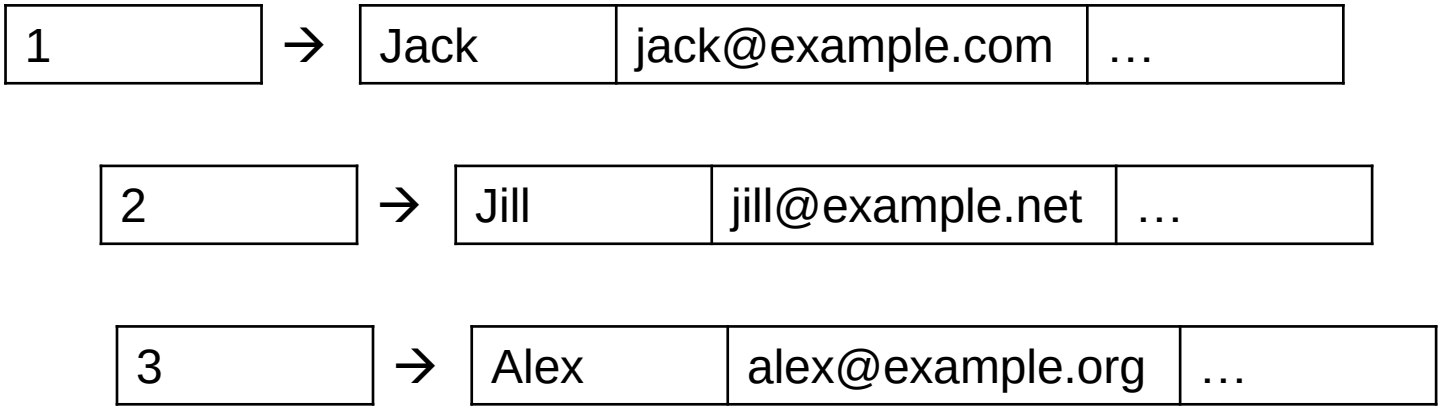


- Multiple masters

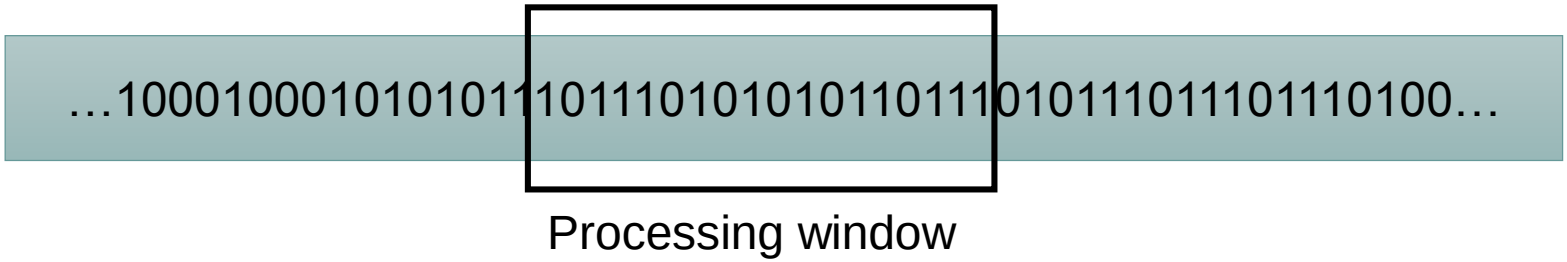


Key-value Stores

ID	Name	Email	…
1	Jack	jack@example.com	
2	Jill	jill@example.net	
3	Alex	alex@example.org	



Streaming



- Sub-second latency for queries
- One scan over the data
- (Partial) preprocessing
- Continuous queries
- Eviction strategies
- In-memory indexes

Structured/Semi-structured

ID	Name	Email	…
1	Jack	jack@example.com	
2	Jill	jill@example.net	
3	Alex	alex@example.org	

Document 1

{ “id”: 1, “name”:”Jack”, “email”: “jack@example.com”, “address”: {“st
“900 university ave”, “city”: “Riverside
“CA”}, “friend_ids”: [3, 55, 123]}

Document 2

{ “id”: 2, “name”: “Jill”, “email”: “jill@example.net”, “hobbies”: [“hiking”, “cooking”]}

Distributed Computing

Big-data Libraries

High-level Languages

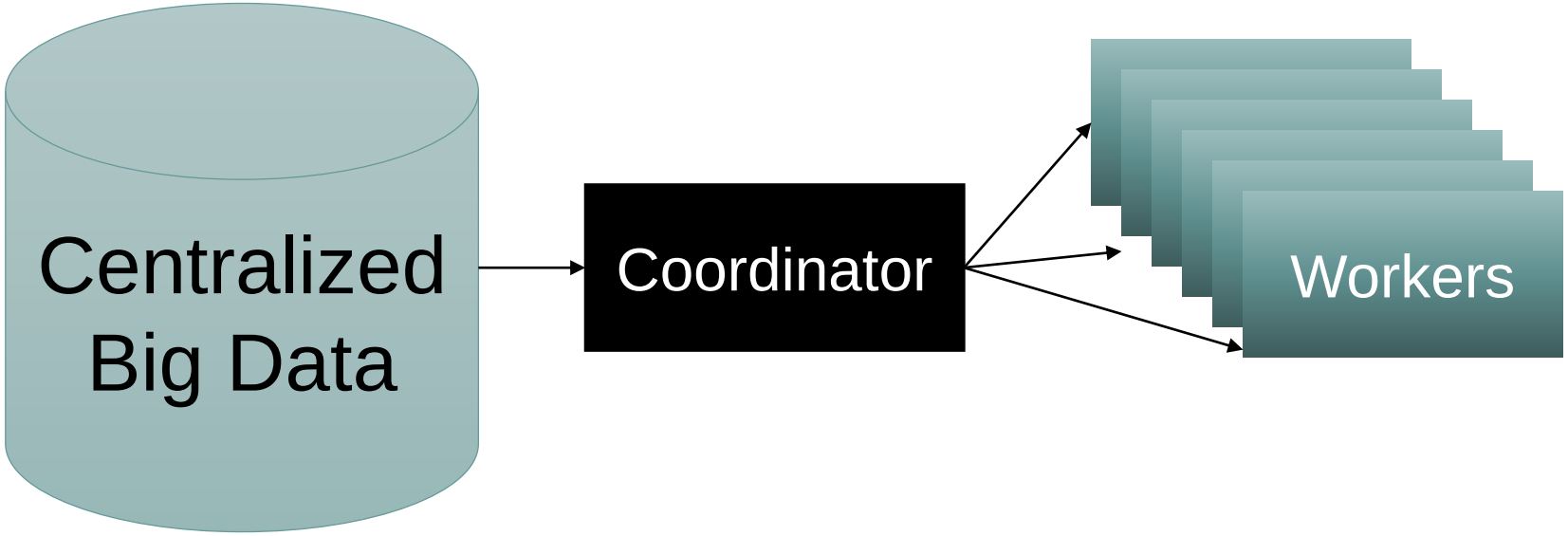
Distributed Computing

Big Data Storage

Cloud Services

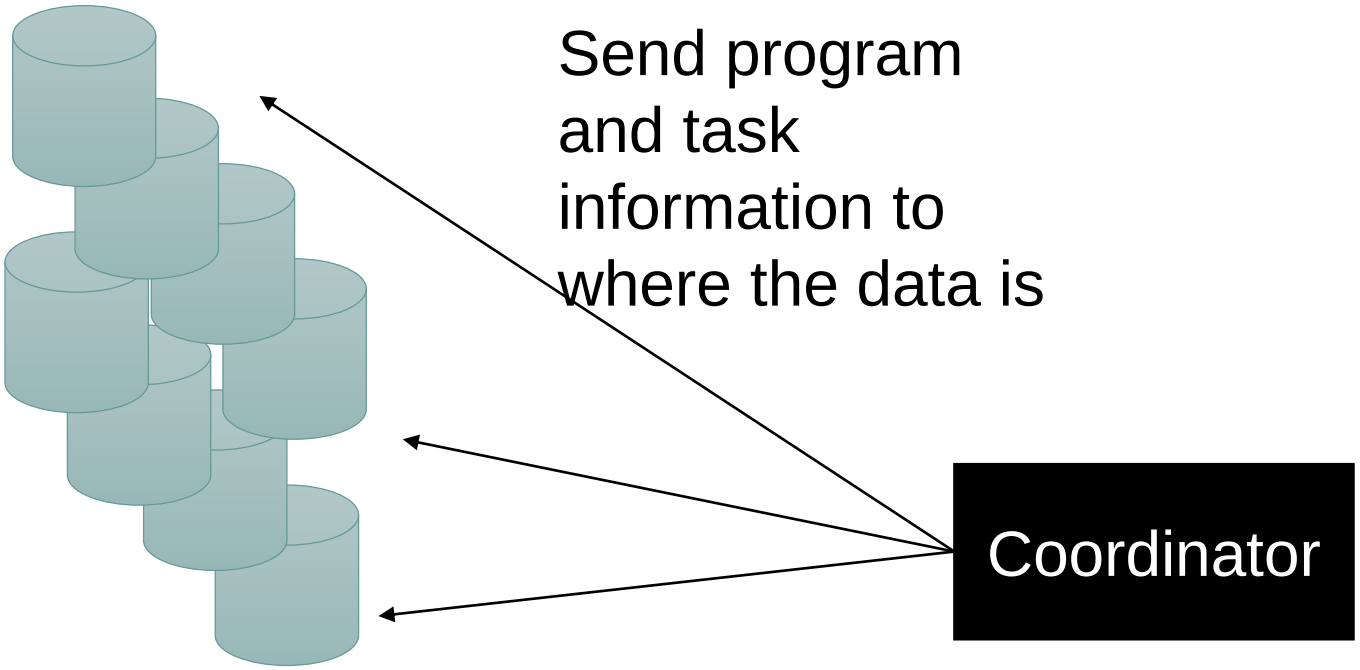
Coordination/
Cluster
Management

Traditional Distributed Computing



Ship data to computation paradigm
e.g., High performance computing (HPC)

Big-data Computing



Ship compute to data paradigm

MapReduce

- Invented by Google
- Implemented in the open-source Hadoop system
- A functional programming technique
- The program is expressed in two functions, Map and Reduce
- Each function processes one or a few records at a time
- The functions are applied in parallel to trillions of records

Example: Word Count

if you cannot fly, then run.
If you cannot run, then walk.
If you cannot walk, then crawl
But whatever you do you have to keep moving forward

Input text file

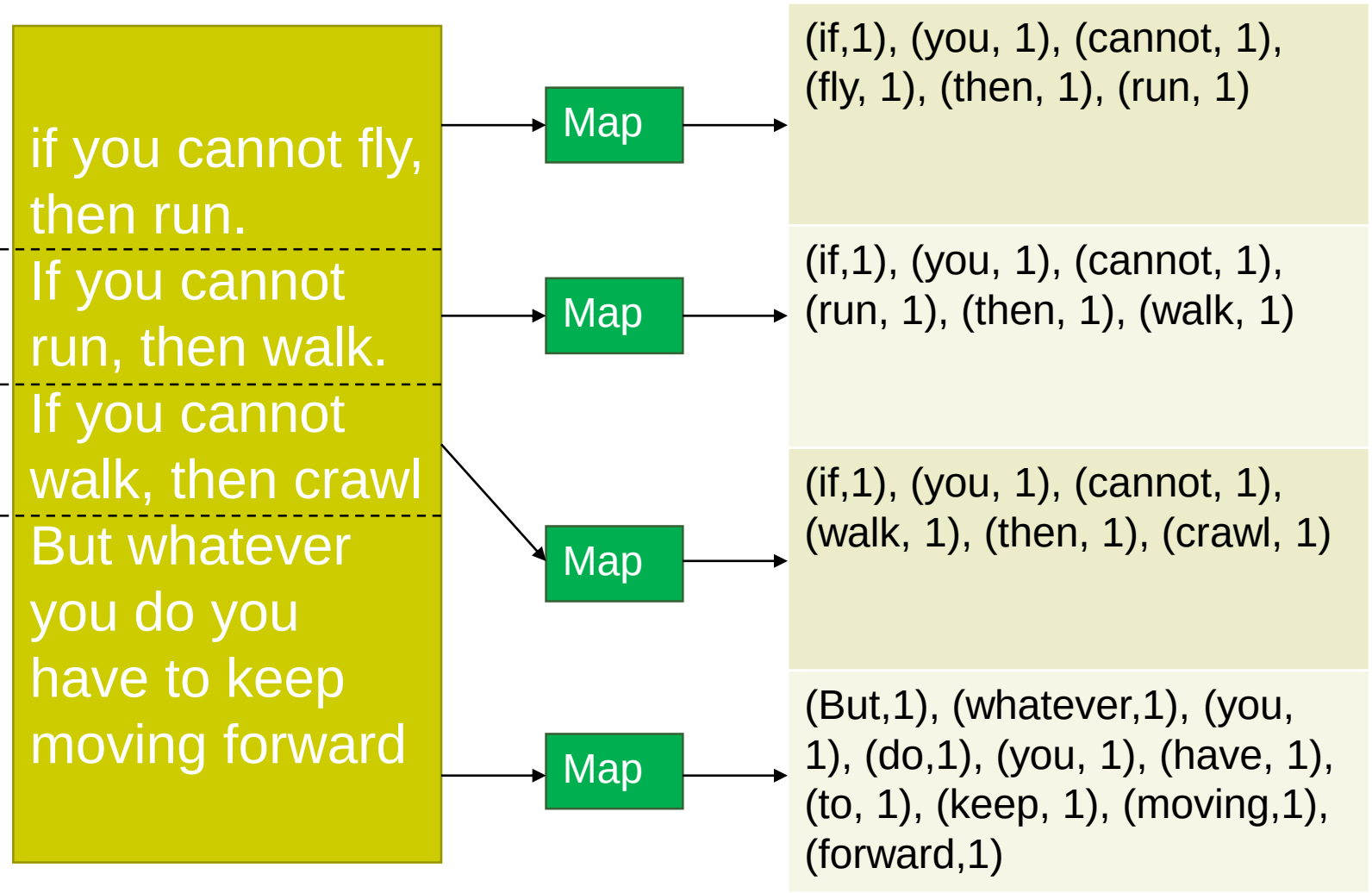
```
Map(line) {
  split line into words
  for each word w
     output (w,1)
}
```

```
Reduce(w, c[]) {
  s = Sum(c)
  output(w, s)
}
```

you: 5
cannot: 3
walk: 2
if: 3
…

Output

Example: Word Count (Map)



Input text file

Map output

Example: Word Count (Shuffle)

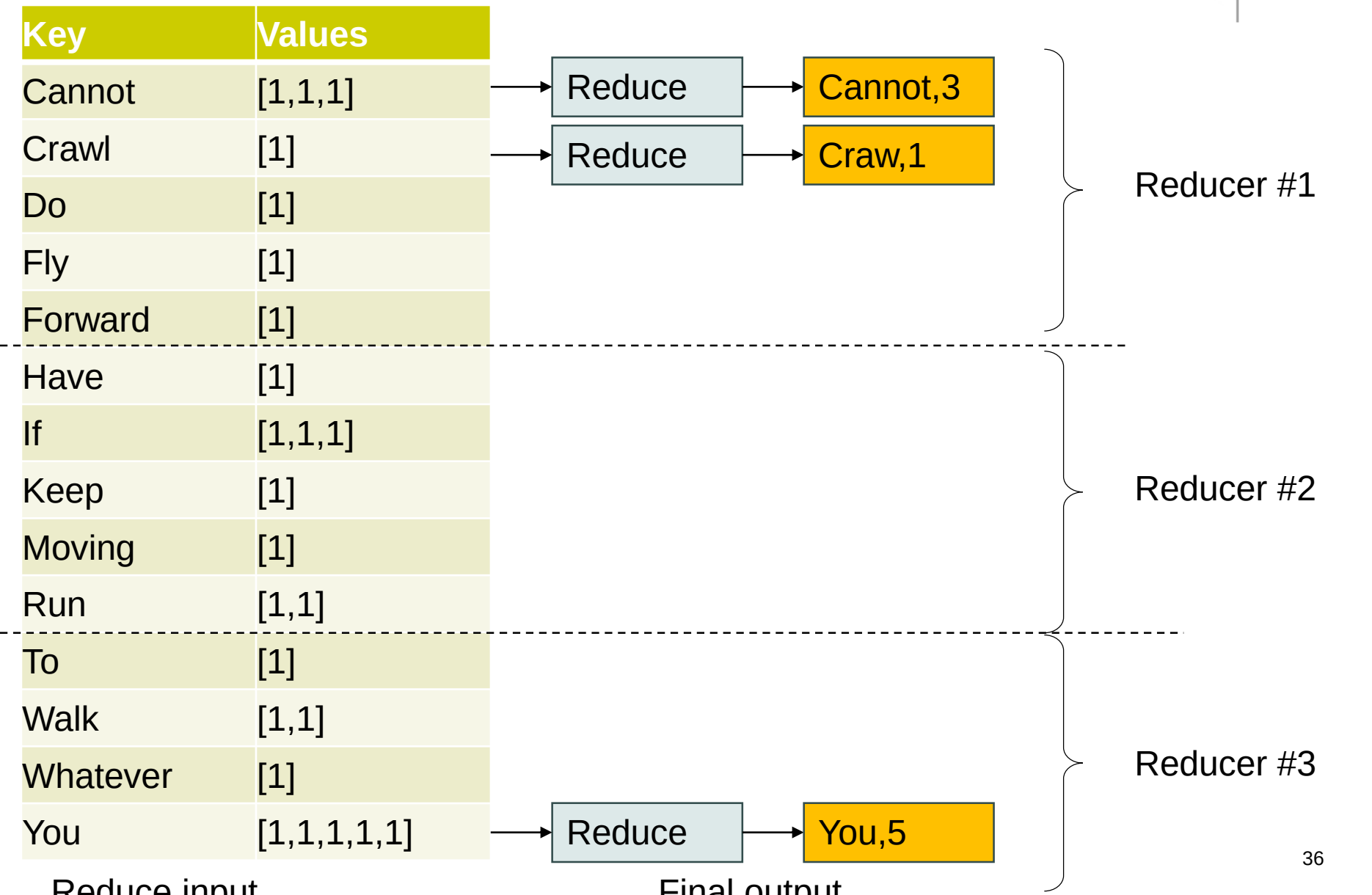
(if,1), (you, 1), (cannot, 1), (fly, 1), (then, 1), (run, 1)
(if,1), (you, 1), (cannot, 1), (run, 1), (then, 1), (walk, 1)
(if,1), (you, 1), (cannot, 1), (walk, 1), (then, 1), (crawl, 1)
(But,1), (whatever,1), (you, 1), (do,1), (you, 1), (have, 1), (to, 1), (keep, 1), (moving,1), (forward,1)

Map output

Group by key

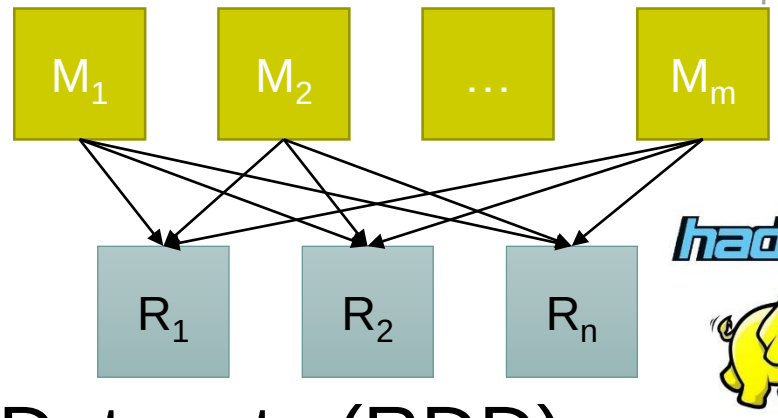
Key	Values
Cannot	[1,1,1]
Crawl	[1]
Do	[1]
Fly	[1]
Forward	[1]
Have	[1]
If	[1,1,1]
Keep	[1]
Moving	[1]
Run	[1,1]
To	[1]
Walk	[1,1]
Whatever	[1]
You	[1,1,1,1,1]

Example: Word Count (Reduce)

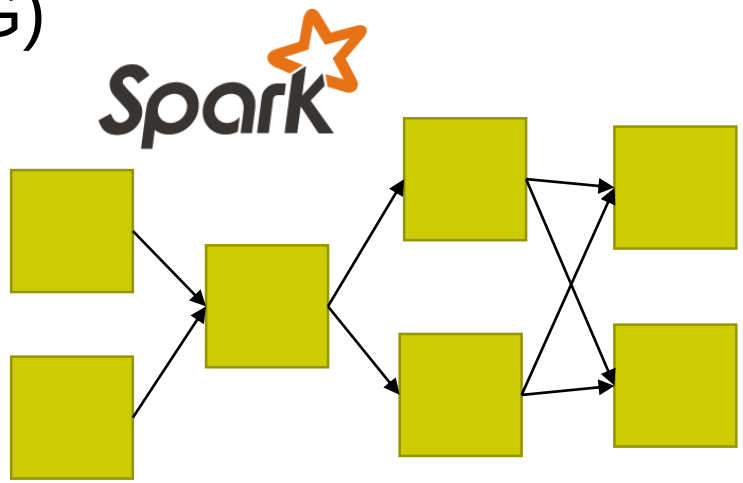


Task Execution

- MapReduce
  - Map-Shuffle- Reduce
  - Resiliency through materialization



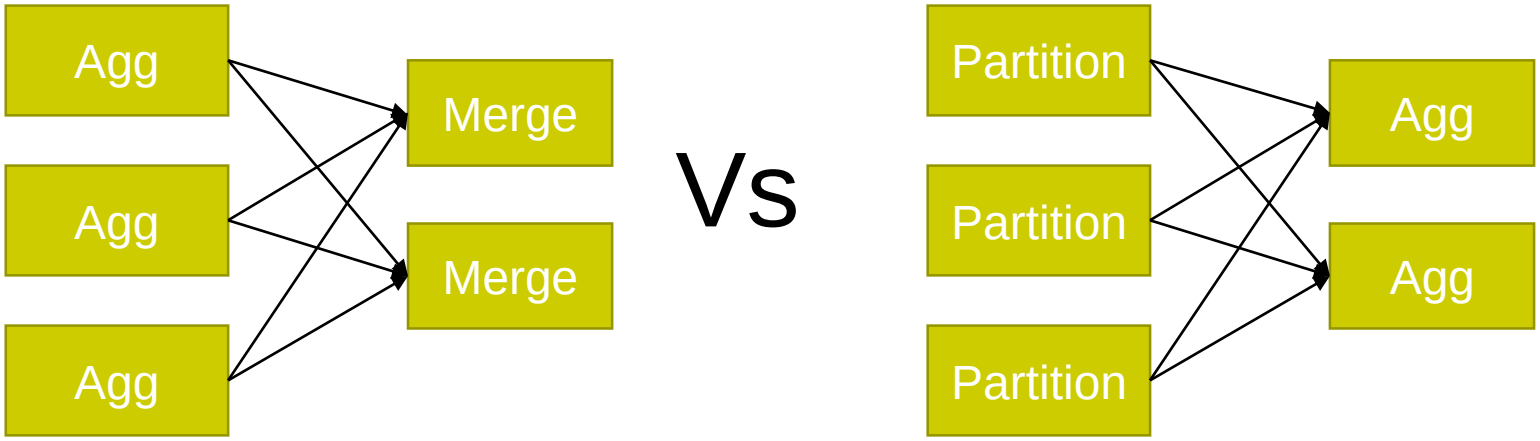
- Resilient Distributed Datasets (RDD)
  - Directed-Acyclic-Graph (DAG)
  - In-memory processing
  - Resiliency through lineages



- Hyracks
- Stragglers
- Load balance

Query Optimization

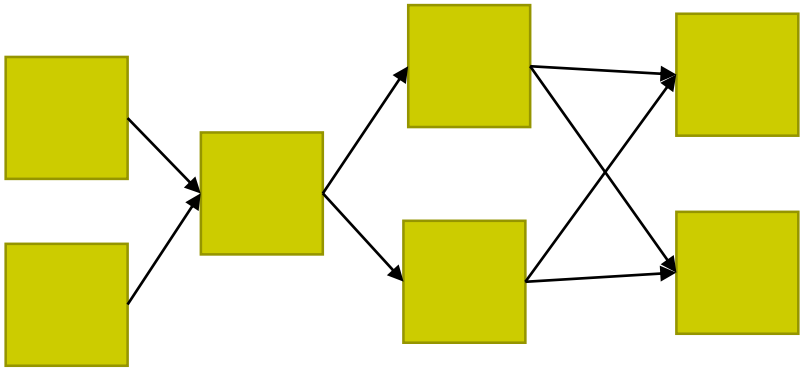
- Finding the most efficient query plan
- e.g., grouped aggregation



- Cost model (CPU – Disk – Network)

Provenance

- Debugging in distributed systems is painful



- We need to keep track of transformations on each record

High-level Languages

Big-data Libraries

High-level Languages

Distributed Computing

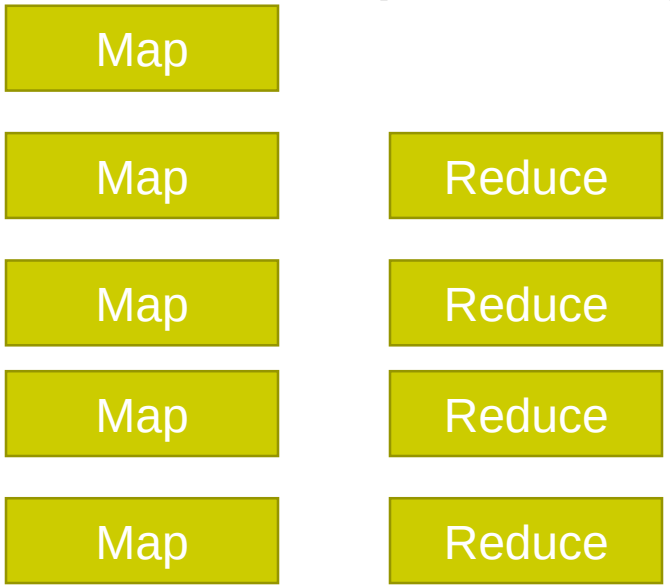
Big Data Storage

Cloud Services

Coordination/
Cluster
Management

Declarative MapReduce

- MapReduce has been used to create many reusable operators (e.g., relational operators)
- Filter
- Aggregate
- Grouped aggregated
- Equi-join
- Non-equi-join



Declarative Languages

- Describe what you want to do not how to do it
- The most popular example is SQL
- Can we compile SQL queries into MapReduce program(s)?

Pig

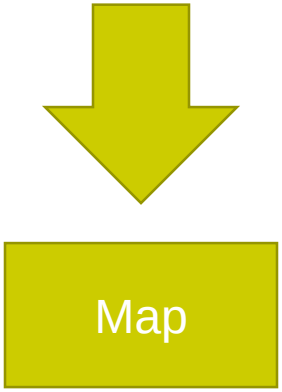
UCR

- A system built on-top of Hadoop (Now supports Spark as well)
- Provides a SQL-ETL-like query language termed Pig Latin
- Compiles Pig Latin programs into MapReduce programs

Examples

- Filter: Return all the lines that have a user-specified response code, e.g., 200.

```
log = LOAD ‘logs.csv’ USING PigStorage()
  AS (host, time, method, url, response, bytes);
ok_lines = FILTER log BY response = ‘200’;
STORE ok_lines into ‘filtered_output’;
```



Examples

- Grouped aggregate
- Find the total number of bytes per response code

```
log = LOAD ‘logs.csv’ USING PigStorage()
  AS (host, time, method, url, response, bytes: int);
grouped = GROUP log BY response;
grouped_aggregate =
    FOREACH grouped GENERATE group, SUM(bytes);
STORE grouped_aggregate into ‘grouped_output’;
```

Examples

- Grouped aggregate
- Find the average number of bytes per response code

```
log = LOAD ‘logs.csv’ USING PigStorage()
  AS (host, time, method, url, response, bytes: int);
grouped = GROUP log BY response;
grouped_aggregate =
    FOREACH grouped GENERATE group, AVG(bytes);
STORE grouped_aggregate into ‘grouped_output’;
```

Examples

- Join: Find pairs of requests that ask for the **same URL**, coming from the **same source**

```
log1 = LOAD ‘logs.csv’ USING PigStorage()
  AS (host, time, method, url, response, bytes: int);
log2 = LOAD ‘logs.csv’ USING PigStorage()
  AS (host, time, method, url, response, bytes: int);
joined = JOIN log1 BY (url, host),
              log2 BY (url, host);
```

Examples

- Join: Find pairs of requests that ask for the **same URL**, coming from the **same source** and happened within an hour of each other

```
log1 = LOAD ‘logs.csv’ USING PigStorage()
  AS (host, time, method, url, response, bytes: int);
log2 = LOAD ‘logs.csv’ USING PigStorage()
  AS (host, time, method, url, response, bytes: int);
joined = JOIN log1 BY (url, host),
              log2 BY (url, host);
filtered = FILTER joined BY
   ABS(log1::time - log2::time) < 3600000;
```

Additional Features

- Lazy execution
  - Nothing gets actually executed until the STORE command is reached
- Consolidation of map-only jobs
  - Map-only jobs (FILTER and FOREACH) can be consolidated into a next job's map function or a previous job's reduce function

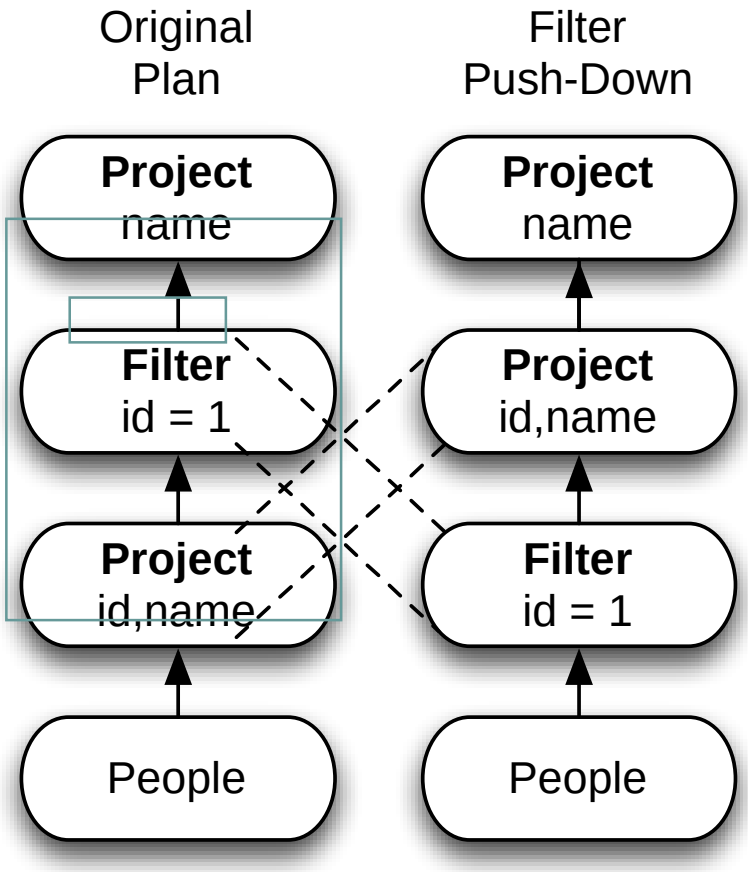
SparkSQL

- Redesigned to consider Spark query model
- Supports all the popular relational operators
- Can be intermixed with RDD operations
- Uses the Dataframe API as an enhancement to the RDD API

Dataframe = RDD + schema

Catalyst Query Optimizer

- Extensible rule-based optimizer
- Users can define their own rules



Big-data Libraries

Big-data Libraries

High-level Languages

Distributed Computing

Big Data Storage

Cloud Services

Coordination/
Cluster
Management

Specialized Big Data Systems

- The flexible big-data systems allowed developers to build many high-level applications
- Graph processing
  - Giraph
  - GraphX
- Machine Learning
  - Mahout
  - MLlib

Cloud Services

Big-data Libraries

High-level Languages

Distributed Computing

Big Data Storage

Cloud Services

Coordination/ Cluster Management

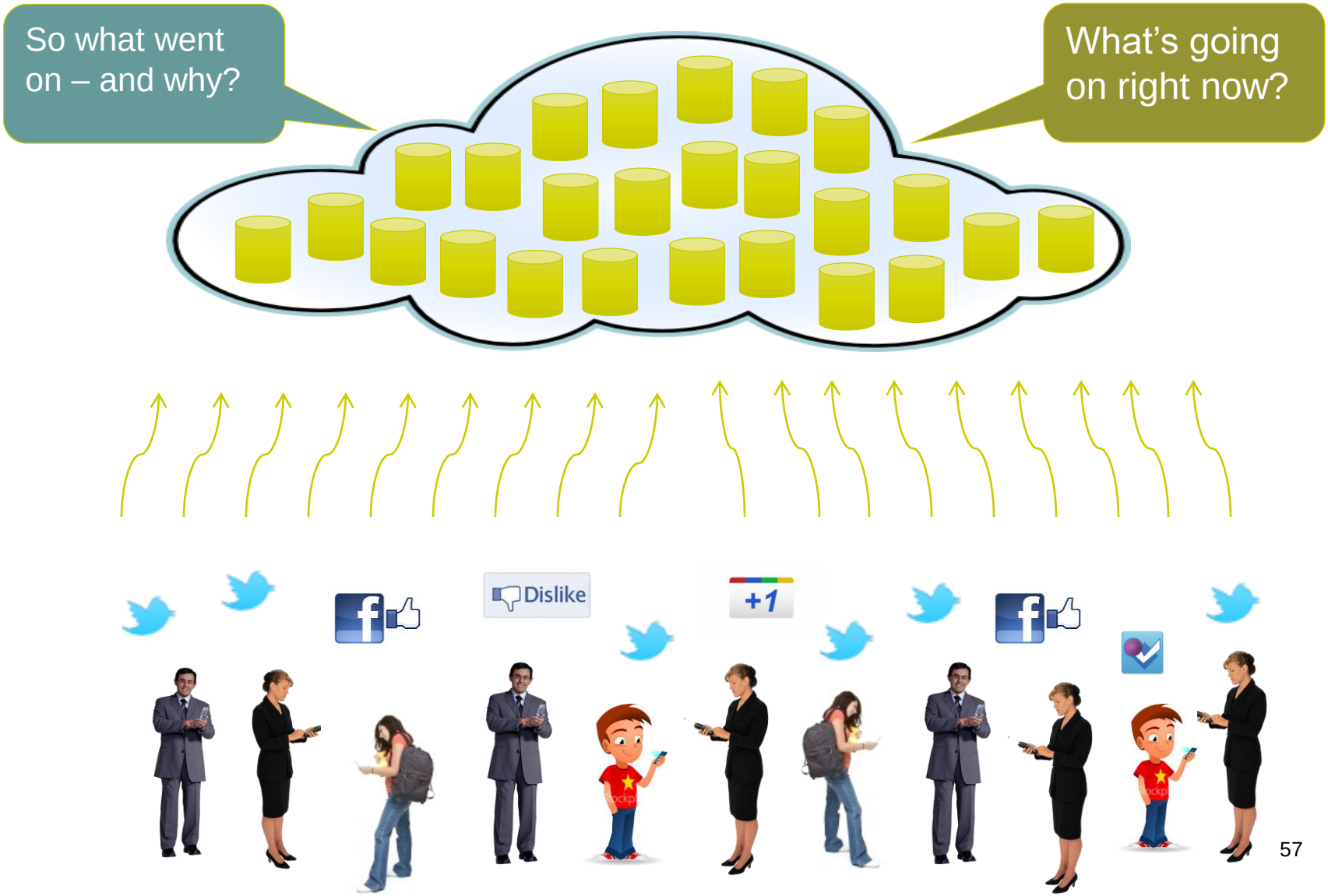
Elastic Clouds

- Clusters are expensive to buy and manage
- Machine could be idle most of the time
- Elastic clouds enables creation and destruction of on-demand clusters

Big-data on the Cloud

- Most cloud services provide ready-to-go big-data clusters, e.g., Hadoop or Spark
- You can easily tailor the hardware specifications according to your application needs, e.g., storage, memory, computing, and network bandwidth
- You only pay for the hours where the machines are running
- Make sure to terminate the machines after you are done!!!

Big Data / Web Warehousing



Also: Today's Big Data Tangle

mongoDB

Cassandra

ly scalable, eventually consistent, distributed, structured key-value store.

AsterixDB: "One Size Fits a Bunch"

BDMS Desiderata:

- Able to **manage** data
- **Flexible** data model
- Full **query** capability
- Continuous data **ingestion**
- Efficient and robust **parallel** runtime
- Cost **proportional** to task at hand
- Support "**Big Data** data types"
  - .
  - .
  - .

ASTERIX Data Model (ADM)

```
CREATE DATAVERSE TinySocial;
USE TinySocial;

CREATE TYPE GleambookUserType AS {
    id: int,
    alias: string,
    name: string,
    userSince: datetime,
    friendIds: {{ int }},
    employment: [EmploymentType]
};
```

```
CREATE TYPE EmploymentType AS {
    organizationName: string,
    startDate: date,
    endDate: date?
};

CREATE DATASET GleambookUsers
                    (GleambookUserType)
PRIMARY KEY id;
```

Highlights include:

- JSON++ based data model
- Rich type support (spatial, temporal, …)
- Records, lists, bags
- ***Open vs. closed types***

Other DDL Features

```
CREATE INDEX gbUserSinceIdx ON GleambookUsers(userSince);
CREATE INDEX gbAuthorIdx ON GleambookMessages(authorId) TYPE BTREE;
CREATE INDEX gbSenderLocIndex ON GleambookMessages(senderLocation) TYPE RTREE;
CREATE INDEX gbMessageIdx ONGleambookMessages(message) TYPE KEYWORD;
//---------- and also ----------
CREATE TYPE AccessLogType AS CLOSED
   { ip: string, time: string, user: string, verb: string, `path`: string,  stat: int32, size: int32 };
CREATE EXTERNAL DATASET AccessLog(AccessLogType) USING localfs
   (("path"="localhost:///Users/mikejcarey/extdemo/accesses.txt"),
     ("format"="delimited-text"), ("delimiter"="|"));

CREATE FEED myMsgFeed USING socket_adapter
    (("sockets"="127.0.0.1:10001"), ("address-type"="IP"),
     ("type-name"="GleambookMessageType"), ("format"="adm"));
CONNECT FEED myMsgFeed TO DATASET GleambookMessages;
START FEED myMsgFeed;
```

External data highlights:

- Equal opportunity access
- Feeds to “keep everything!”
- Ingestion, *not* streams

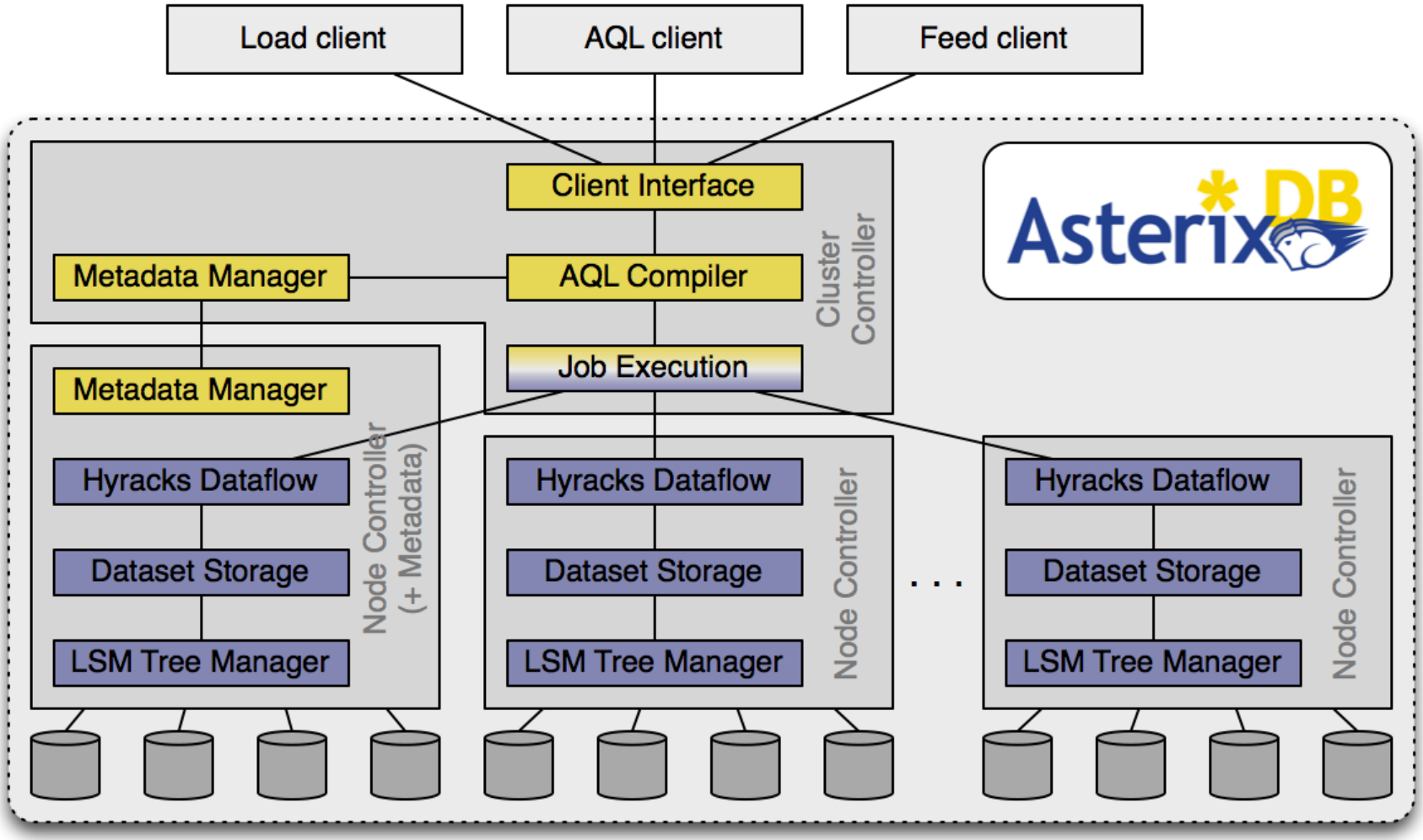
ASTERIX Queries (SQL++ or AQL)

- *Q1:* List the user names and messages sent by Gleambook social network users with less than 3 friends:

```
SELECT user.name AS uname,
                (SELECT VALUE msg.message
                 FROM GleambookMessages msg
                 WHERE msg.authorId = user.id) AS messages
FROM GleambookUsers user
WHERE COLL_COUNT(user.friendIds) < 3;
```

```
{ "uname": "NilaMilliron", "messages": [  ] }
{ "uname": "WoodrowNehling", "messages": [ " love acast its 3G is good:)" ] }
{ "uname": "IsbelDull", "messages": [ " like product-y the plan is amazing", " like
   product-z its platform is mind-blowing" ] }
 . . .
```

AsterixDB System Overview



Hyracks Dataflow Runtime

- Partitioned-parallel platform for data-intensive computing
- Job = dataflow DAG of operators and connectors
  - Operators consume and produce ***partitions*** of data
  - Connectors ***route*** (repartition) data between operators
- Hyracks *vs.* the "competition"
  - Based on time-tested parallel database principles
  - *vs.* Hadoop MR: More flexible model and less "pessimistic"
  - *vs.* newer SQL-on-Hadoop runtimes: Emphasis on out-of-core execution and adherence to memory budgets
  - Fast job activation, data pipelining, binary format, state-of-the-art DB style operators (hash-based, indexed, ...)
- Early test at Yahoo! Labs on 180 nodes (1440 cores, 720 disks)

Hyracks *(cont.)*

Query

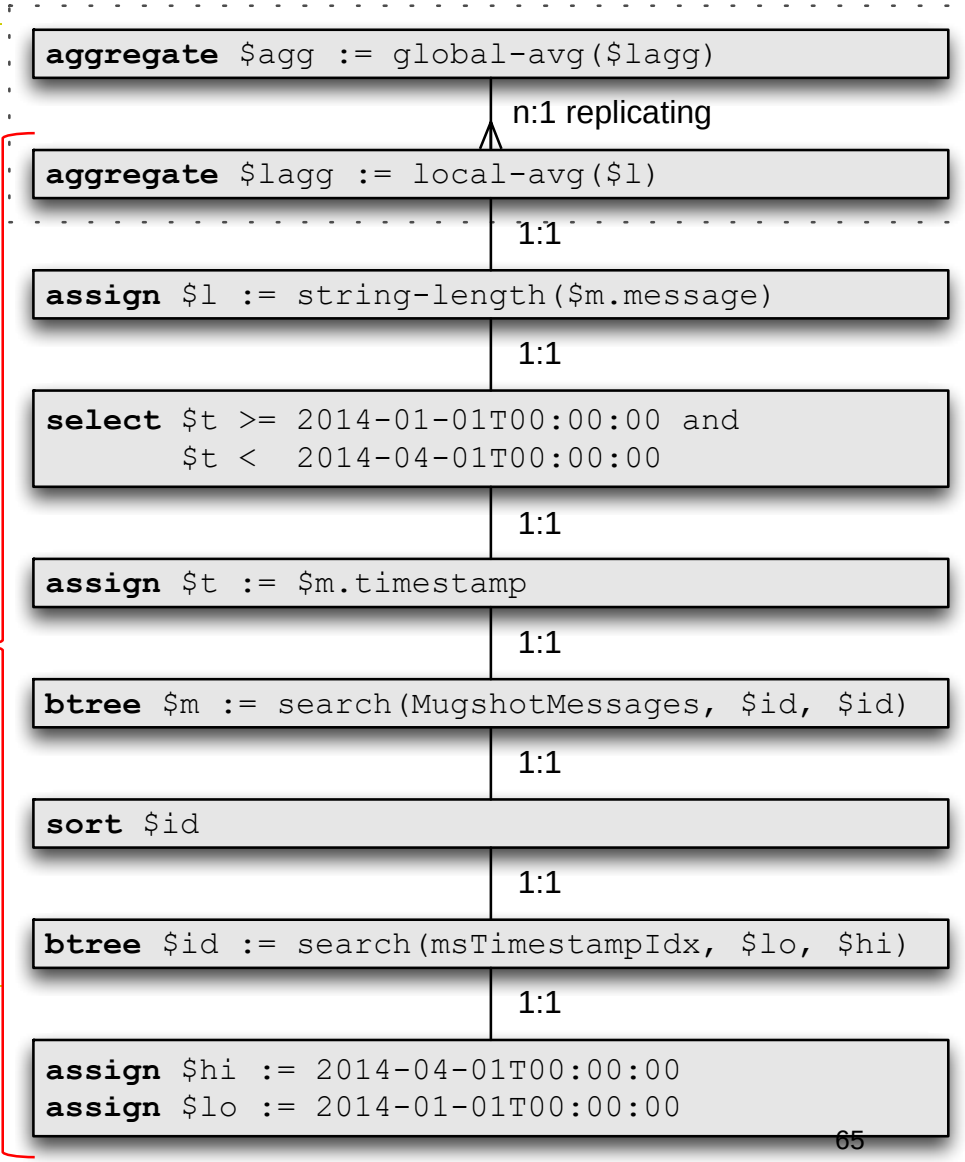
```
use dataverse TinySocial

avg (
 for $m in dataset MugshotMessages
 where $m.timestamp >=
     datetime("2014-01-01T00:00:00")
  and $m.timestamp <
     datetime("2014-04-01T00:00:00")
 return string-length($m.message)
)}
```

Select Options | Clear Query | Run

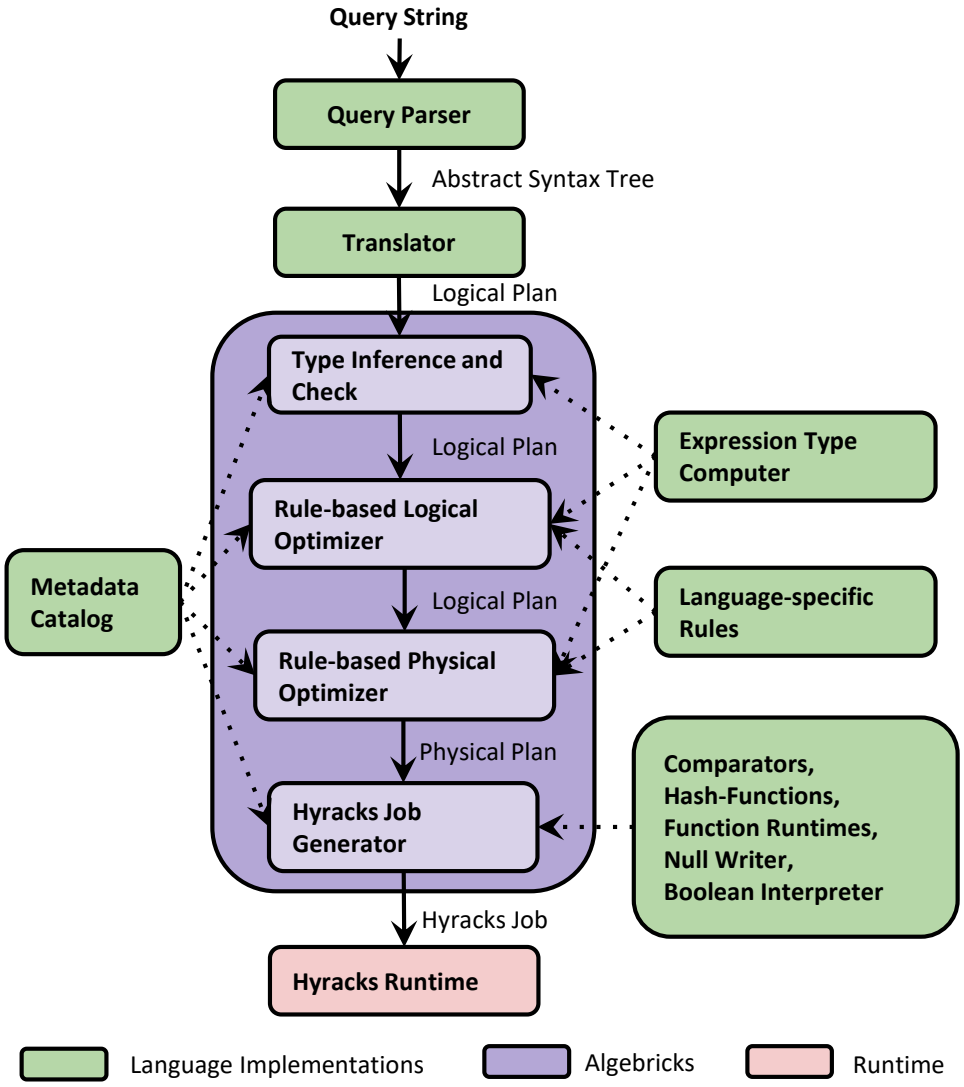
- [ ] Print parsed expressions
- [ ] Print rewritten expressions
- [ ] Print logical plan
- [ ] Print optimized logical plan
- [x] Print Hyracks job
- [x] Execute query

Algebricks

Partitioned Parallelism!



Algebricks Query Compiler Framework



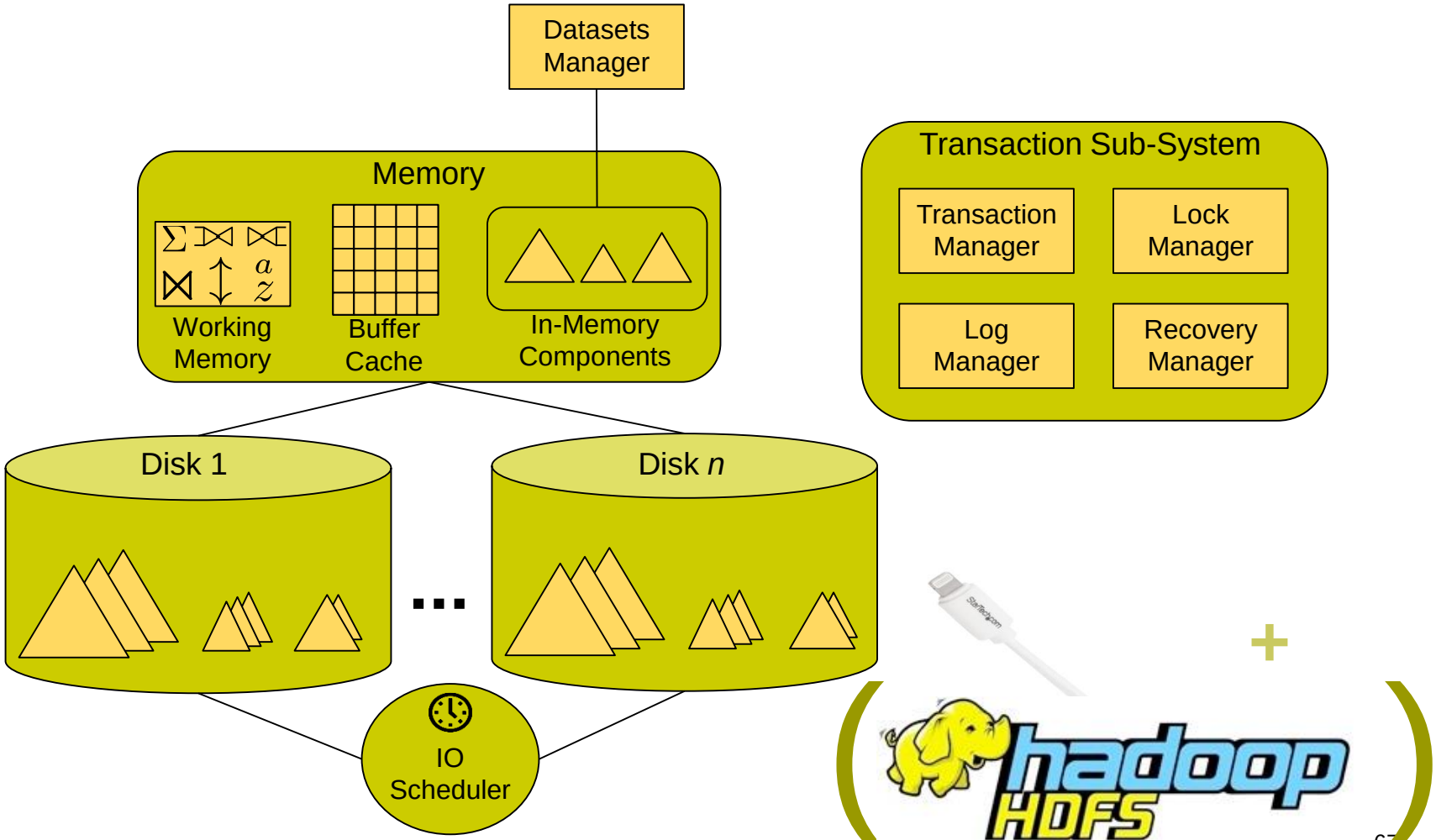
Algebricks

- Logical Operators
- Logical Expressions
- Metadata Interface
- Model-Neutral Logical Rewrite Rules
- Physical Operators
- Model-Neutral Physical Rewrite Rules
- Hyracks Job Generator

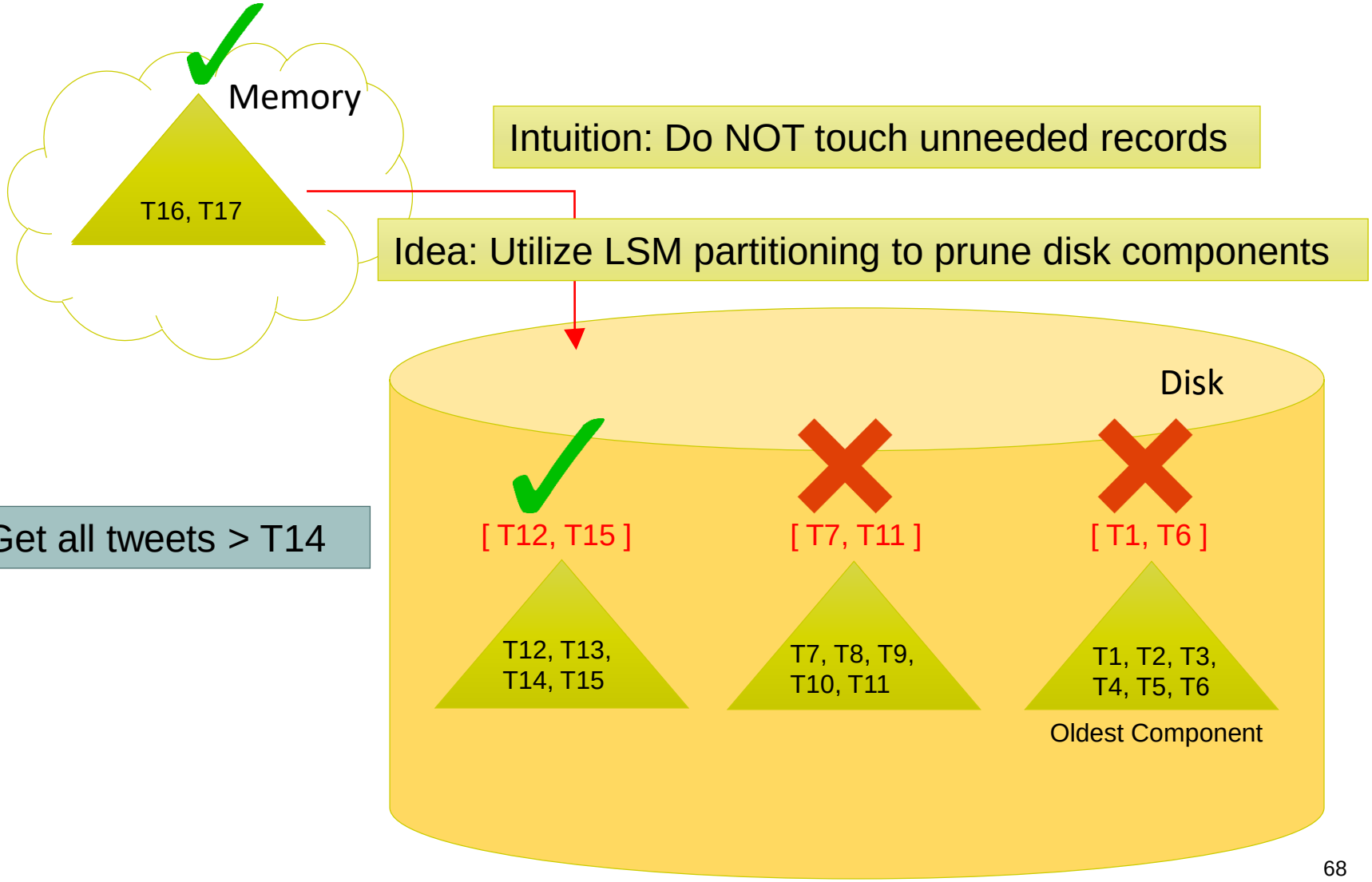
Target Query Language

- Query Parser (AST)
- AST Translator
- Metadata Catalog
- Expression Type Computer
- Logical Rewrite Rules
- Physical Rewrite Rules
- Language Specifics

Native Storage Management



LSM-Based Filters



For More Information

- Apache AsterixDB home
  - http://asterixdb.apache.org/
- SQL++ Primer
  - http://asterixdb.apache.org/docs/0.9.4.1/sqlpp/primer-sqlpp.html

Conclusion

Big-data Libraries
MLlib (Machine Learning), GraphX

High-level Languages
SparkSQL, Pig, SQL++, HiveQL

Distributed Computing
MapReduce (Hadoop and Google), Resilient Distributed Dataset (Spark), Hyracks (AsterixDB)

Big Data Distributed Storage
Hadoop Distributed File System, Cloud storage systems (Amazon S3 and Google File System), Key-value stores

Cloud Services
Amazon Web Services, Microsoft Azure, and Google Cloud Platform

Coordination/Cluster Management
Oozie, Yarn, Kubernetes

Thank You!

Questions?