CSCI E-88 Principles Of Big Data Processing

Harvard University Extension, Fall 2019

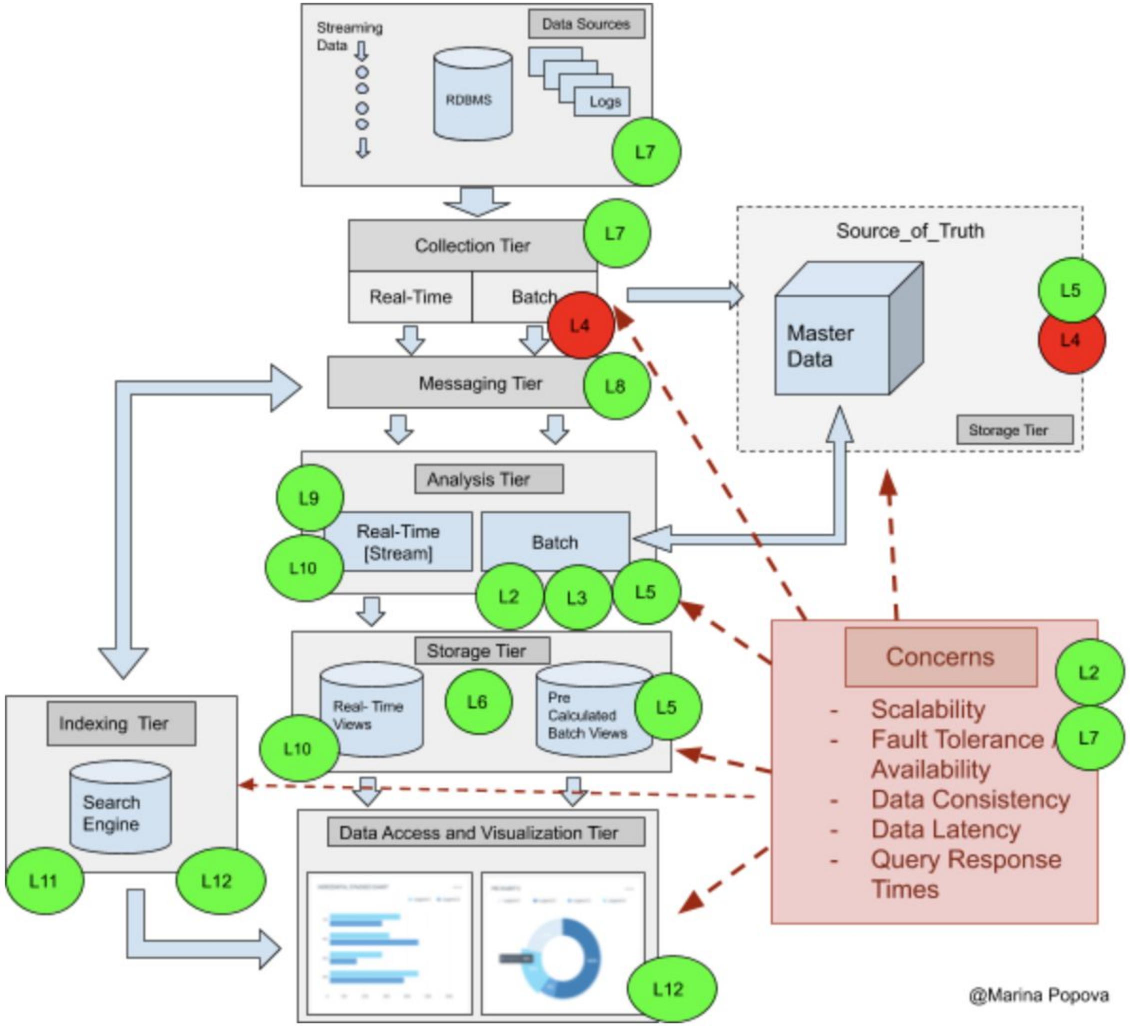
Marina Popova

Lecture 4 - Hadoop MR, Master Datasets: modeling and storage formats

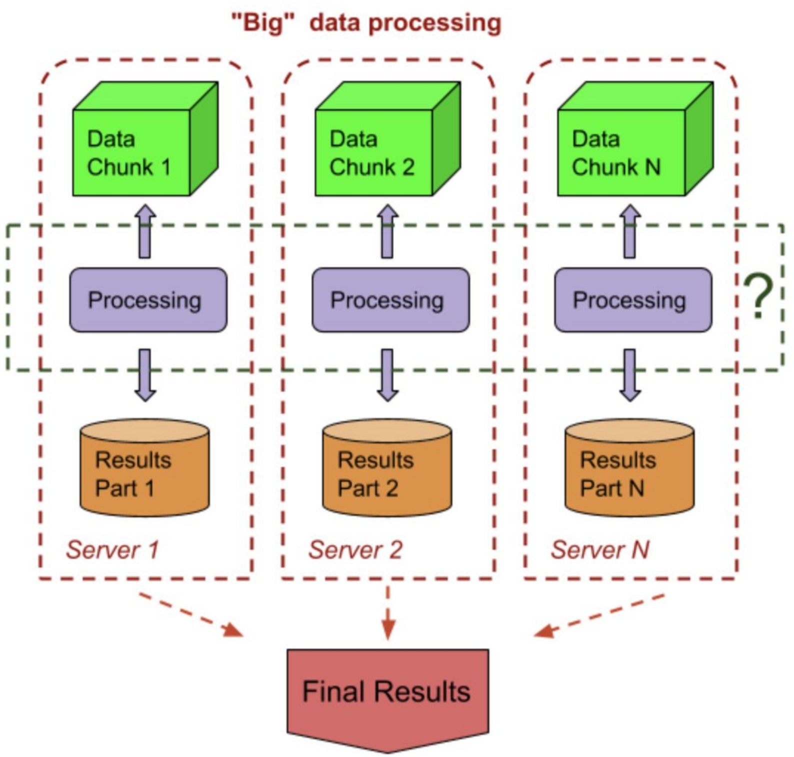
Agenda

- MapReduce in details
- Hadoop MapReduce
- Data Storage: Master Datasets:
    - modeling
    - storage formats

Where Are We?


Data Sources
Streaming Data
RDBMS
Logs
L7
Collection Tier
L7
Real-Time
Batch
L4
Source_of_Truth
Master Data
L5
L4
Storage Tier
Messaging Tier
L8
Analysis Tier
L9
Real-Time [Stream]
Batch
L10
L2
L3
L5
Storage Tier
Real- Time Views
L6
Pre Calculated Batch Views
L5
L10
Indexing Tier
Search Engine
L11
L12
Data Access and Visualization Tier
L12
Concerns
- Scalability
- Fault Tolerance / Availability
- Data Consistency
- Data Latency
- Query Response Times
L2
L7
@Marina Popova

Scaling of Data Processing - focus on processing



Requirements for Data Processing are the same:

- **Scalability**
- **Fault Tolerance**
- **Data Consistency**
- **Data Latency**
- **Query Performance**

how do we implement this? ...

@Marina Popova

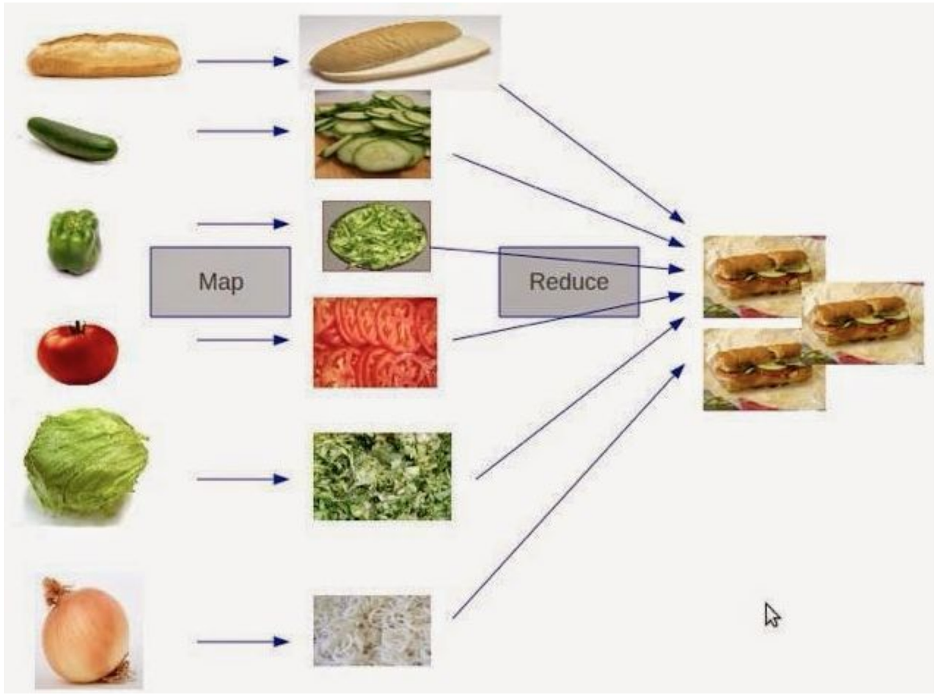
... and the solution is: MR

Map-Reduce paradigm:
A way to process large volumes of data in parallel by dividing the work into a set of independent tasks.

In essence, it is a set of **Map** and **Reduce** tasks that are combined to get final results:

- Map function transforms the piece of data into **key-value pairs** and then the keys are sorted
- Reduce function is applied to **merge the values** based on the key into a single output

Picture source:
https://www.datasciencecentral.com/forum/topics/what-is-map-reduce



MapReduce Framework

The main Open-Source framework that implements MR (and more) is Apache Hadoop.
This is why most tutorial and discussions are expressed in terms of Hadoop MapReduce.
However, there are many other implementations/distributions of Hadoop:

- Cloudera
- MapR
- Hortonworks
- IBM (BigInsights)
- Others

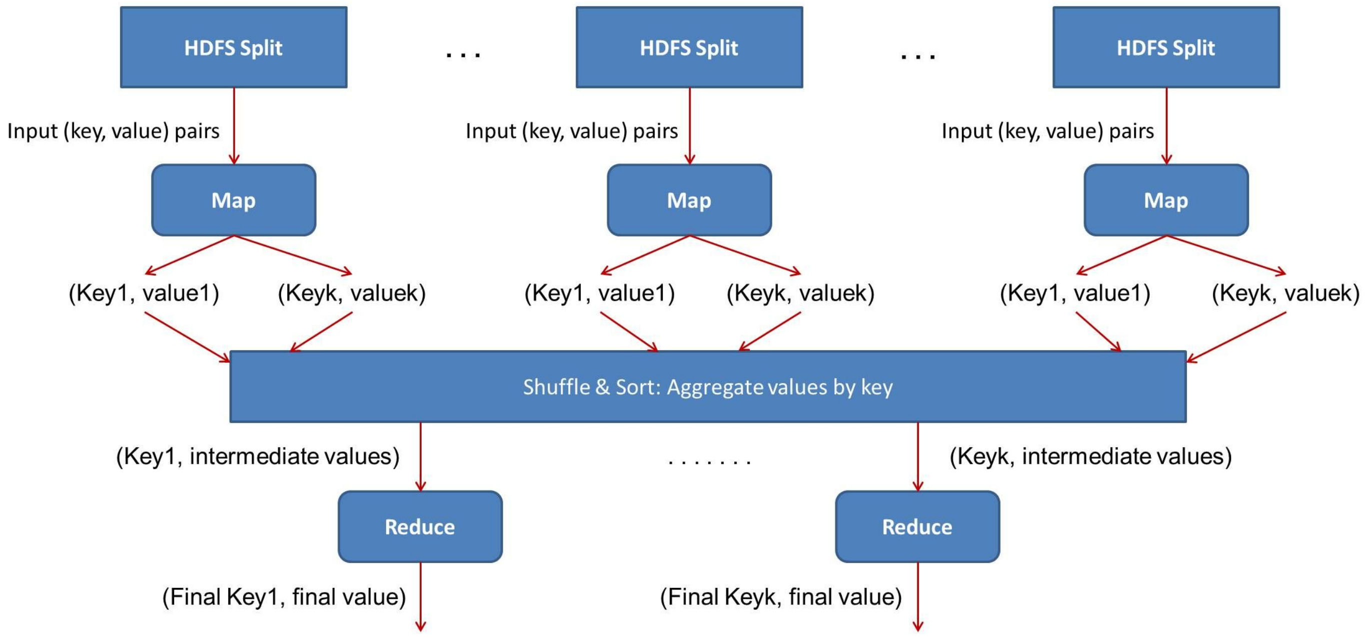
Those implementations hide the messy **details of parallelization, fault-tolerance, data distribution and load balancing in a library.** It uses re-execution as the primary mechanism for fault tolerance.

We will discuss Hadoop MR later on, and for now focus on the generic concepts

MapReduce Execution Flow

References: http://data-flair.training/blogs/hadoop-mapreduce-introduction-tutorial-comprehensive-guide/

1. Input splitting
2. Task allocation
3. Map phase
4. Combiner phase
5. Partition phase
6. Shuffle / Sort phase
7. Reduce phase



@Marina Popova

MapReduce Execution Flow

1. Input splitting

- Input data should be BIG to benefit from MR processing
- in order to parallelize data processing, input data (file or files) are split into large chunks, or , how they are called, "splits"
- split size is a user defined value and you can choose your own split size based on your volume of data

2. Task allocation

- The number of resulting splits will determine the number of Mappers needed to process your data
- MR will create that number of instances of your Map code, and will assign splits for each Map job
- One split can be processed by one Map job only

MapReduce Execution Flow

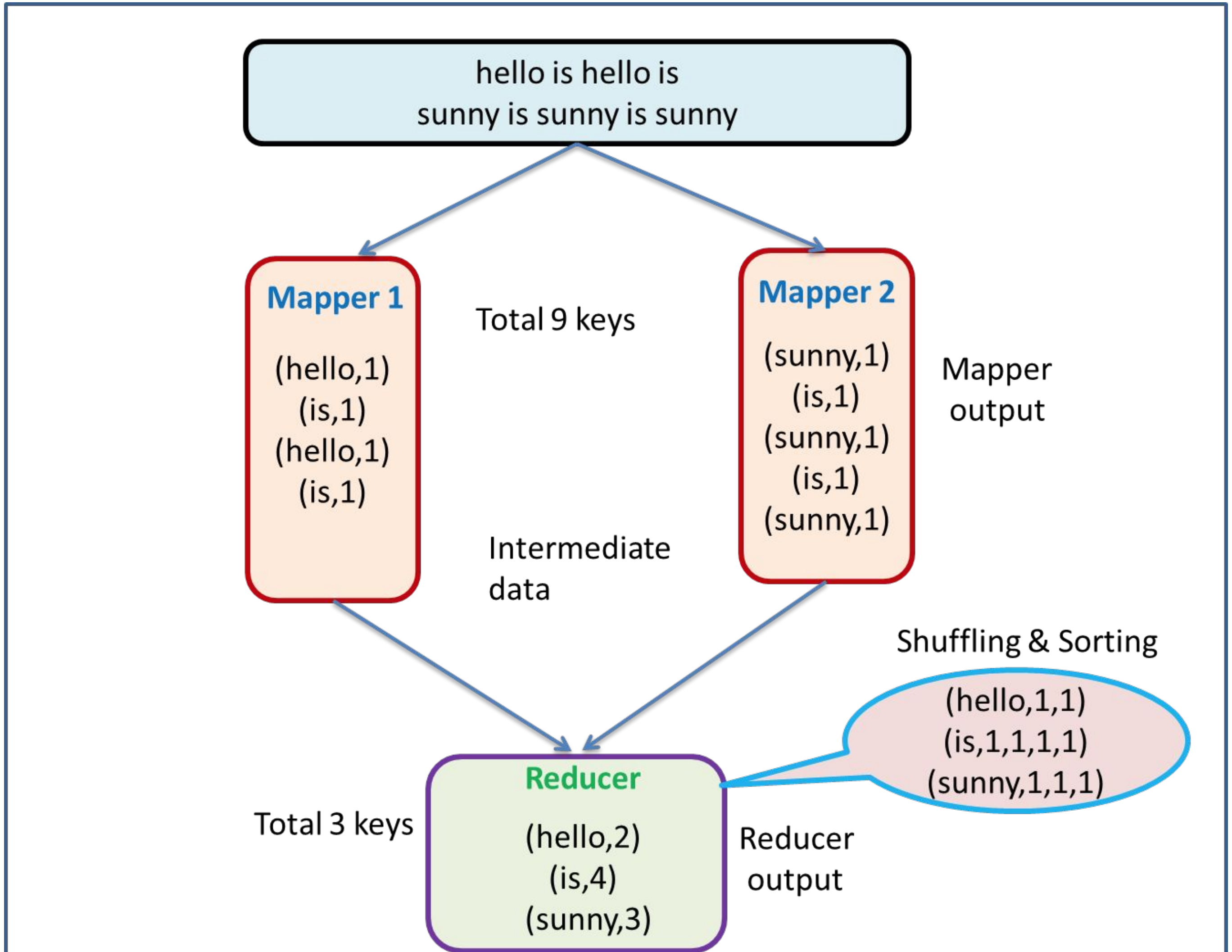
3. Map Phase

- Each Map job will reads its assigned spit of data
- It parses it as a list of <key, value> pairs
- For each <key, value> pair it invokes its map function (your code)
- The output of the map function is another <key, value> pair
- The output of map function is called "intermediate results" and is buffered in the Mapper memory

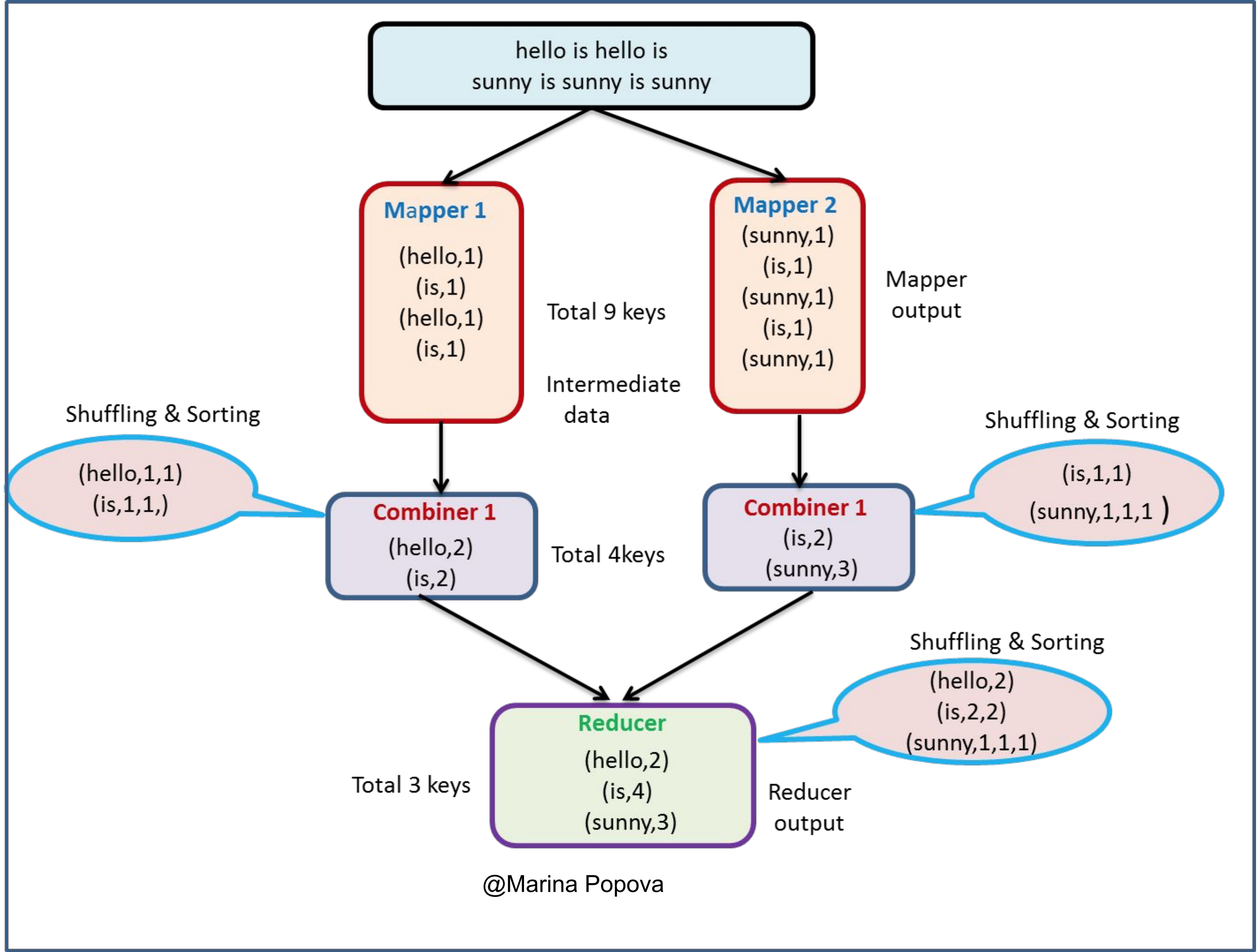
4. Combiner Phase

- Combiner is optional
- It can pre-aggregate results of the Map function
- This can drastically reduce the amount of data sent over the network to reducers
- **Sorting also can be done** in this phase - on the in-memory buffer data only

No combiner:


hello is hello is
sunny is sunny is sunny
Mapper 1
(hello,1)
(is,1)
(hello,1)
(is,1)
Total 9 keys
Mapper 2
(sunny,1)
(is,1)
(sunny,1)
(is,1)
(sunny,1)
Mapper output
Intermediate data
Shuffling & Sorting
(hello,1,1)
(is,1,1,1,1)
(sunny,1,1,1)
Reducer
(hello,2)
(is,4)
(sunny,3)
Total 3 keys
Reducer output

with combiner:



MapReduce Execution Flow

5. Partitioning Phase

- If there are more than one Reducer - Partitioning has to happen
- Partitioner is invoked when:
  - Mapper's memory buffer is full or
  - when Mapper is done and
  - after Combiner potentially pre-aggregated the intermediate results
  - before the <key, value> pairs from memory are written onto the local disk
- Partitioner calculates hash values of each <key> - either using default or your own logic - based on the number of partitions used
- The total **number of partitions is the same as the number of reduce tasks** for the job. Thus, this controls which reducer each key is sent to
- It then groups and writes all <key, value> into separate locations on disk - separate per each partition

Ref: http://bigdata.devcodenote.com/2012/11/map-reduce-shuffle-and-sort.html

MapReduce Execution Flow

6. Shuffle/ Sort Phase

- Up until now all data processing was local to the Map jobs - all writes were to local disks only
- Shuffle and Sort phase is where the actual data sending over the network is done (via HTTP, usually)
- The work here is done on all results from the Map job - which can be multiple local files generated from each in-memory buffer
- All these local files are merged, and another round of sorting is performed, per each partition
- Then each partition data is sent over the network to the corresponding Reducer

MapReduce Execution Flow

7. Reduce Phase

- Data from all Mappers will come to one or more Reducer, based on the partitioning
- There can be many keys (and their associated values) in each partition, but the **records for any given key are all in a single partition**
- Since there maybe multiple [locally] sorted files for the same keys arriving from multiple Mappers, an additional sort and merge has to happen before the actual Reduce function is invoked
- If all data fits into Reducer's memory - this sort and merge can happen in memory; if not - it has to happen on-disk (local to Reducer)
- Once done - the final list of <key, value> pairs is passed to the Reduce function (your code)
- The results are written out

When determining an optimal number of Reducers, one rule of thumb is to aim for reducers that each run for five minutes or so, and which produce a large block of output . Too many reducers may lead to lots of small files which is very inefficient

MapReduce detailed execution flow:

How does MR addresses key scaling requirements?

Scalability:

- operation parallelization

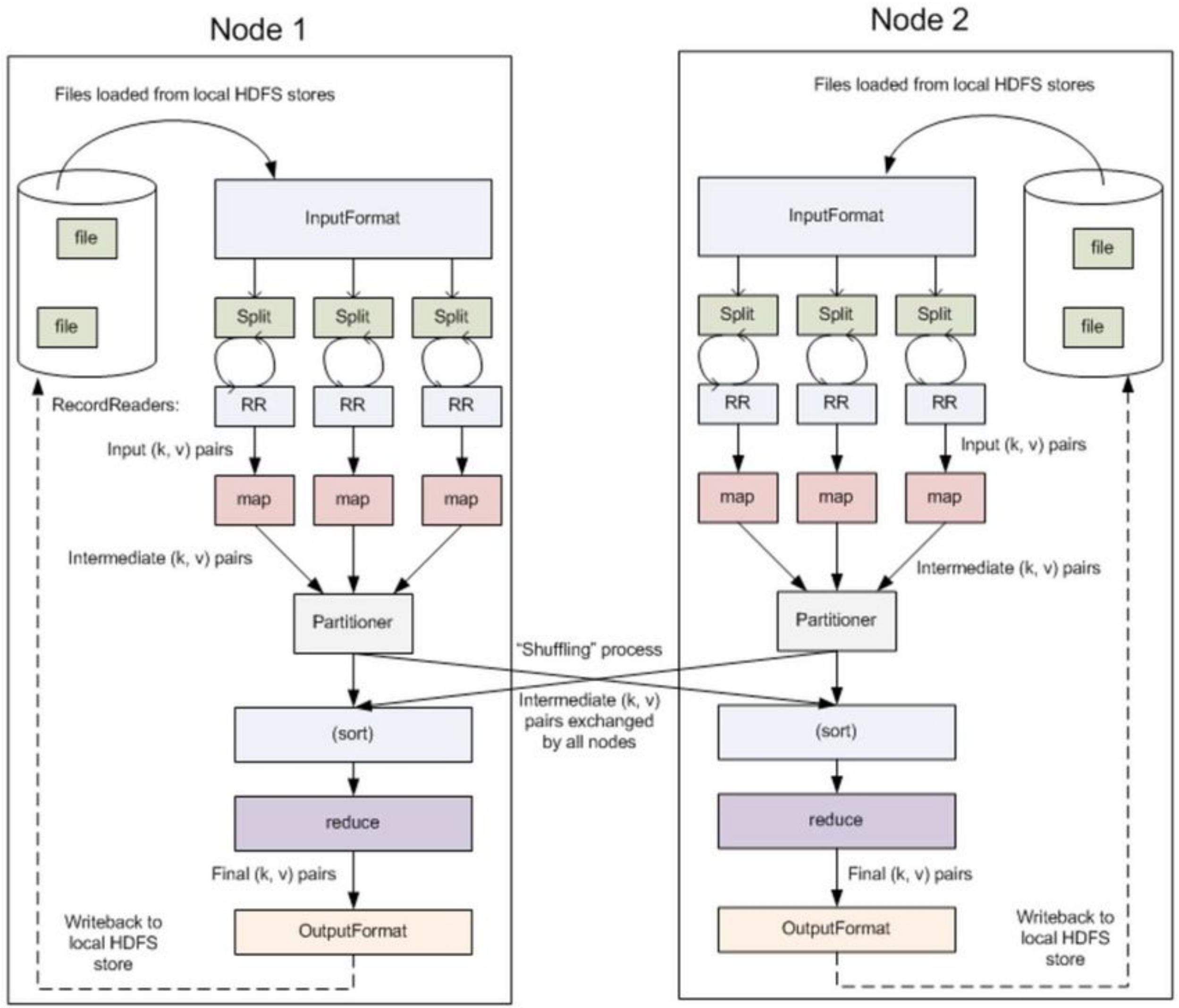
Fault Tolerance

- idempotent operations (detecting and re-running failed Map and/or Reduce jobs)

Data Consistency:

- data replication (HDFS)

Ref: https://www.researchgate.net/figure/Detailed-Hadoop-MapReduce-Data-Flow-14_fig1_224255467


Node 1
Node 2
Files loaded from local HDFS stores
Files loaded from local HDFS stores
file
file
InputFormat
Split
RecordReaders:
RR
Input (k, v) pairs
map
Intermediate (k, v) pairs
Partitioner
"Shuffling" process
Intermediate (k, v) pairs exchanged by all nodes
(sort)
reduce
Final (k, v) pairs
OutputFormat
Writeback to local HDFS store

MapReduce - examples

- MapReduce programming model is used in many programming languages and frameworks
- Java:
  - ForkJoinPool
  - Java 8 streams
- Spark
- AWS Athena
- many others ....

We will review the original implementation by the Hadoop MR framework.

Hadoop MR Architecture Overview

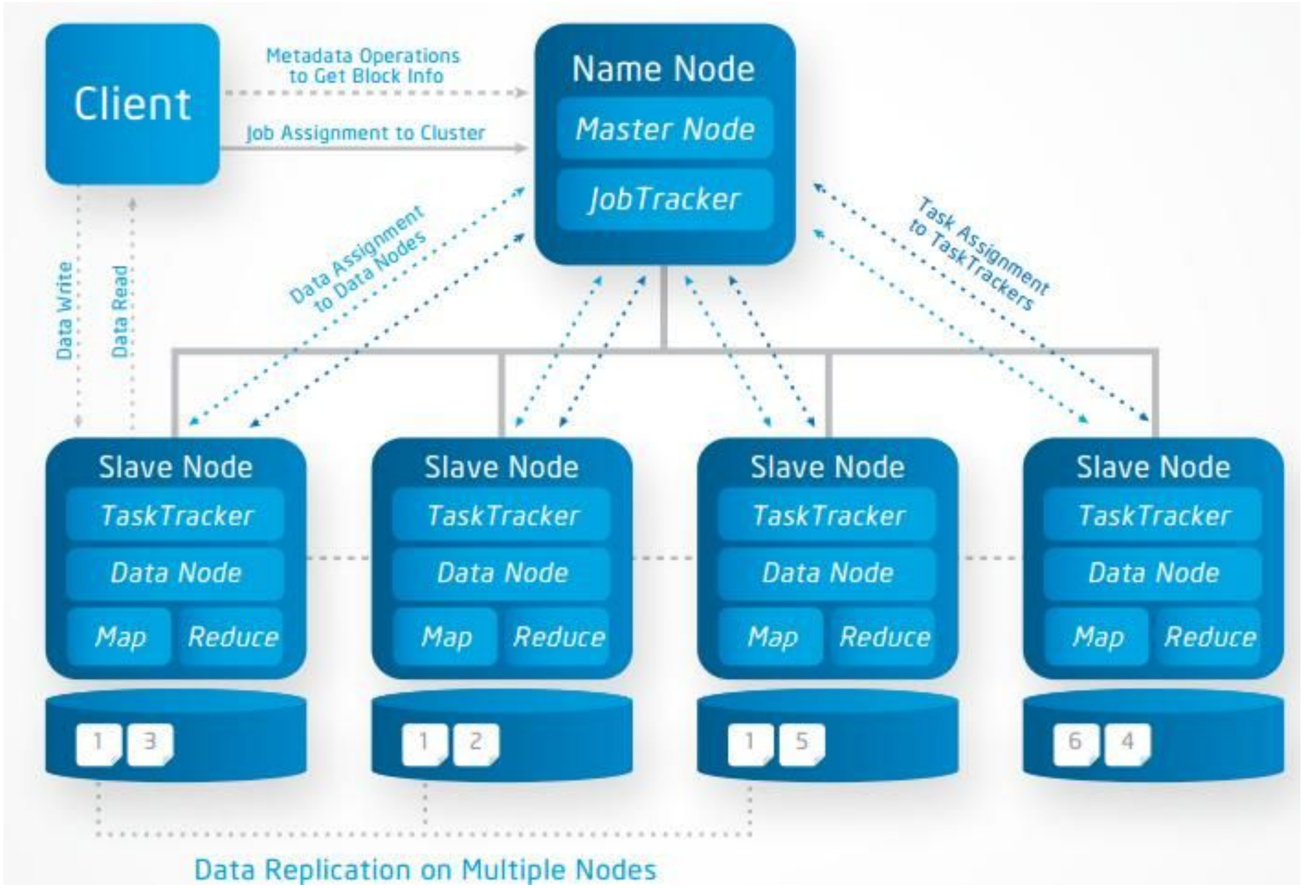
References:
http://hadoop.apache.org/docs/stable/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html
https://helloworldpoc.wordpress.com/2015/02/16/hdfs-architecture/

Typically the **compute nodes and the storage nodes are the same**, that is, the MapReduce framework and the HDFS are running on the same set of nodes. This configuration allows the framework to effectively schedule tasks on the nodes where data is already present, resulting in very high aggregate bandwidth across the cluster.

The MapReduce framework consists of:

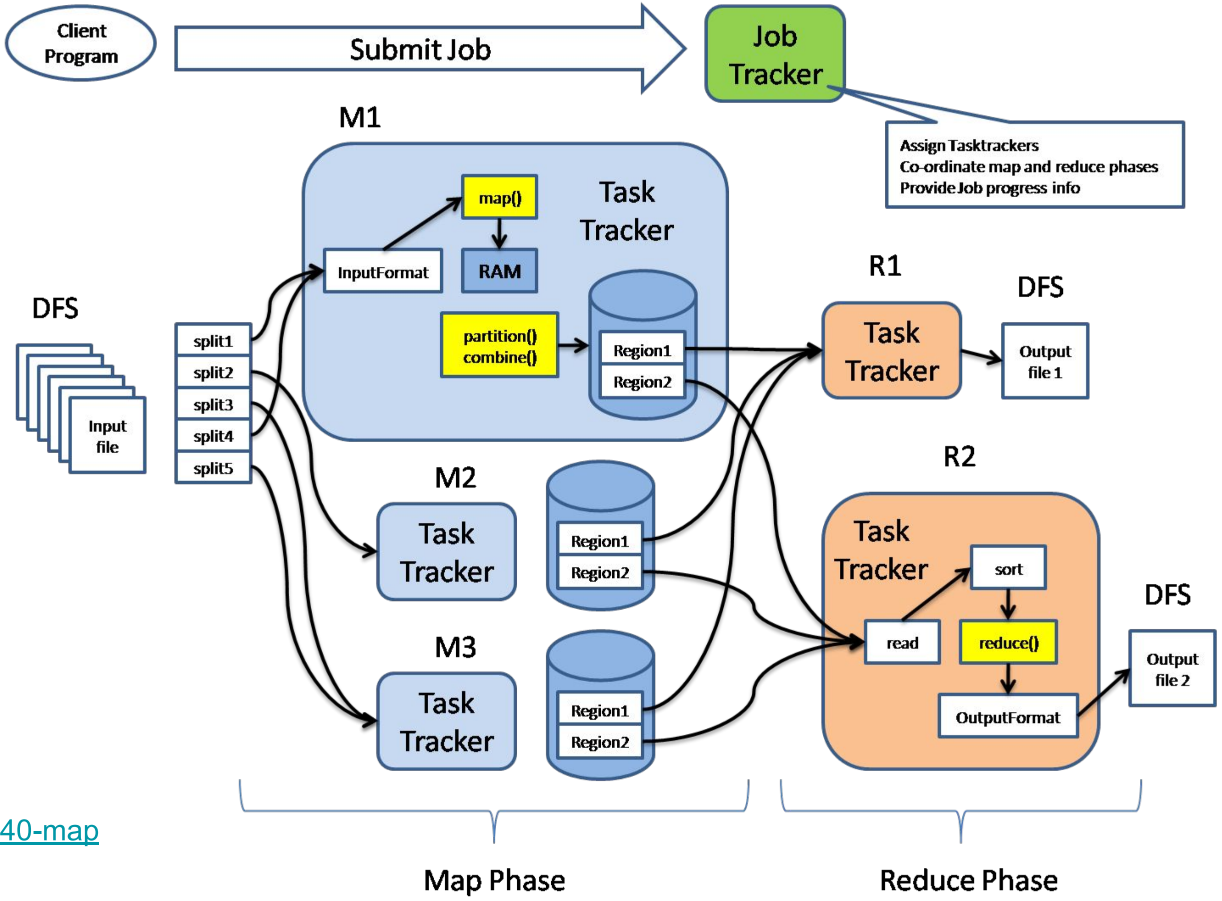
- a single master JobTracker (or ResourceManager in v2)
- one slave TaskTracker (NodeManager in v2) per cluster-node
- in v2 - MRAppMaster per application

v2 refers to the MR v2 on YARN


Client
Metadata Operations to Get Block Info
Job Assignment to Cluster
Name Node
Master Node
JobTracker
Data Write
Data Read
Data Assignment to Data Nodes
Task Assignment to TaskTrackers
Slave Node
TaskTracker
Data Node
Map
Reduce
1
3
1
2
1
5
6
4
Data Replication on Multiple Nodes

@Marina Popova

MapReduce control flow:

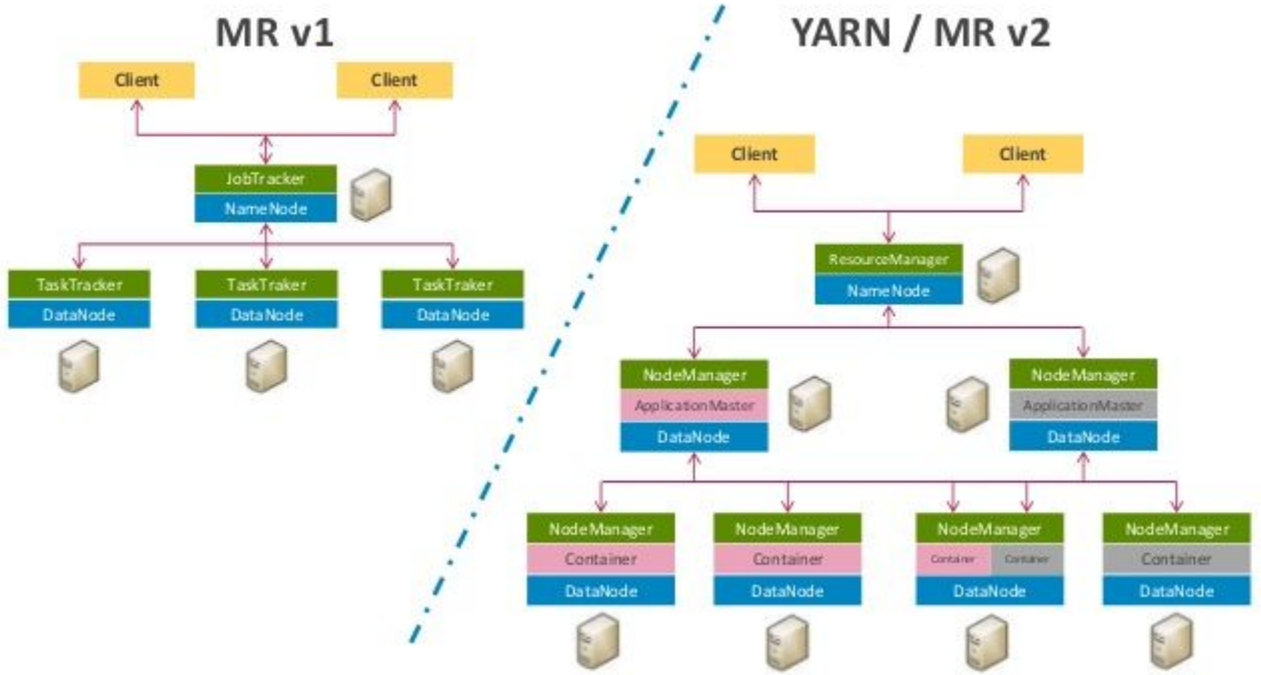


Ref:
http://a4academics.com/tutorials/83-hadoop/840-map-reduce-architecture

@Marina Popova

Ref: https://www.slideshare.net/PhilippeJulio/hadoop-architecture/25-MR_VS_YARN_ARCHITECTURE_MR

MR VS. YARN ARCHITECTURE



- **YARN** : **Y**et **A**nother **R**esource **N**egotiator
- **MR** : MapReduce

Hadoop MapReduce - how it works

Minimally, your **application does:**

- **specifies the input/output locations**
- **supplies *map* and *reduce* functions via implementations of appropriate interfaces**
- creates the ***job configuration*** with these and other job parameters

The Hadoop ***job client*** then submits the job (jar/executable etc.) and configuration to the JobTracker (or ResourceManager) which does:

- **distribution of the software/configuration to the slaves**
- scheduling and monitoring of tasks
- providing status and diagnostic information back to the job-client

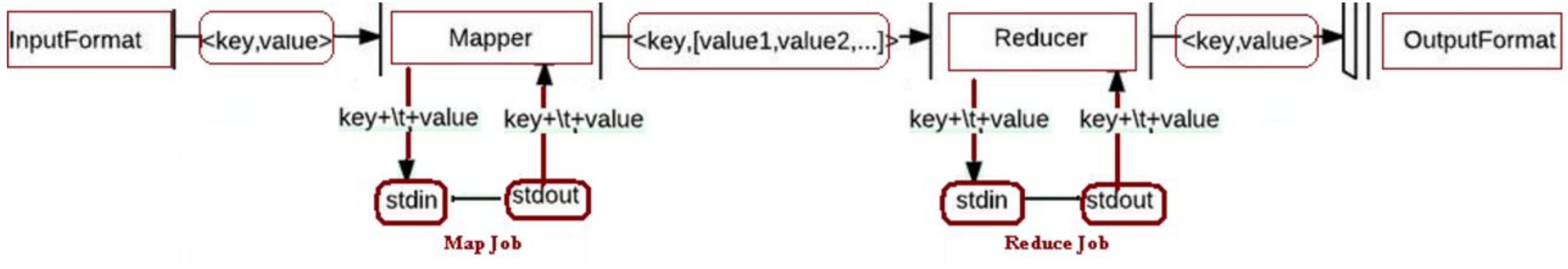
Hadoop Streaming

Although the Hadoop framework is implemented in Java™, MapReduce applications need not be written in Java - thanks to the Hadoop Streaming

- Hadoop Streaming is a utility which allows users to create and run jobs with any executables (e.g. shell utilities) as the mapper and/or the reducer
- Hadoop Streaming is a generic API which allows writing Mappers and Reduces in any language. But the basic concept remains the same. Mappers and Reducers receive their input and output on stdin and stdout as (key, value) pairs
- Apache Hadoop uses streams as per UNIX standard between your application and Hadoop system - basically , it uses 'stdin' and 'stdout' streams
- Ref: http://www.devx.com/opensource/introduction-to-hadoop-streaming.html
- NOT to be confused with Stream Processing - which is a totally different concept of processing real-time data

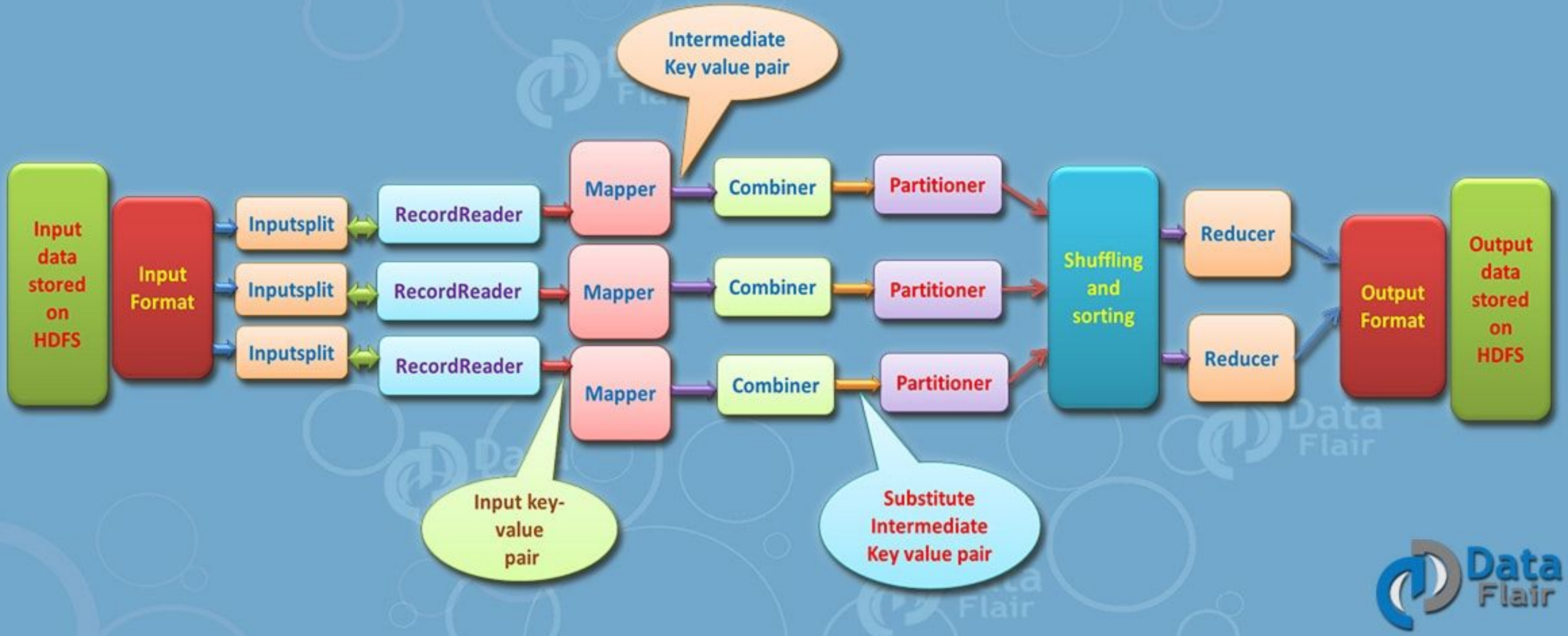
Hadoop MapReduce Framework

The following diagram shows the process flow in a streaming job



Next, we will review main Java MR APIs for building MR jobs

MapReduce Job Execution Flow



Hadoop MapReduce Java APIs

Main APIs

Mapper and Reducer interfaces: have to be implemented to provide the map and reduce methods

Mapper interface: maps input key/value pairs to a set of intermediate key/value pairs

public class Mapper<KEYIN, VALUEIN, KEYOUT, VALUEOUT>

The framework first calls setup(org.apache.hadoop.mapreduce.Mapper.Context), followed by *map(Object, Object, org.apache.hadoop.mapreduce.Mapper.Context)* for each key/value pair in the InputSplit.

Example: *public class TokenCounterMapper extends Mapper<Object, Text, Text, IntWritable>*

Hadoop MapReduce Java APIs

- **Maps** are the individual tasks that transform input records into intermediate records
- The transformed intermediate records do not need to be of the same type as the input records
- A given input pair may map to zero or many output pairs.
- The Hadoop MapReduce framework spawns one map task for each InputSplit generated by the InputFormat for the job.
- Mapper implementations are passed to the job via *Job.setMapperClass(Class)* method.

Hadoop MapReduce Java APIs

Reducer interface:

org.apache.hadoop.mapreduce.Reducer<KEYIN, VALUEIN, KEYOUT, VALUEOUT>

- **Reducer** reduces a set of intermediate values which share a key to a smaller set of values.
- The number of reduces for the job is set by the user via *Job.setNumReduceTasks(int)*.
- Reducer implementations are set for the job via the *Job.setReducerClass(Class)* method
- The framework then calls *reduce(WritableComparable, Iterable<Writable>, Context)* method for each <key, (list of values)> pair in the grouped inputs.

Hadoop MapReduce Java APIs

Job represents a MapReduce job configuration.
Job is the primary interface for a user to describe a MapReduce job to the Hadoop framework for execution.

Job is typically used to specify the Mapper, combiner (if any), Partitioner, Reducer, InputFormat, OutputFormat implementations.

Job Control:
Job.waitForCompletion(boolean) : Submit the job to the cluster and wait for it to finish

Hadoop MapReduce Java APIs

InputFormat describes the input-specification for a MapReduce job:
org.apache.hadoop.mapreduce.InputFormat<K,V>

The MapReduce framework relies on the InputFormat of the job to:

1. Validate the input-specification of the job.
2. Split-up the input file(s) into logical InputSplit instances, each of which is then assigned to an individual Mapper.
3. Provide the RecordReader implementation used to glean input records from the logical InputSplit for processing by the Mapper.

TextInputFormat is the default InputFormat.

Hadoop MapReduce Java APIs

OutputFormat describes the output-specification for a MapReduce job
org.apache.hadoop.mapreduce.OutputFormat<K,V>

The MapReduce framework relies on the OutputFormat of the job to:

1. Validate the output-specification of the job; for example, check that the output directory doesn't already exist.
2. Provide the RecordWriter implementation used to write the output files of the job. Output files are stored in a FileSystem

TextOutputFormat is the default OutputFormat

What are we doing? :)

At this point, lets make sure we can still see the forest through the trees and have not lost the sight of what's important :)

@Marina Popova

Processing System Types - Refresher

Type 1: Real-time [Limited range] + Ad-hoc queries systems:
Query = function (windowed real-time data)

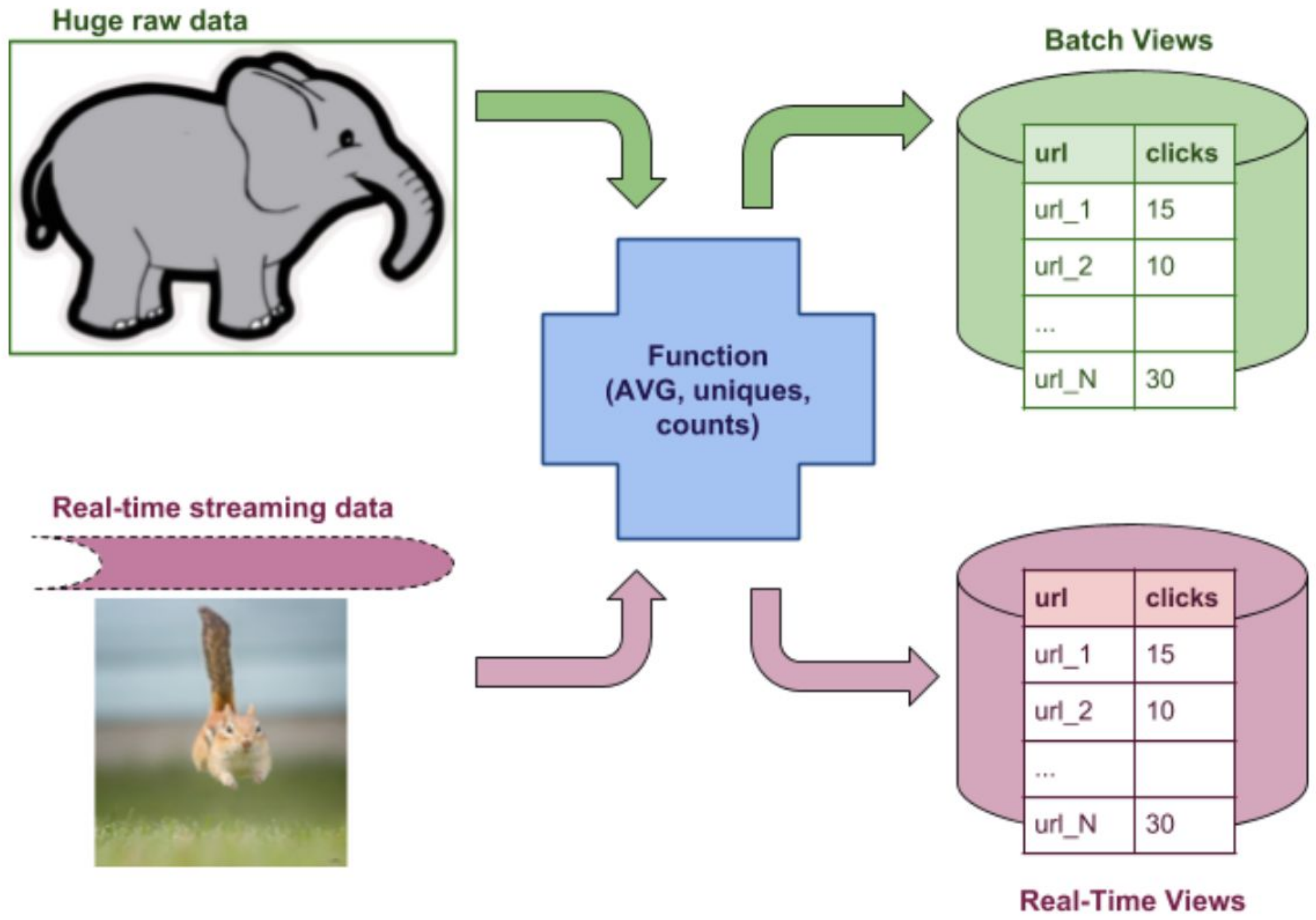
Type 2:
Full Historical + Limited Real-Time + Pre-defined queries systems:
Query = function(aggregated windowed data)
== function [function(windowed raw data)]

Type 3: Full historical Ad-hoc query systems:

Query = function (all raw data)

Type 4: Full Historical and Real-time ad-hoc query systems:
Query = function (all raw + real-time data)

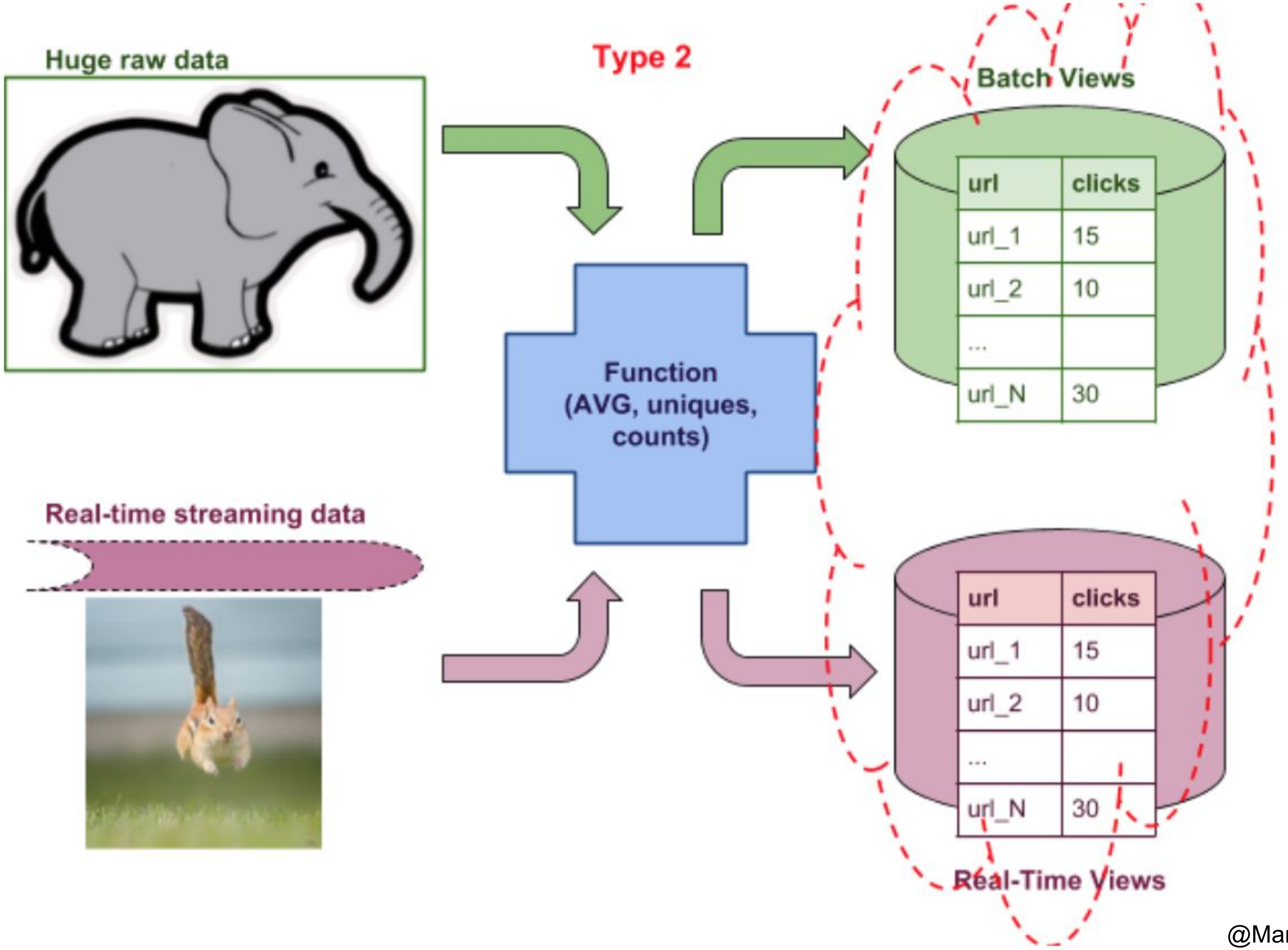
@Marina Popova



url	clicks
url_1	15
url_2	10
...	
url_N	30

url	clicks
url_1	15
url_2	10
...	
url_N	30

@Marina Popova

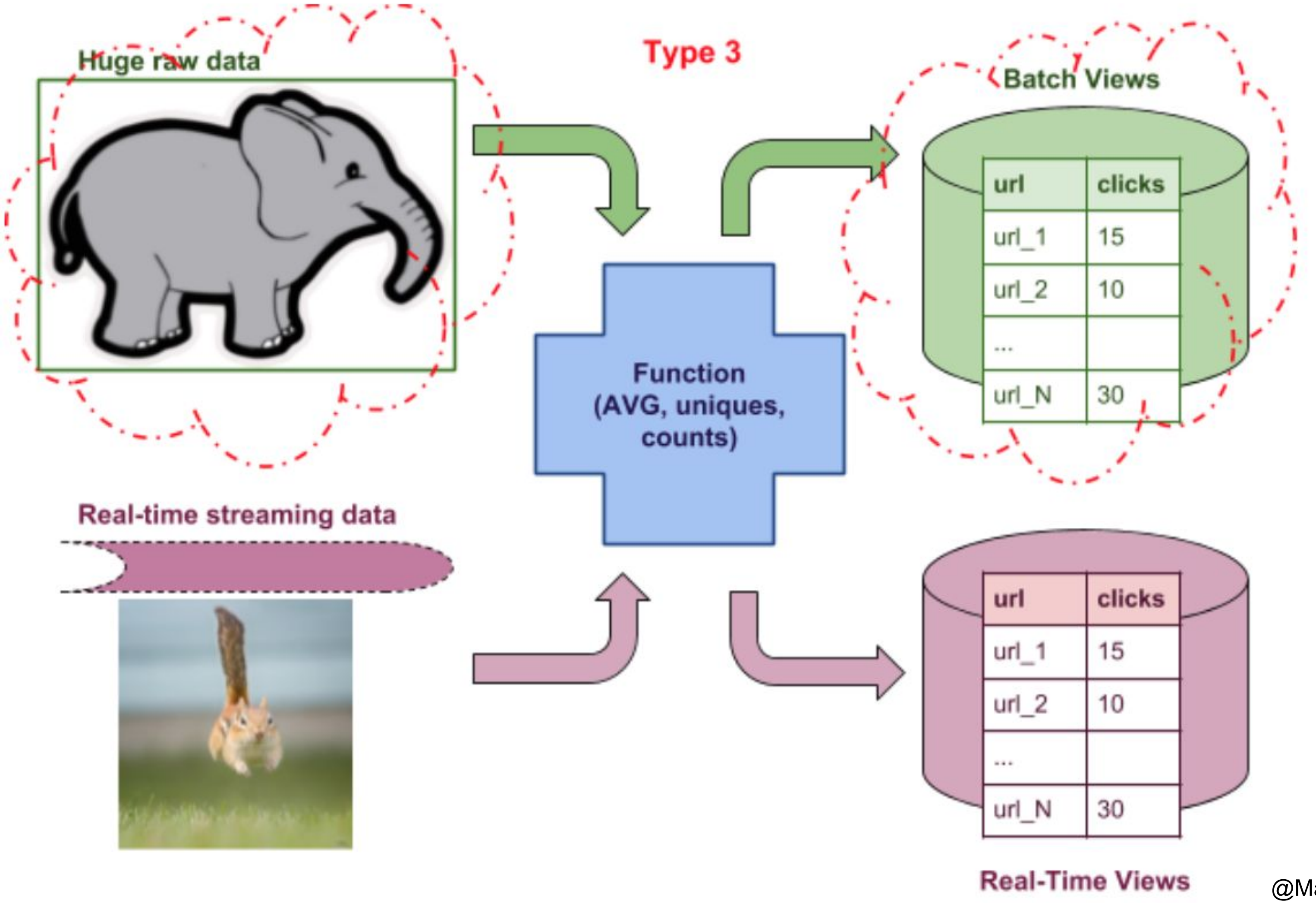


url	clicks
url_1	15
url_2	10
...	
url_N	30

url	clicks
url_1	15
url_2	10
...	
url_N	30

@Marina Popova

Type 3

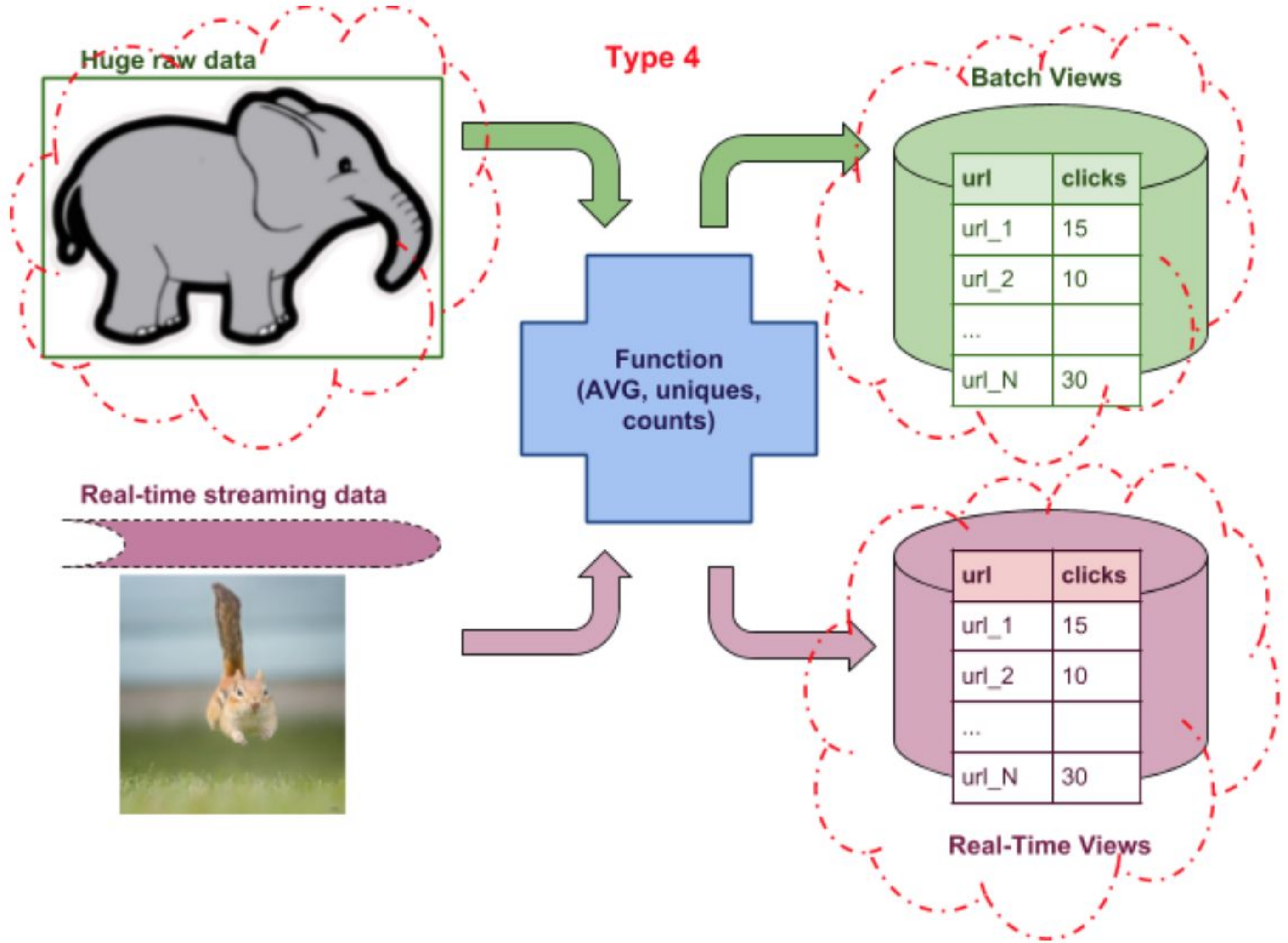


url	clicks
url_1	15
url_2	10
...	
url_N	30

url	clicks
url_1	15
url_2	10
...	
url_N	30

@Marina Popova

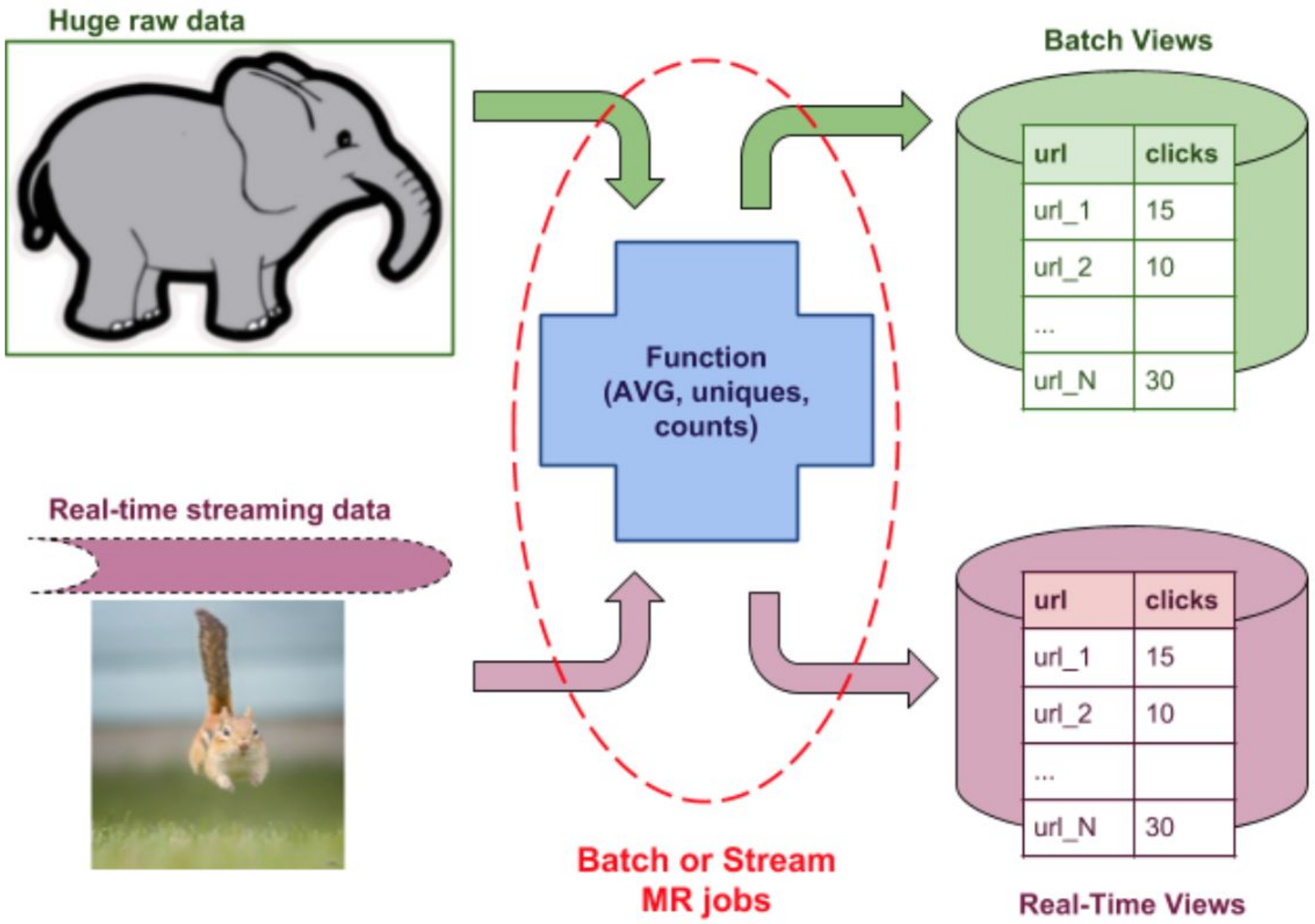
Type 4



url	clicks
url_1	15
url_2	10
...	
url_N	30

url	clicks
url_1	15
url_2	10
...	
url_N	30

@Marina Popova

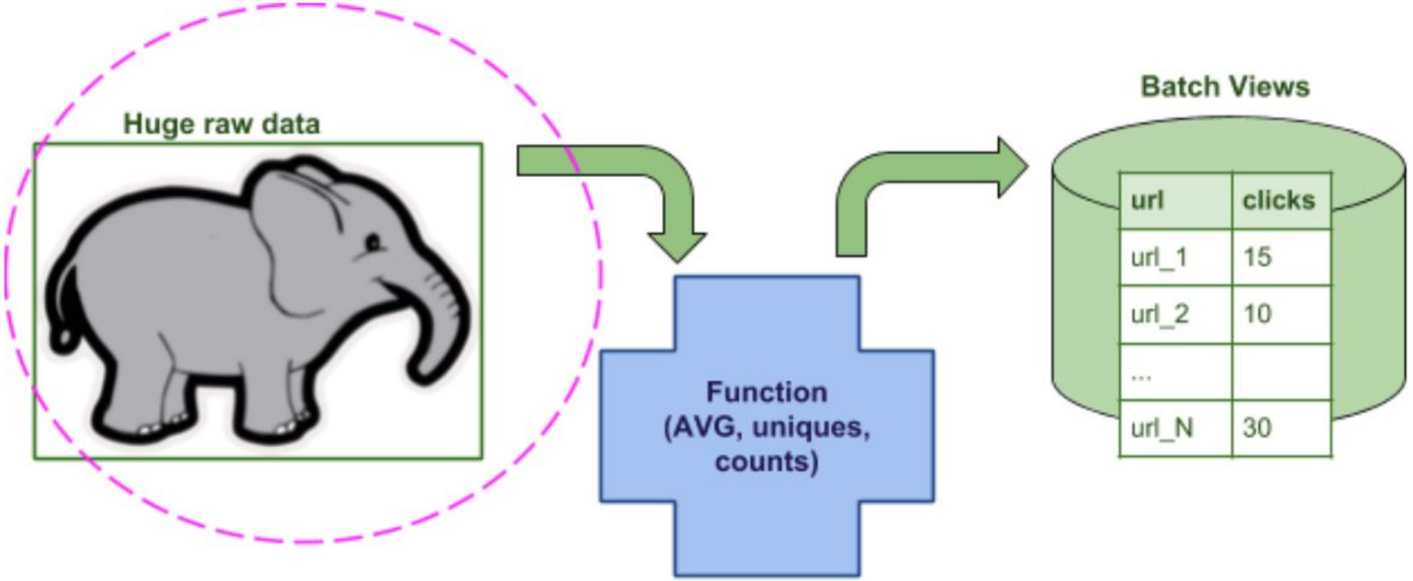


url	clicks
url_1	15
url_2	10
...	
url_N	30

url	clicks
url_1	15
url_2	10
...	
url_N	30

@Marina Popova

What are we doing?



url	clicks
url_1	15
url_2	10
...	
url_N	30

This is our focus next!

@Marina Popova

Master Datasets

Let's get back to the definition of the Big Data Systems we discussed in Lecture 1:

A **Data System** is a [set of] application[s] that provides answers to general questions about information that **has being collected so far** or is being collected in real time.
A **"Big" Data System** is a data system that , on top of that, can work on Terabyte- and Petabyte- scale of data.

Master Datasets - are the datasets that contain historical raw input data,
as opposed to other types of datasets with pre-calculated metrics, aggregates, indexes , etc.

Lets review and formalize the most important data model and storage properties for such datasets

Master Datasets

When designing and planning for Master Data Storage, the following important aspects have to be addressed:

- What to store (content and model) ?
- Where to store to (the actual data storage systems) ?
- How to store - formats and serialization ?
- How to store - algorithms and operations ?

We will be discussing the first two or three aspects in this Lecture, and the last one a bit later

Master Datasets - Requirements and Modeling

First, we will formalize the requirements for such datasets - to be able to answer the first question: **what to store?**

data should be row, **not derived**

In order to design a data system that can provide answers to as many questions as possible, we need to store raw data. By "raw data" we mean the most raw form of data such that other data points can be derived from.

Examples:

Master Datasets - Requirements and Modeling

Raw data: Apache logs: (source: http://pathalizer.sourceforge.net/wiki-access.log)

```
222.64.146.118 - - [19/Jun/2005:06:44:17 +0200] "GET /wximages/wxwidgets02-small.png HTTP/1.1" 200 12468
"http://blog.vckbase.com/bastet/" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; TencentTraveler )"
218.84.191.50 - - [19/Jun/2005:06:46:05 +0200] "GET /wximages/wxwidgets02-small.png HTTP/1.1" 200 12468
"http://blog.vckbase.com/bastet/" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
138.243.201.10 - - [19/Jun/2005:06:48:40 +0200] "GET /wiki.pl?WxWidgets_Bounties HTTP/1.1" 200 8873
"http://www.wxwidgets.org/toolbar.htm" "Mozilla/5.0 (Windows; U; Windows NT 5.1; en-GB; rv:1.7.7) Gecko/20050414
Firefox/1.0.3"
68.251.52.253 - - [19/Jun/2005:06:50:49 +0200] "GET /wiki.pl?WxWidgets_Compared_To_Other_Toolkits HTTP/1.1" 200
19476 "http://www.google.com/search?q=wxWidget+designer" "Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.7.8)
Gecko/20050511 Firefox/1.0.4"
```

Master Datasets - Requirements and Modeling

What is Derived data?
Answers to the queries or parsed information:

Ex1: number of unique URLs clicked per hour:

hour	count_of_uniq_urls
19/Jun/2005:06:00	3
19/Jun/2005:07:00	10

Ex2:
count of unique visitors
per URL per hour:

hour	url	count_of_visitors
19/Jun/2005:06:00	/wximages/wxwidgets02-small.png	2
19/Jun/2005:06:00	/wiki.pl?WxWidgets_Bounties	1
19/Jun/2005:06:00	/wiki.pl?WxWidgets_Compared_To_Other_Toolkits	1

You might ask: why not store this data as the master raw data??

Master Datasets - Requirements and Modeling

Problems with the derived data as the Master Data:

1. **You cannot ask uniqueness-based queries for a different bucket granularity**: (probably the most important)
   a. You cannot get correct answer to: "get a count of unique visitors for each URL over 12 hr time periods"
   b. Why? Because unique values can overlap in 1-hr buckets!
2. You cannot recover from potential human-faults (bugs)
   a. Lets assume you had a bug in your code accessing shared counts, that caused some counts to be doubled and some counts to be lost ….
   b. You fix your code - but you cannot re-calculate the counts anymore, since your raw data is not preserved
3. Semantics of data could change over time:
   a. Say you've added cookie-based user identification and determined that requests from 222.64.146.118 and 68.251.52.253 are from the same person
   b. Your counts are now incorrect

Conclusion: you have to store raw data as your Master Dataset, not derived data

Master Datasets - Requirements and Modeling

Raw data storage V1:

timestamp	url	clientID
19/Jun/2005:06:44:17 +0200	/wximages/wxwidgets02-small.png	222.64.146.118
19/Jun/2005:06:46:05 +0200	/wximages/wxwidgets02-small.png	218.84.191.50
19/Jun/2005:06:48:40 +0200	/wiki.pl?WxWidgets_Bounties	222.64.146.118
19/Jun/2005:06:50:49 +0200	/wiki.pl?WxWidgets_Compared_To_Other_Toolkits	68.251.52.253

Master Datasets - Requirements and Modeling

Further problems with the V1 of our Raw data model:

1. **We cannot answer questions for un-parsed parts of the logs**:
   a. You were asked to answer the following question: " get all unique URLs where a number of requests with the response codes other than 200 is more than 10 per hour"
   b. This is where response code is in the logs:

```
222.64.146.118 - - [19/Jun/2005:06:44:17 +0200] "GET /wximages/wxwidgets02-small.png HTTP/1.1" 200 12468
"http://blog.vckbase.com/bastet/" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; TencentTraveler )"
218.84.191.50 - - [19/Jun/2005:06:46:05 +0200] "GET /wximages/wxwidgets02-small.png HTTP/1.1" 200 12468
"http://blog.vckbase.com/bastet/" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
```

   c. You can add 'response_code' field to your master data set - but this will take effect only for new data , not for the data already processed; for the historical data - you cannot retrieve this information anymore

Master Datasets - Requirements and Modeling

Problems with the V1 of our Raw data model:

2. **We cannot recover from parsing/bugs in the code**
    a. Assume your parsing was incorrectly chopping off URL's value after the first '?' symbol (actually, there are good use cases to do that on purpose)
    b. This would lead to these two URLs to be considered the same: /wiki.pl?WxWidgets_Bounties and /wiki.pl?WxWidgets_Compared_To_Other_Toolkits
    c. The counts for all url-related queries would be wrong and not recoverable

Conclusion: you have to store your original raw log as well as parsed/interpreted data
[obviously this is not a set-in-stone rule - and depends on your business requirements]

V2:

timestamp	url	clientID	raw_event
19/Jun/2005:06:44:17 +0200	/wximages/wxwidgets02-small.png	222.64.146.118	222.64.146.118 - - [19/Jun/2005:06:44:17 +0200] "GET /wximages/wxwidgets02-small.png HTTP/1.1" 200 12468 "http://blog.vckbase.com/bastet/" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; TencentTraveler)"
19/Jun/2005:06:46:05 +0200	/wximages/wxwidgets02-small.png	218.84.191.50	218.84.191.50 - - [19/Jun/2005:06:46:05 +0200] "GET /wximages/wxwidgets02-small.png HTTP/1.1" 200 12468 "http://blog.vckbase.com/bastet/" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
19/Jun/2005:06:48:40 +0200	/wiki.pl?WxWidgets_Bounties	222.64.146.118	138.243.201.10 - - [19/Jun/2005:06:48:40 +0200] "GET /wiki.pl?WxWidgets_Bounties HTTP/1.1" 200 8873 "http://www.wxwidgets.org/toolbar.htm" "Mozilla/5.0 (Windows; U; Windows NT 5.1; en-GB; rv:1.7.7) Gecko/20050414 Firefox/1.0.3"
19/Jun/2005:06:50:49 +0200	/wiki.pl?WxWidgets_Compared_To_Other_Toolkits	68.251.52.253	68.251.52.253 - - [19/Jun/2005:06:50:49 +0200] "GET /wiki.pl?WxWidgets_Compared_To_Other_Toolkits HTTP/1.1" 200 19476 "http://www.google.com/search?q=wxWidget+designer" "Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.7.8) Gecko/20050511 Firefox/1.0.4"

Master Datasets - Requirements and Modeling

Unstructured vs Semantically Normalized ??

What we saw above is a manifestation of a more general concept: **semantic normalization** - which is a process of reshaping a free form information into a structured form of data.

Consider a simple example:
Your web app collects clicks of each user, and it also asks each user to enter their location in a free-form text box. You implemented algorithm to normalize the location data into a standard <city, state, country> format using some public location lookup datasets:

Master Datasets - Requirements and Modeling

V1 of your location normalization code does not recognize "Back Bay" as a neighborhood in Boston ...

User entered string	Normalized location
Boston, MA	Boston, MA, USA
Boston, 02141	Boston, MA, USA
Back Bay	NULL

…. but a later version might:

User entered string	Normalized location
Boston, MA	Boston, MA, USA
Boston, 02141	Boston, MA, USA
Back Bay	Boston, MA, USA

Master Datasets - Requirements and Modeling

In the earlier log example - one piece of information that is potentially a "moving [for interpretation] target" is the UserAgent part:

```
222.64.146.118 - - [19/Jun/2005:06:44:17 +0200] "GET /wximages/wxwidgets02-small.png HTTP/1.1" 200 12468
"http://blog.vckbase.com/bastet/" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; TencentTraveler )"
```

Rules of Thumb:

- if your algorithm for parsing/extracting the data is simple and accurate (like determining the HTTP request type from the Apache log - GET/ POST/PUT …) or getting a clear field value from a web page form (like first and last names) - store the results of extraction as your Master Data
- if the algorithm is subject to change , due to improvements or changed requirements, store the unstructured data
- and if you are not sure which case it is - store both :) (but don't go crazy….)

Master Datasets - Requirements and Modeling

data should be **immutable**
- *CAN NOT delete or update*
- *CAN add only*

data should be **timestamped**

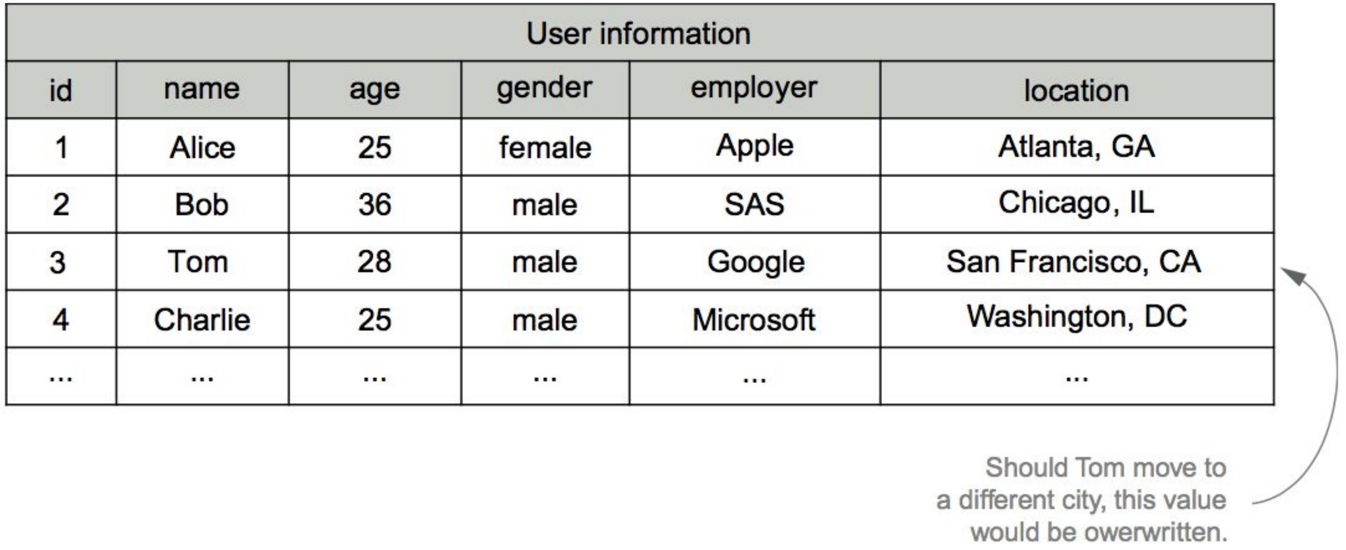
data should be **eternally TRUE !**

why?
- simplicity for storage layer: no random read/write/index functionality is required, only append
- can provide historical trends
- human-fault tolerance

Master Datasets - Requirements and Modeling

Lets consider an Example (from the "Big Data" book by Nathan Martz)

Mutable schema



User information					
id	name	age	gender	employer	location
1	Alice	25	female	Apple	Atlanta, GA
2	Bob	36	male	SAS	Chicago, IL
3	Tom	28	male	Google	San Francisco, CA
4	Charlie	25	male	Microsoft	Washington, DC
...	...	...	...	...	...

Should Tom move to a different city, this value would be owerwritten.

@Marina Popova

Master Datasets - Requirements and Modeling

Immutable
Timestamped
Eternally True
de-normalized

user_id	name	age	gender	employer	location	timestamp
1	Alice	25	F	Apple	Atlanta	2012/03/29 08:12:24
2	Bob	36	M	SAS	Chicago	2012/04/12 14:47:51
3	Tom	28	M	Googe	San Fran	2012/04/04 18:31:24
3	Tom	28	M	Google	Los Angeles	2012/06/17 20:09:48
3	Tom	28	M	Facebook	Los Angeles	2012/06/18 20:00:00

Master Datasets - Requirements and Modeling

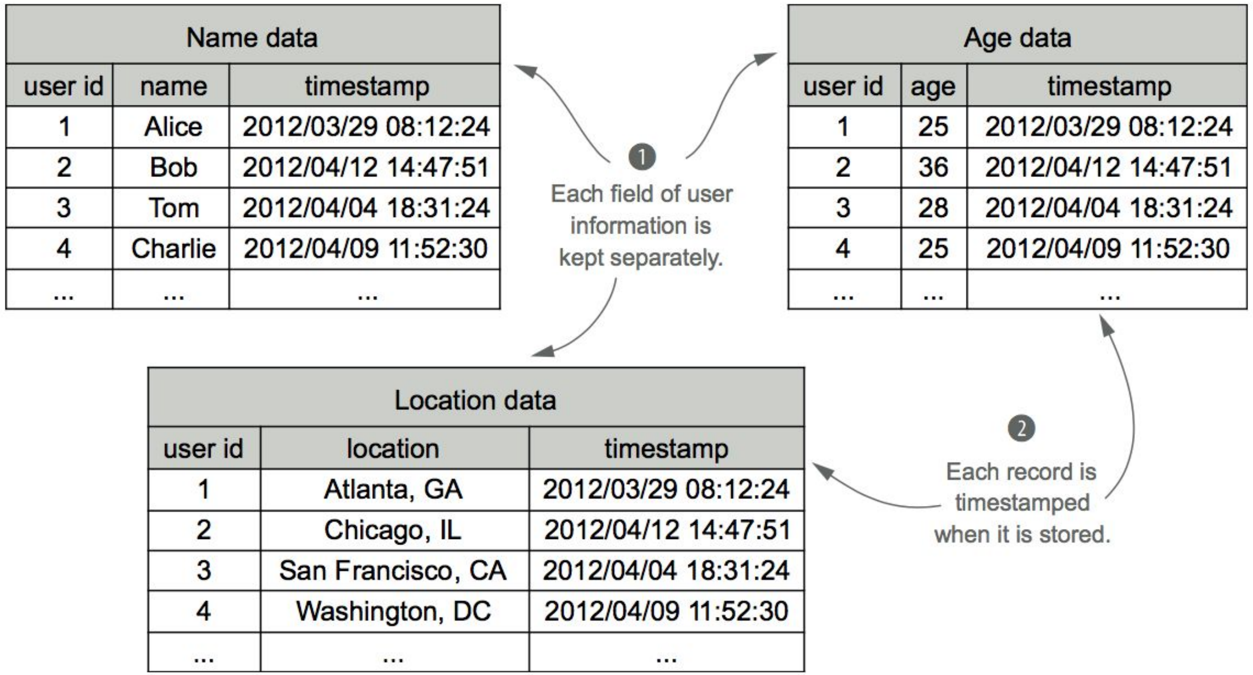
Normalized (Star, Facts) vs De-normalized ??

- the model above was de-nomalized
- we could choose a **Fact-based Model** for representing data instead:
  - data is deconstructed into fundamental units called **facts** which are:
    - atomic (cannot be divided into smaller pieces)
    - timestamped
  - no data duplication between facts

same example in a Normalized schema:

Master Datasets - Requirements and Modeling

Immutable
Timestamped
Eternally True
Normalized



Name data		
user id	name	timestamp
1	Alice	2012/03/29 08:12:24
2	Bob	2012/04/12 14:47:51
3	Tom	2012/04/04 18:31:24
4	Charlie	2012/04/09 11:52:30
...	...	...

Age data		
user id	age	timestamp
1	25	2012/03/29 08:12:24
2	36	2012/04/12 14:47:51
3	28	2012/04/04 18:31:24
4	25	2012/04/09 11:52:30
...	...	...

Location data		
user id	location	timestamp
1	Atlanta, GA	2012/03/29 08:12:24
2	Chicago, IL	2012/04/12 14:47:51
3	San Francisco, CA	2012/04/04 18:31:24
4	Washington, DC	2012/04/09 11:52:30
...	...	...

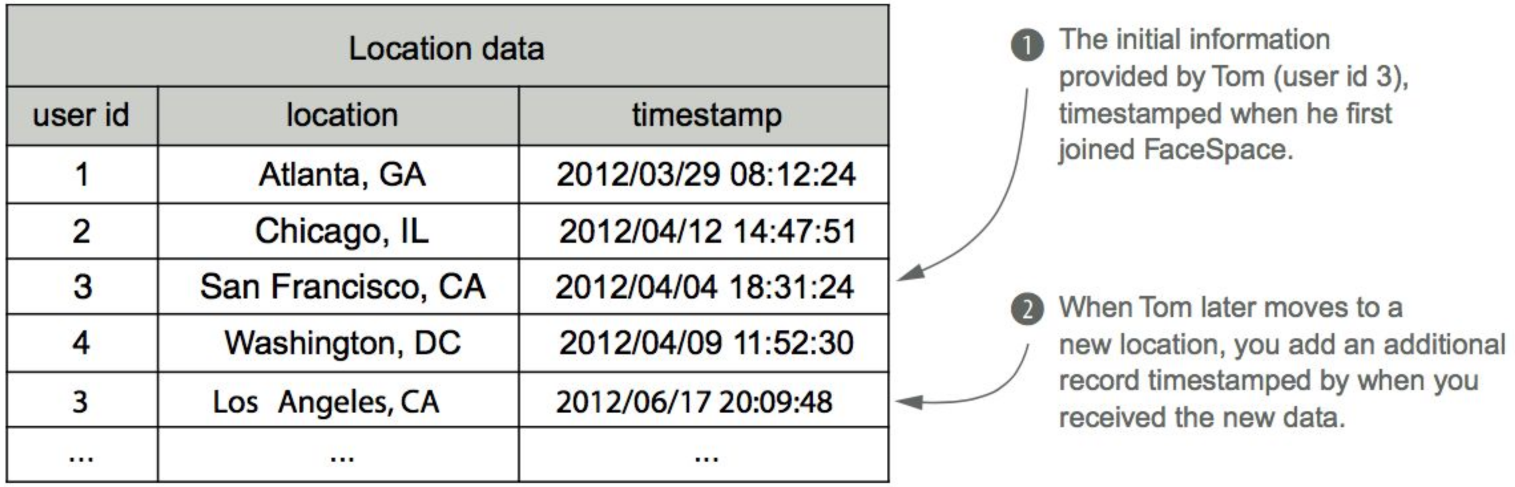
1 Each field of user information is kept separately.

2 Each record is timestamped when it is stored.

@Marina Popova

Master Datasets - Requirements and Modeling

When user 'Tom' moves to a new location -
you simply add a new record with the corresponding timestamp:



Location data		
user id	location	timestamp
1	Atlanta, GA	2012/03/29 08:12:24
2	Chicago, IL	2012/04/12 14:47:51
3	San Francisco, CA	2012/04/04 18:31:24
4	Washington, DC	2012/04/09 11:52:30
3	Los Angeles, CA	2012/06/17 20:09:48
...	...	...

1. The initial information provided by Tom (user id 3), timestamped when he first joined FaceSpace.
2. When Tom later moves to a new location, you add an additional record timestamped by when you received the new data.

@Marina Popova

Master Datasets - Requirements and Modeling

Normalized (Star, Facts) vs De-normalized ??

Normalized Pros - De-normalized Cons

- analyze sub-groups of facts independently
- data storage volume/cost concerns
- have to use hierarchical models - joins

Normalized Cons - De-normalized Pros

- store unstructured or less-structured data like logs
- efficient bulk processing
- queries do not require joins

Master Datasets - Requirements and Modeling

data should be **identifiable**

Timestamp is not enough to uniquely identify each event/fact - why?

- precision might not be enough: timestamp may be of a second precision, but you could have two [the same otherwise] events happening during the same millisecond

```
222.64.146.118 - - [19/Jun/2005:06:44:17 +0200] "GET /wximages/wxwidgets02-small.png HTTP/1.1" 200 12468 "http://blog.vckbase.com/bastet/"
"Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; TencentTraveler )"
222.64.146.118 - - [19/Jun/2005:06:44:17 +0200] "GET /wximages/wxwidgets02-small.png HTTP/1.1" 200 12468 "http://blog.vckbase.com/bastet/"
"Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; TencentTraveler )"
```

- events can be duplicated due to network partitioning between data collection apps/nodes and master data store
  - when the network connectivity is restored - collection app has to re-send the data from the last batch, without knowing whether the batch was partially processed by the data store or not

Master Datasets - Requirements and Modeling

Solution:

add a UUID for each event - either in the data generation layer (if you have control over it) or in the first processing/collecting layer

- In Java, such UUID could easily be generated using the java.util.UUID utility: UUID.randomUUID()
- Example generated UUIDs:
- `067e6162-3b6f-4ae2-a171-2470b63dff00`
- `54947df8-0e9e-4471-a2f9-9af509fb5889`

Raw Data Storage
V3:

UUID	timestamp	url	clientID	raw_event
uuid1	19/Jun/2005:06:44:17 +0200	/wximages/wxwidgets02-small.png	222.64.146.118	222.64.146.118 - - [19/Jun/2005:06:44:17 +0200] "GET /wximages/wxwidgets02-small.png HTTP/1.1" 200 12468 "http://blog.vckbase.com/bastet/" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; TencentTraveler)"
uuid2	19/Jun/2005:06:46:05 +0200	/wximages/wxwidgets02-small.png	218.84.191.50	218.84.191.50 - - [19/Jun/2005:06:46:05 +0200] "GET /wximages/wxwidgets02-small.png HTTP/1.1" 200 12468 "http://blog.vckbase.com/bastet/" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)"
uuid3	19/Jun/2005:06:48:40 +0200	/wiki.pl?WxWidgets_Bounties	222.64.146.118	138.243.201.10 - - [19/Jun/2005:06:48:40 +0200] "GET /wiki.pl?WxWidgets_Bounties HTTP/1.1" 200 8873 "http://www.wxwidgets.org/toolbar.htm" "Mozilla/5.0 (Windows; U; Windows NT 5.1; en-GB; rv:1.7.7) Gecko/20050414 Firefox/1.0.3"
uuid4	19/Jun/2005:06:50:49 +0200	/wiki.pl?WxWidgets_Compared_To_Other_Toolkits	68.251.52.253	68.251.52.253 - - [19/Jun/2005:06:50:49 +0200] "GET /wiki.pl?WxWidgets_Compared_To_Other_Toolkits HTTP/1.1" 200 19476 "http://www.google.com/search?q=wxWidget+designer" "Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.7.8) Gecko/20050511 Firefox/1.0.4"

Master Datasets - Storage Requirements

✓ What to store (content and model) ?
○ Where to store to (the actual data storage systems) ?
○ How to store - formats and serialization ?
○ How to store - algorithms and operations ?

First, lets identify requirements for a data storage suitable for Master Datasets

Master Datasets - Storage Requirements

The most important requirement for Master data set is: **NO DATA LOSS !!!**

When identifying requirements for any data storage, the main questions to ask are:

1. What are the write patterns for your data?
2. What are the read patterns?
3. What are the latency requirements for both ?

Master Datasets - Storage Requirements

Given the already identified requirements for the master data, here are the most important info on **write patterns**:

- Data is immutable
- Data is eternally true
- Data is constantly growing by means of adding new data
- Data has to be stored as fast as it comes
- Data should be compressible - due to the size

This dictates the following WRITE requirements::

- the master storage has to be able to append new bulk data FAST
- the master storage has to be scalable: able to handle very large (Ts and Ps of data) , growing data sets
- ability to compress the data
- does NOT need to support updates
- nice-to-have: ability to enforce immutability (for example, do not allow updates of data)

Master Datasets - Storage Requirements

Read-related considerations:

- Due to the huge size - we will want to read batches of data, not individual items
- To batch-process the data - the processing has to be parallelized

This dictates the following requirements:

- support for bulk reads
- support for parallel reads
- support for data partitioning/chunking
- does NOT need to support random reads

Master Datasets - Storage Options

Storage Options

There are many storage systems available these days - and we will review more generic requirements/ characteristics and classes of such systems in the following lectures.

For now - we will focus on two main types of such systems

- Key-Value Storage Systems
- Distributed File Systems

Master Datasets - Storage Options

Side Note:
Key-Value WIde-Column DBs (like Cassandra) do not have the same characteristics and are suitable for master data storage - if data volume and/or budget are acceptable

Key-Value Storage Systems

Main characteristics:

- Each data fact has to be stored as a key/value pair
    - What is value? Your data
    - What is key? Good question! :-)
- Provide fine-grained access to each individual key, which means:
    - Cannot bulk-compress
    - Keys are indexed
- Mostly, they are meant for mutable storage, which means they cannot enforce data immutability
- Provide random read/write access, which means:
    - Have very complex implementation
    - Performance of reads/writes can be affected

Conclusion: not an ideal storage option for very large master datasets

Master Datasets - Storage Options

Distributed File Systems

Key characteristics of distributed FS:

- Files are sequences of bytes, stored sequentially on disk
- Can read and write them sequentially
- Can compress
- Have permission system to enforce immutability
- Unlike regular filesystems, which are limited to one server, distributed FS are implemented over multiple nodes:
    - this means they are scalable horizontally by adding more nodes!
- Fault-tolerant to a single node failure
- Can NOT access file in the middle (no random reads)
- Can NOT modify existing files (there are other ways to delete old and add new data)

Conclusion: Distributed FS satisfies all of the requirements for the master data storage

Example distributed filesystems: HDFS, S3

Master Datasets

Reminder for Master Data Storage - the important aspects to address are:

- ✓ What to store (content and model) ?
- ✓ Where to store to (the actual data storage systems) ?
- ○ How to store - formats and serialization ?
- ○ How to store - algorithms and operations ?

We will focus on the third aspect in the next Lecture!