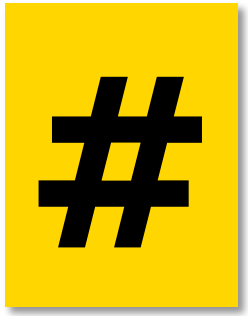

hadoop



MAP REDUCE


BIG
DATA
INFORMATION
TECHNOLOGIES
MANAGEMENT
ANALYSIS
RESEARCH
SEARCH
PETABYTES
STORAGE
LARGE
SETS
MILLION
SOFTWARE
SCIENCE
FUTURE
DEVELOPMENT
TECHNOLOGY
PROCESS
VOLUME
GOVERNMENT
INTERNET
LEARNING
BASED
NEW
BILLION
PARALLEL
SOCIAL
NAS
SAN
SIZE
DATABASES
PROCESSING
EXABYTES
AMOUNT
REAL
WORLD
EVERY
INITIATIVE
TERABYTES
SENSOR
MAPREDUCE

Big Data Definition

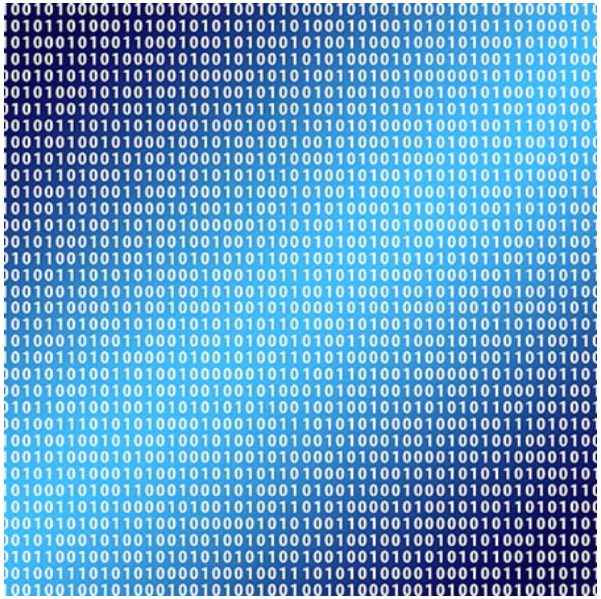
- Big Data is a collection of large datasets that cannot be processed using traditional computing techniques.

- Any data that can challenge our current technology

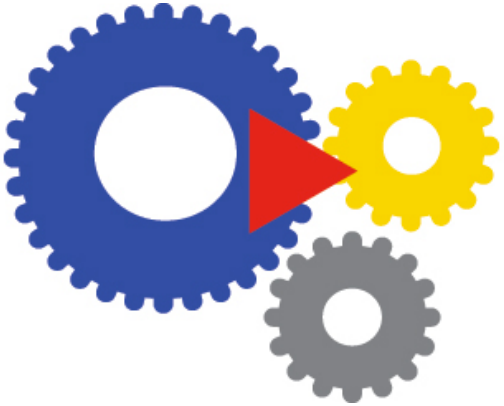
Big Data

Big data is about the Insight that we want to extract from information.

Data



Computing



Big Data Frameworks properties

- Big Data Frameworks properties
  - Elasticity
  - Errors Transparency
  - Scalability

Abstraction

- The primary MapReduce abstraction is parallelizable, easy to understand, and hides the details of distributed computing, thus allowing Big Data engine to guarantee correctness.

- Specification of computation as a mapping then a reduction.

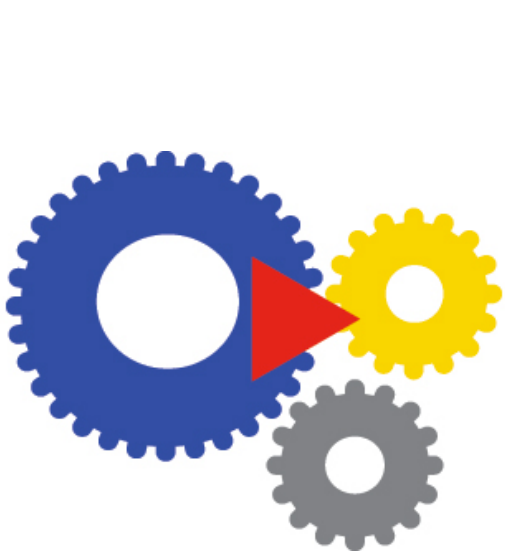
MapReduce

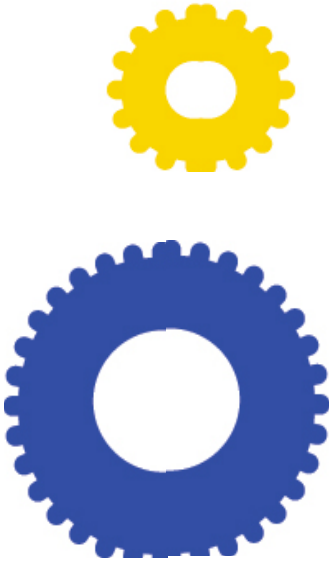
- Map Reduce
  - Inspired from functional programming paradigms.
- Two functions
  - Map : specify the per-reccord computation
  - Reduce : specify result aggregation

- MapReduce :
  - Theoretical framework.
- Hadoop MapReduce Framework :
  - open source implementation

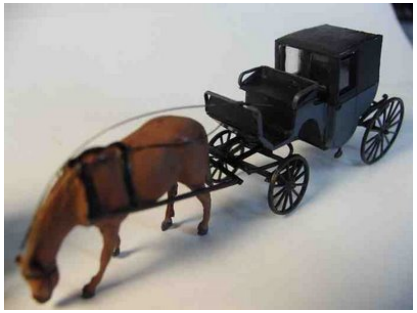
MapReduce

- Re-think the best approach to an old problem.
- Need to "unlearn" things in order to full utilize the power of a framework.
- There may be even better solutions that we didn't discovered yet.





Interface & Encapsulation = New paradigm



Carroserie Hypomobile

Domain Maturity

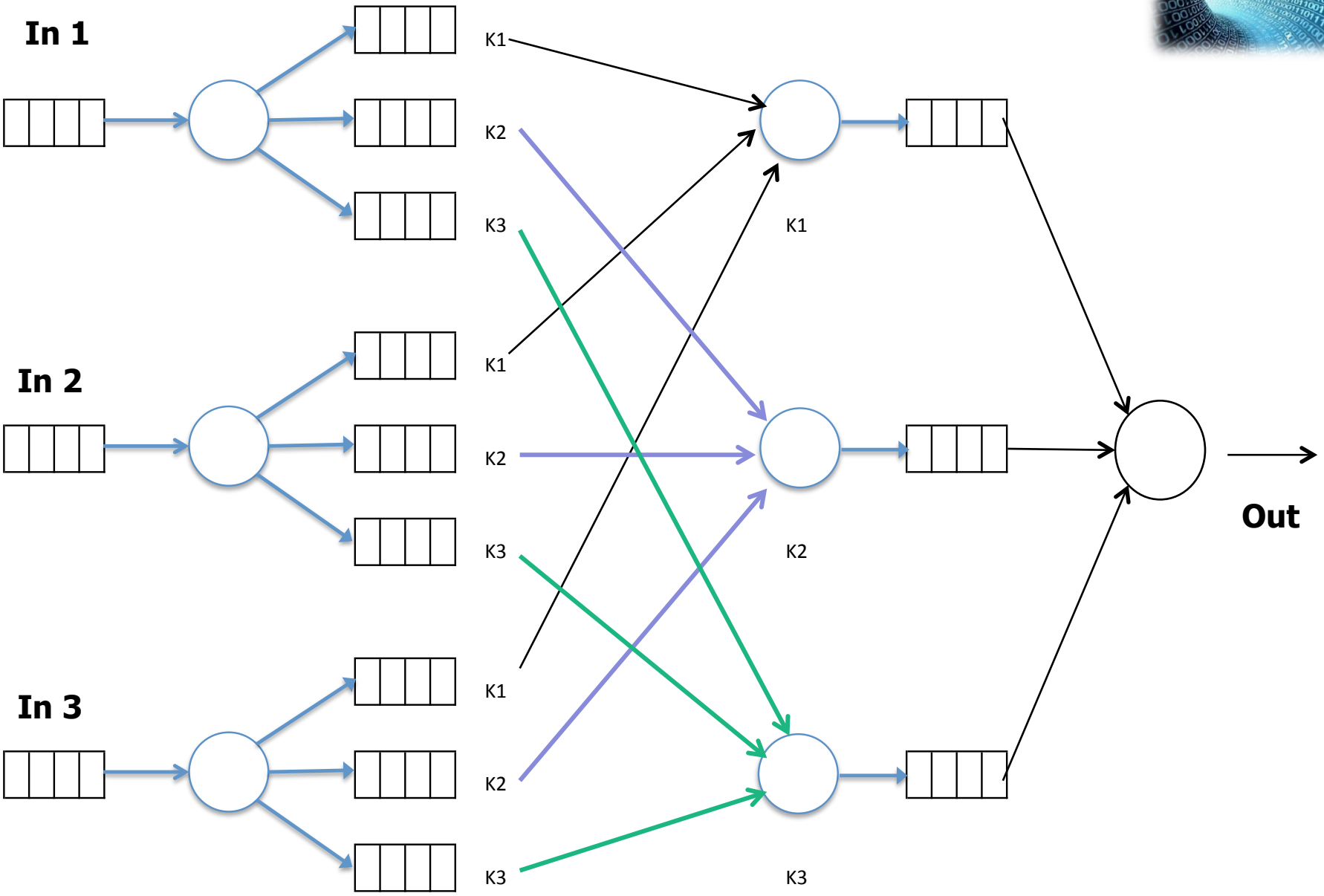
Traduction en cours

Back to the past !!!

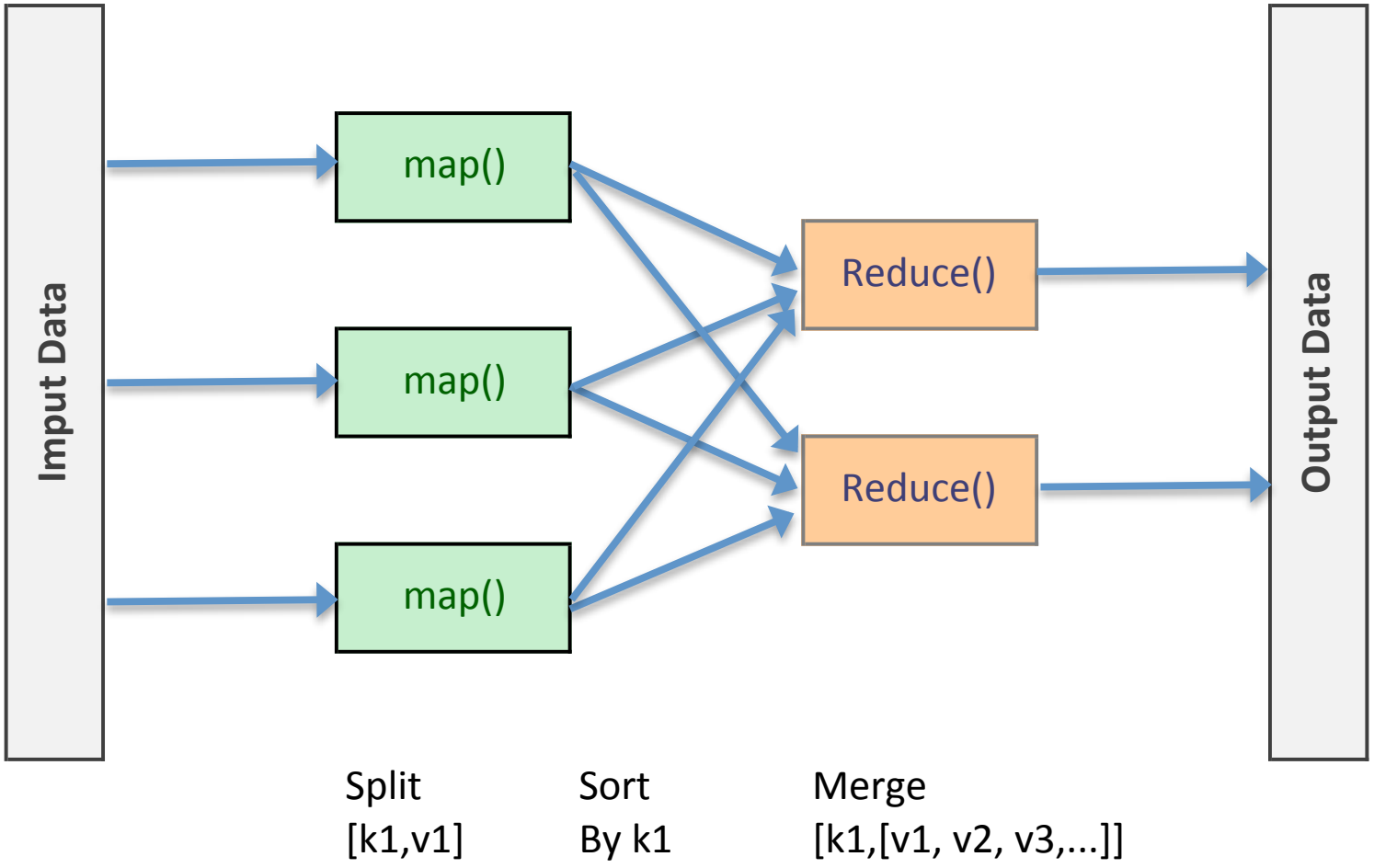
MAP REDUCE

http://martinfowler.com/articles/nosqlKeyPoints.html

- The map task reads data from an aggregate and boils it down to relevant key-value pairs.
- Maps only read a single record at a time and can thus be parallelized and run on the node that stores the record.
- Reduce tasks take many values for a single key output from map tasks and summarize them into a single output.
- Each reducer operates on the result of a single key, so it can be parallelized by key.


In 1
In 2
In 3
K1
K2
K3
K1
K2
K3
K1
K2
K3
Out

MapReduce


Imput Data
map()
map()
map()
Reduce()
Reduce()
Output Data
Split
[k1,v1]
Sort
By k1
Merge
[k1,[v1, v2, v3,...]]

if you have a hammer, everything looks like a nail.

Map Reduce

- Tranform lists of input data elements into lists of output data elements.

- Twice with two different list processsing approach :

  - 1) MAP : Partitioning

  - 2) Reduce : Merge Select (join)

Traduction en cours

Lists

- Key Value Lists.
  - 1) MAP : Partitioning on Key
  - 2) Reduce : Merge operation on Value

Map Reduce Canonical Exemple

- How many times different words appear in a set of sentences ?

Counting Words

Pangrams

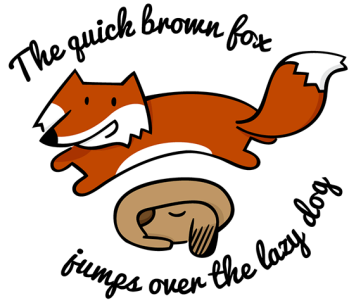
- Pangrams :
  - The quick brown fox jumps over the lazy dog.
  - (35 letters)

- A pangram, or holoalphabetic sentence, includes every letter of the alphabet at least once.

- "Portez ce vieux whisky au juge blond qui fume"

The Fox

The
quick
brown
fox
jumps
over
the
lazy
dog.



Map

Key	Value
The	1
quick	1
brown	1
fox	1
jumps	1
over	1
the	1
lazy	1
dog.	1

Map

Key	Value
The	1
quick	1
brown	1
fox	1
jumps	1
over	1
the	1
lazy	1
dog.	1

- The map task reads data from an aggregate and boils it down to relevant key-value pairs.

Sentences

- Dogs That Look Like a Fox
- How to Stop Your Dog Jumping on Guests
- Dog Who Looks Like A Blue Eyed Fox

Map

Dogs
That
Look
Like
a
Fox

How
to
Stop
Your
Dog
Jumping
on
Guests

Dog
Who
Looks
Like
A
Blue
Eyed
Fox

Key	Value
Dogs	1
That	1
Look	1
Like	1
a	1
Fox	1

Key	Value
How	1
to	1
Stop	1
Your	1
Dog	1
Jumping	1
on	1
Guests	1

Key	Value
Dog	1
Who	1
Looks	1
Like	1
A	1
Blue	1
Eyed	1
Fox	1

Map 1

Key	Value
brown	1
dog.	1
fox	1
jumps	1
lazy	1
over	1
quick	1
The	1
the	1

MAP

brown	dog.	fox	jumps	lazy	over	quick	The	the
1	1	1	1	1	1	1	1	1

Map 1

Key	Value
brown	1
dog.	1
fox	1
jumps	1
lazy	1
over	1
quick	1
The	1
the	1

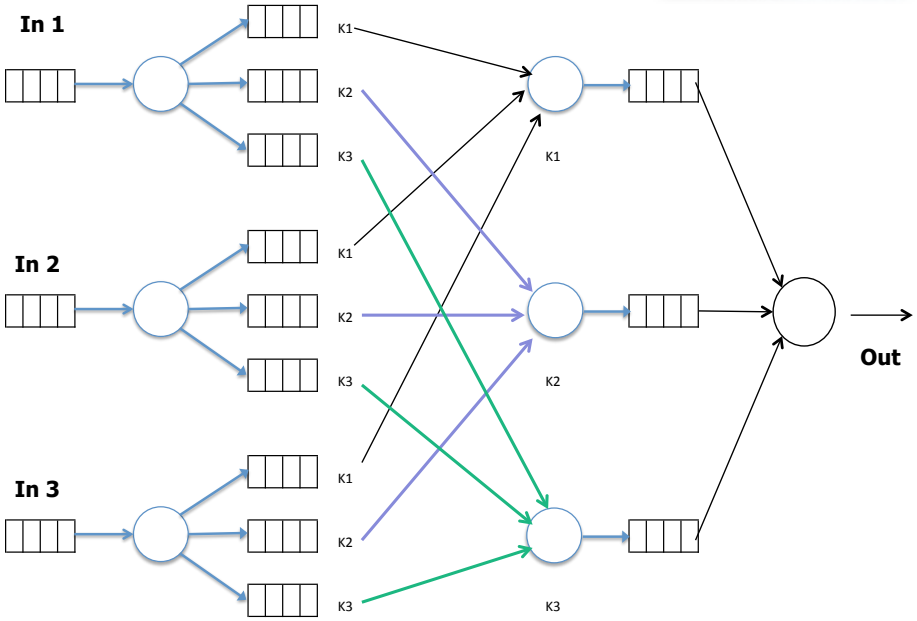
MAP

brown	dog.	fox	jumps	lazy	over	quick	The	the
1	1	1	1	1	1	1	1	1

Partitioning

brown	dog.	fox	jumps	lazy	over	quick	The	the
1	1	1	1	1	1	1	1	1

Map 1



Key	Value
brown	1
dog.	1
fox	1
jumps	1
lazy	1
over	1
quick	1
The	1
the	1

MAP

brown	dog.	fox	jumps	lazy	over	quick	The	the
1	1	1	1	1	1	1	1	1

Partitioning

brown	dog.	fox	jumps	lazy	over	quick	The	the
1	1	1	1	1	1	1	1	1

Map 2

Key	Value
That	1
Look	1
Like	1
Fox	1
Dogs	1
a	1

MAP

That	Look	Like	Fox	Dogs	a
1	1	1	1	1	1

Partitioning

That	Look	Like	Fox	Dogs	a
1	1	1	1	1	1

Map 2

Key	Value
That	1
Look	1
Like	1
Fox	1
Dogs	1
a	1

MAP

That	Look	Like	Fox	Dogs	a
1	1	1	1	1	1

Partitioning

That	Look	Like	Fox	Dogs	a
1	1	1	1	1	1

Map 3

Key	Value
Dog	1
Guests	1
How	1
Jumping	1
on	1
Stop	1
to	1
Your	1

MAP

Dog	Guests	How	Jumping	on	Stop	to	Your
1	1	1	1	1	1	1	1

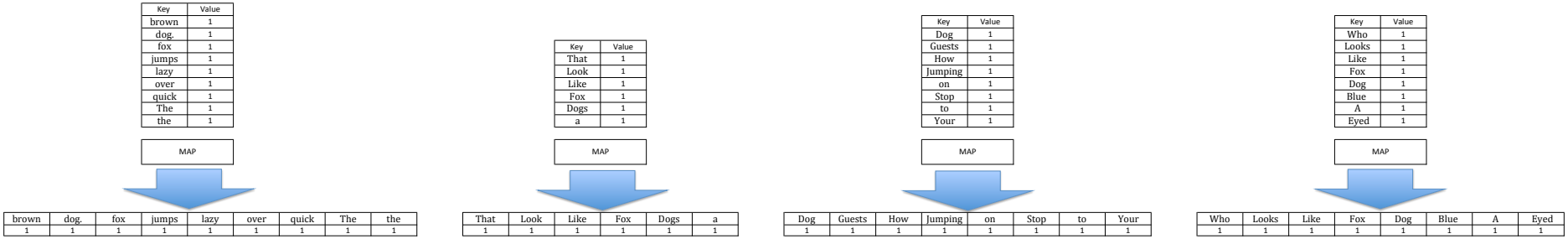
Map 4

Key	Value
Who	1
Looks	1
Like	1
Fox	1
Dog	1
Blue	1
A	1
Eyed	1

MAP

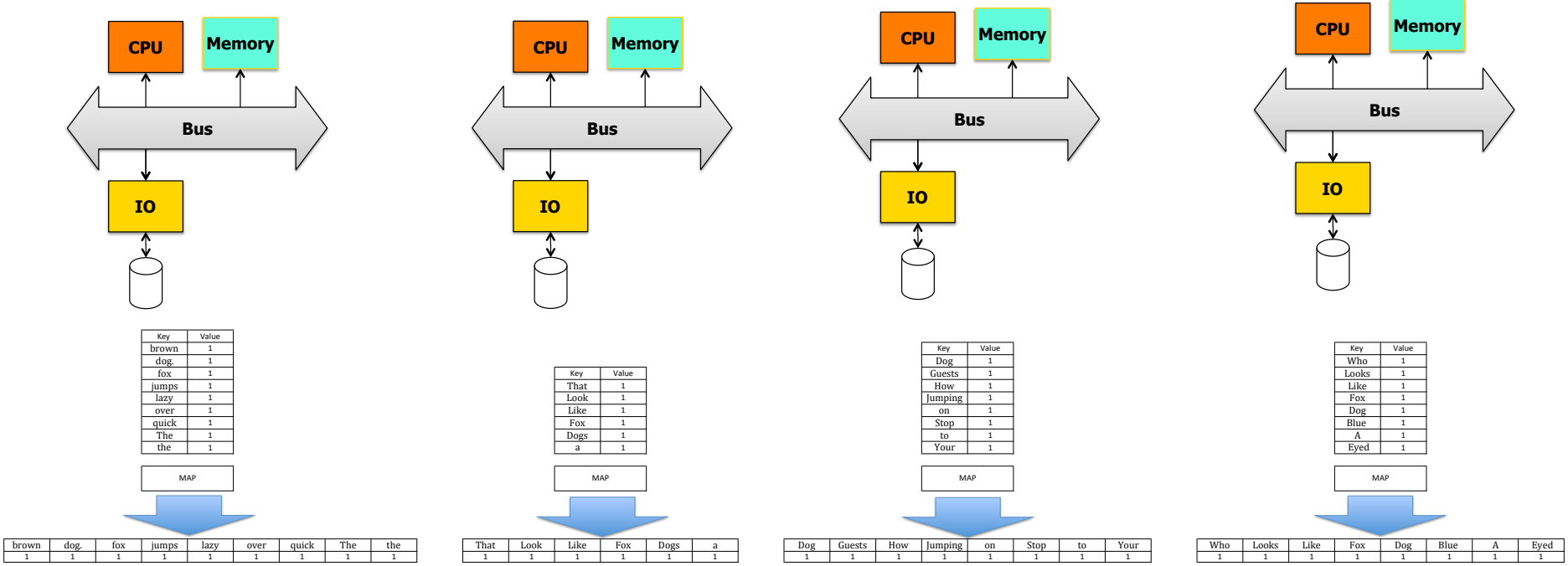
Who	Looks	Like	Fox	Dog	Blue	A	Eyed
1	1	1	1	1	1	1	1

Map



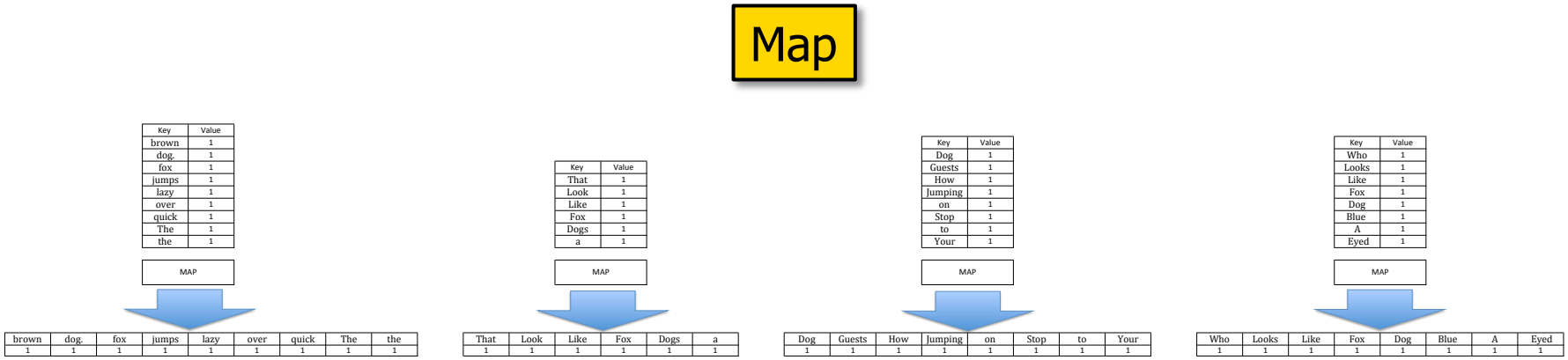
- Maps only read a single record at a time and can thus be parallelized and run on the node that stores the record.

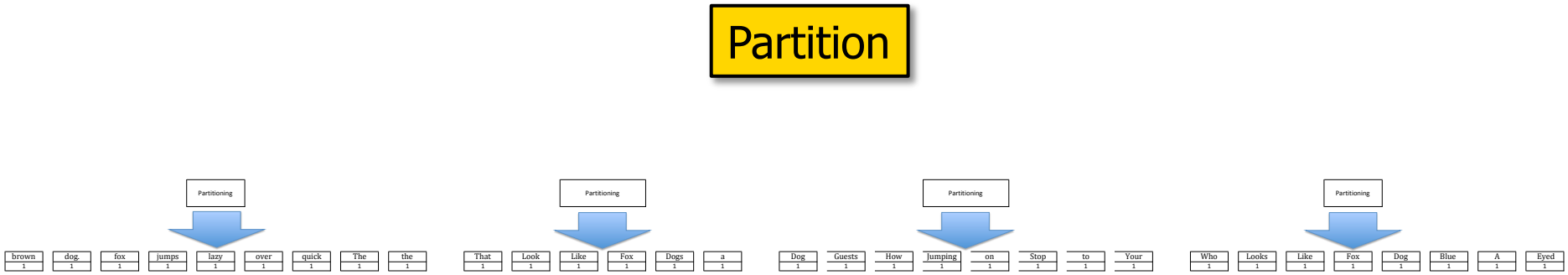
Map



- Maps only read a single record at a time and can thus be parallelized and run on the node that stores the record.

Map


Map
Key Value
brown 1
dog. 1
fox 1
jumps 1
lazy 1
over 1
quick 1
The 1
the 1
MAP
brown dog. fox jumps lazy over quick The the
1 1 1 1 1 1 1 1 1
Key Value
That 1
Look 1
Like 1
Fox 1
Dogs 1
a 1
MAP
That Look Like Fox Dogs a
1 1 1 1 1 1
Key Value
Dog 1
Guests 1
How 1
Jumping 1
on 1
Stop 1
to 1
Your 1
MAP
Dog Guests How Jumping on Stop to Your
1 1 1 1 1 1 1 1
Key Value
Who 1
Looks 1
Like 1
Fox 1
Dog 1
Blue 1
A 1
Eyed 1
MAP
Who Looks Like Fox Dog Blue A Eyed
1 1 1 1 1 1 1 1


Partition
Partitioning
brown dog. fox jumps lazy over quick The the
1 1 1 1 1 1 1 1 1
Partitioning
That Look Like Fox Dogs a
1 1 1 1 1 1
Partitioning
Dog Guests How Jumping on Stop to Your
1 1 1 1 1 1 1 1
Partitioning
Who Looks Like Fox Dog Blue A Eyed
1 1 1 1 1 1 1 1

Key Partitioning

Fox	1
Fox	1
Fox	1

A	1
A	1

Like	1
Like	1

Your	1

Who	1

to	1

The	1

the	1

That	1

Stop	1

quick	1

over	1

on	1

Looks	1

Look	1

lazy	1

jumps	1

Jumping	1

How	1

Guests	1

Dogs	1

brown	1

Blue	1

Eyed	1

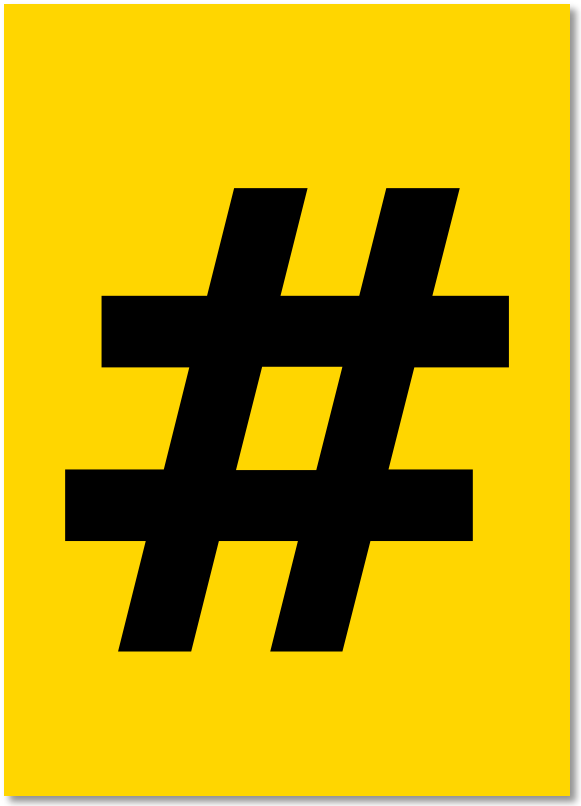
Dog	1
Dog	1
Dog	1

Fox	1
Fox	1
Fox	1

A	1
A	1

Like	1
Like	1

Key Partitioning



Fox	1
Fox	1
Fox	1
A	1
A	1
Like	1
Like	1
Your	1
Who	1
to	1
The	1
the	1
That	1
Stop	1
quick	1
over	1
on	1
Looks	1
Look	1
lazy	1
jumps	1
Jumping	1
How	1
Guests	1
Dogs	1
brown	1
Blue	1
Eyed	1

Dog	1
Dog	1
Dog	1

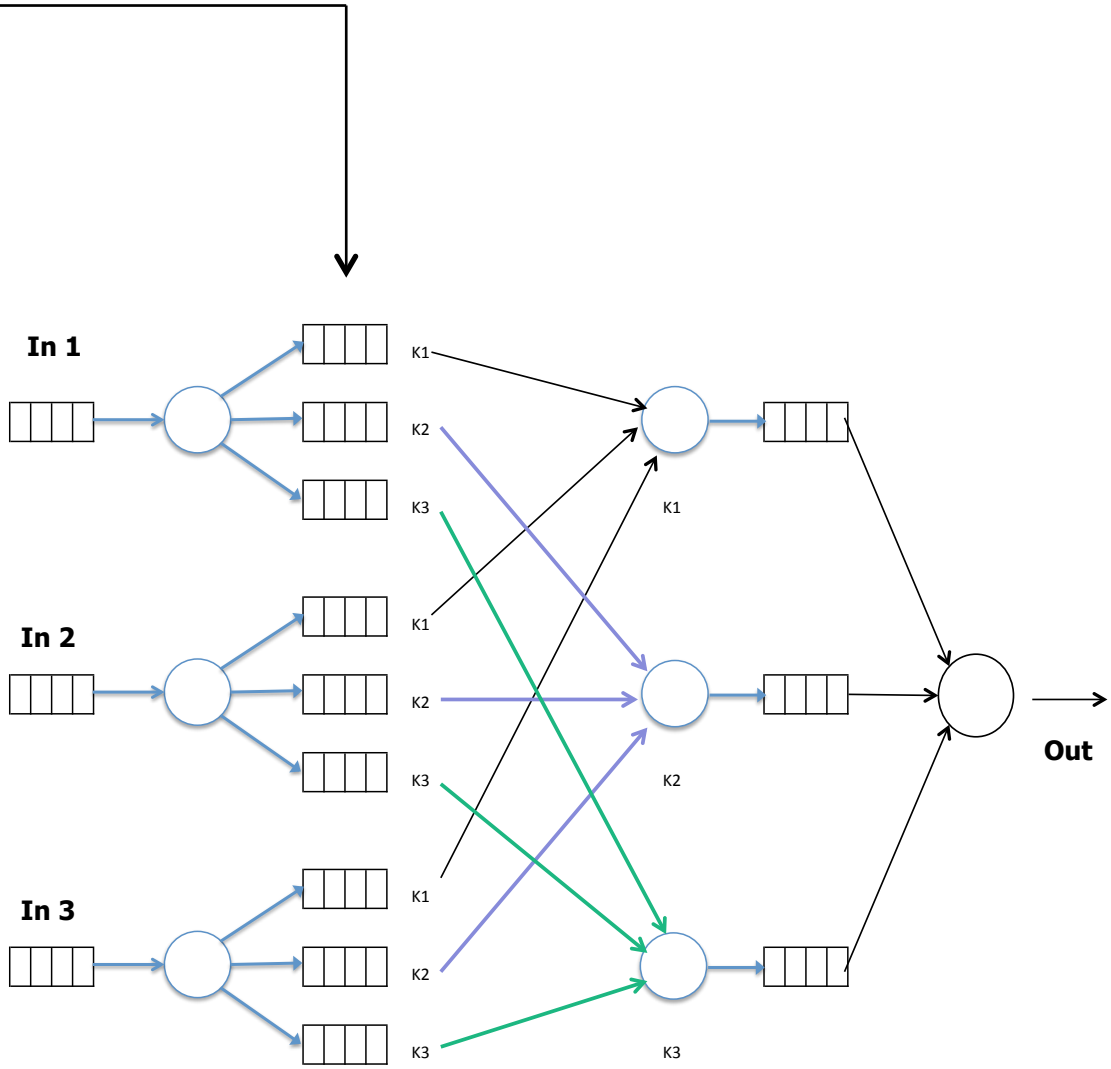
Fox	1
Fox	1
Fox	1

A	1
A	1

Like	1
Like	1

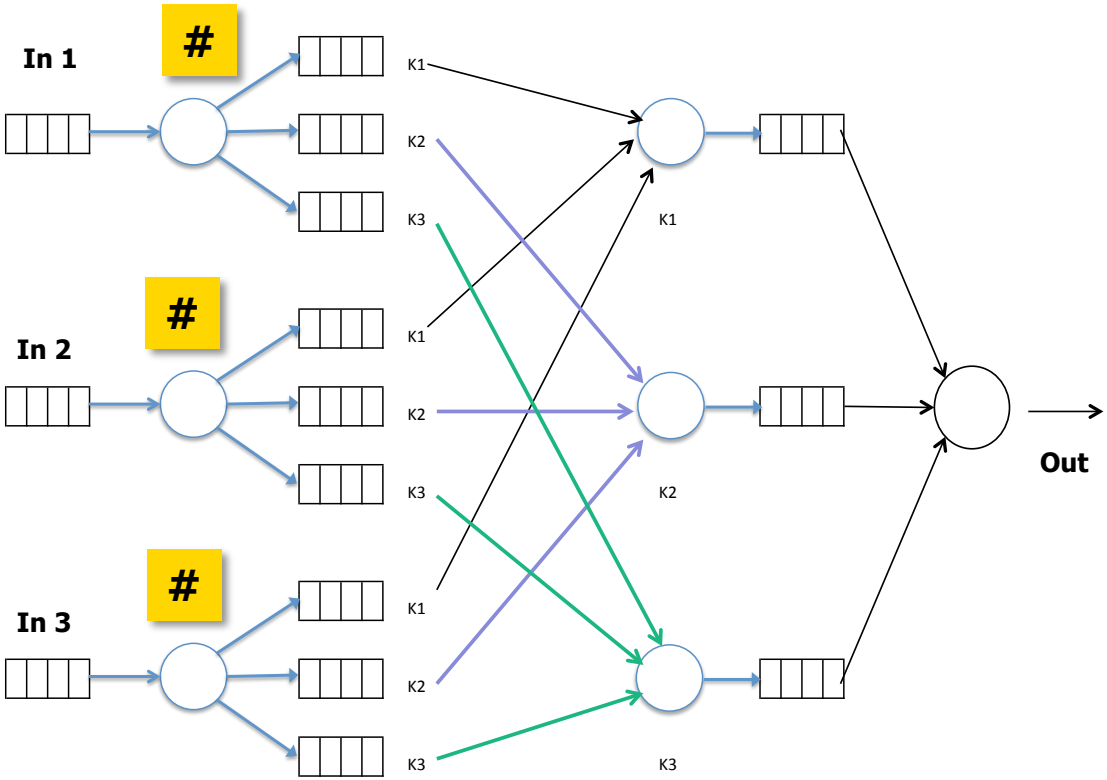
Key Partitioning

Key	Value
brown	1
dog.	1
fox	1
jumps	1
lazy	1
over	1
quick	1
The	1
the	1



Key Partitioning

Key	Value
brown	1
dog.	1
fox	1
jumps	1
lazy	1
over	1
quick	1
The	1
the	1



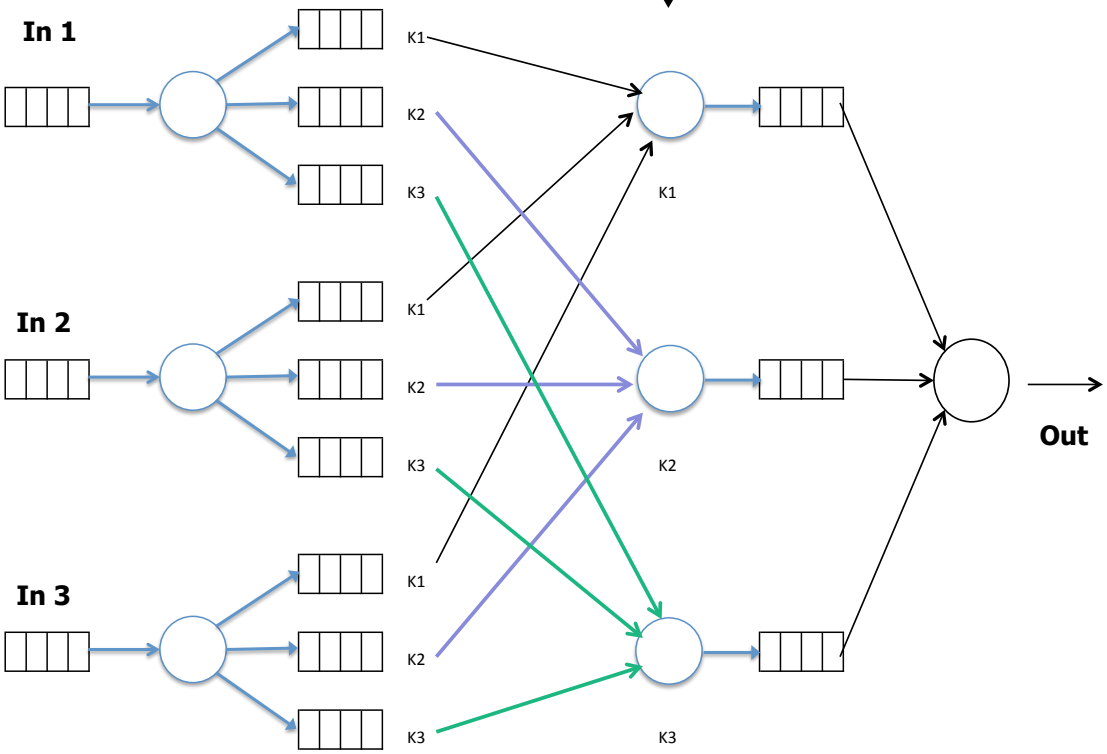
Key Partitioning

Dog	1
Dog	1
Dog	1

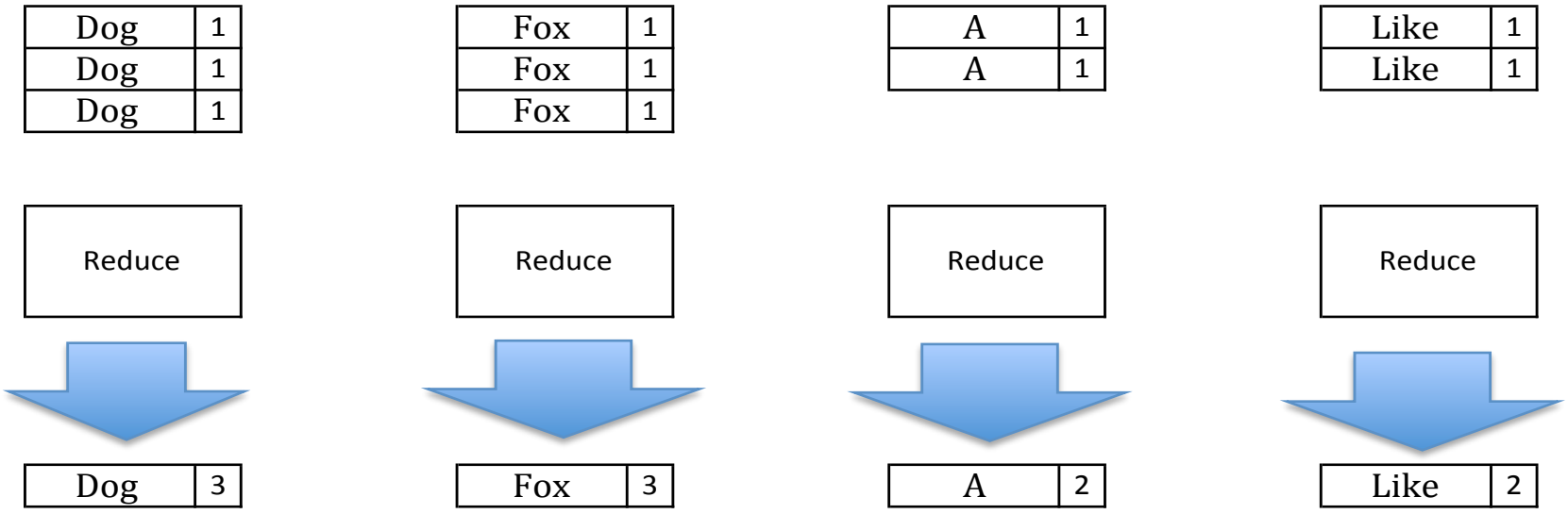
Fox	1
Fox	1
Fox	1

A	1
A	1

Like	1
Like	1

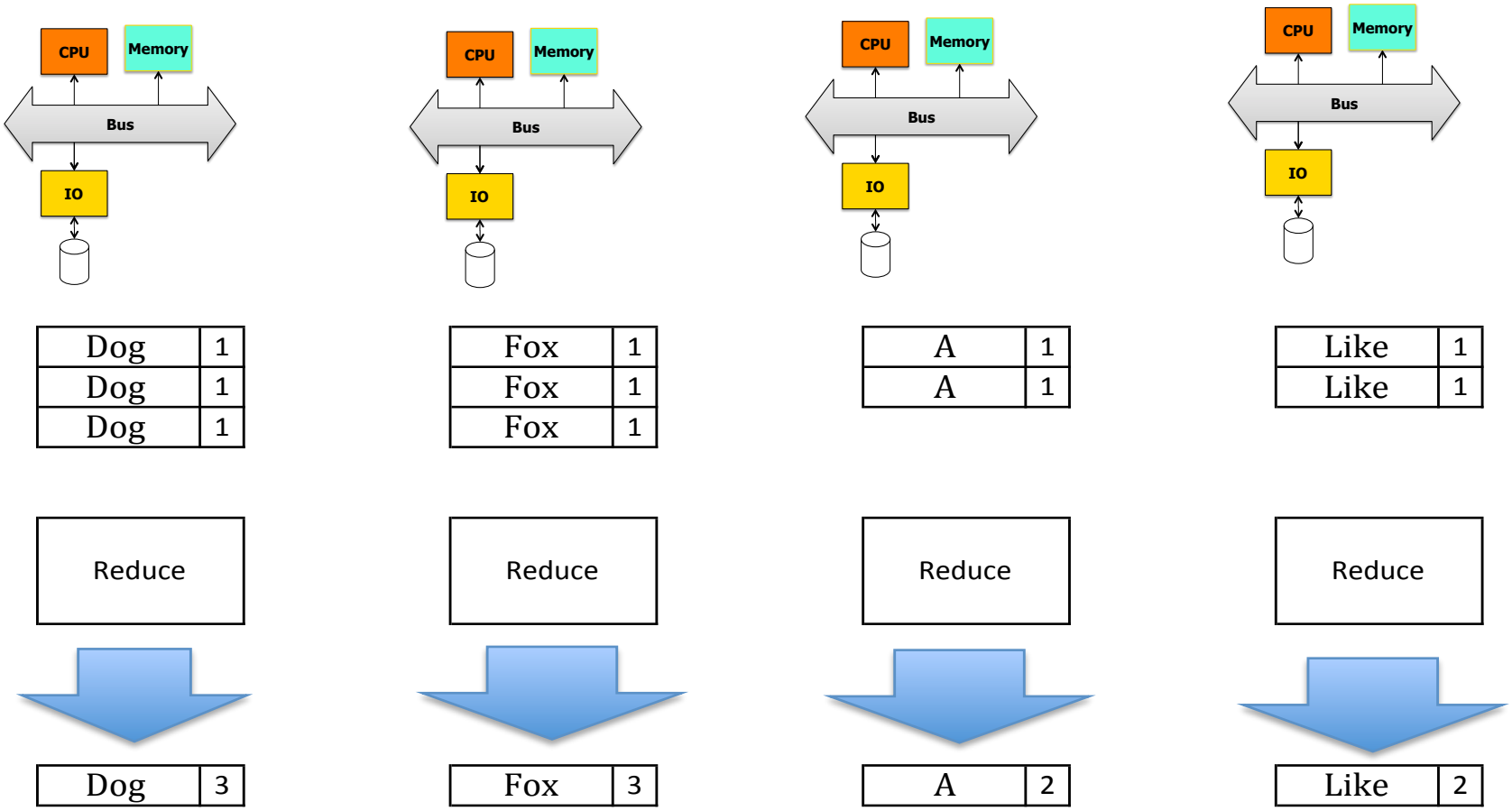


Reduce



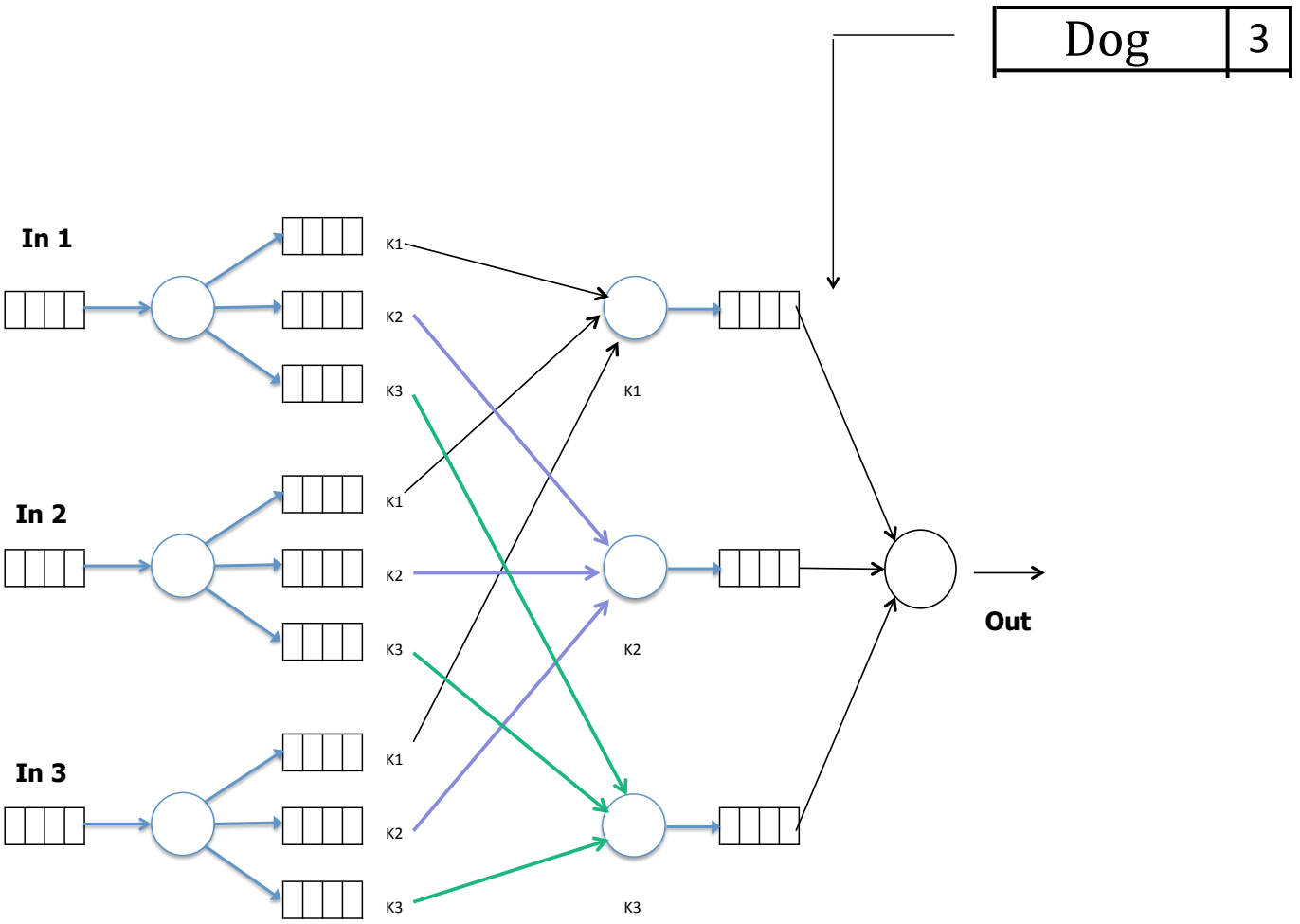
- Reduce tasks take many values for a single key output from map tasks and summarize them into a single output.

Reduce



- Each reducer operates on the result of a single key, so it can be parallelized by key.

Reduce

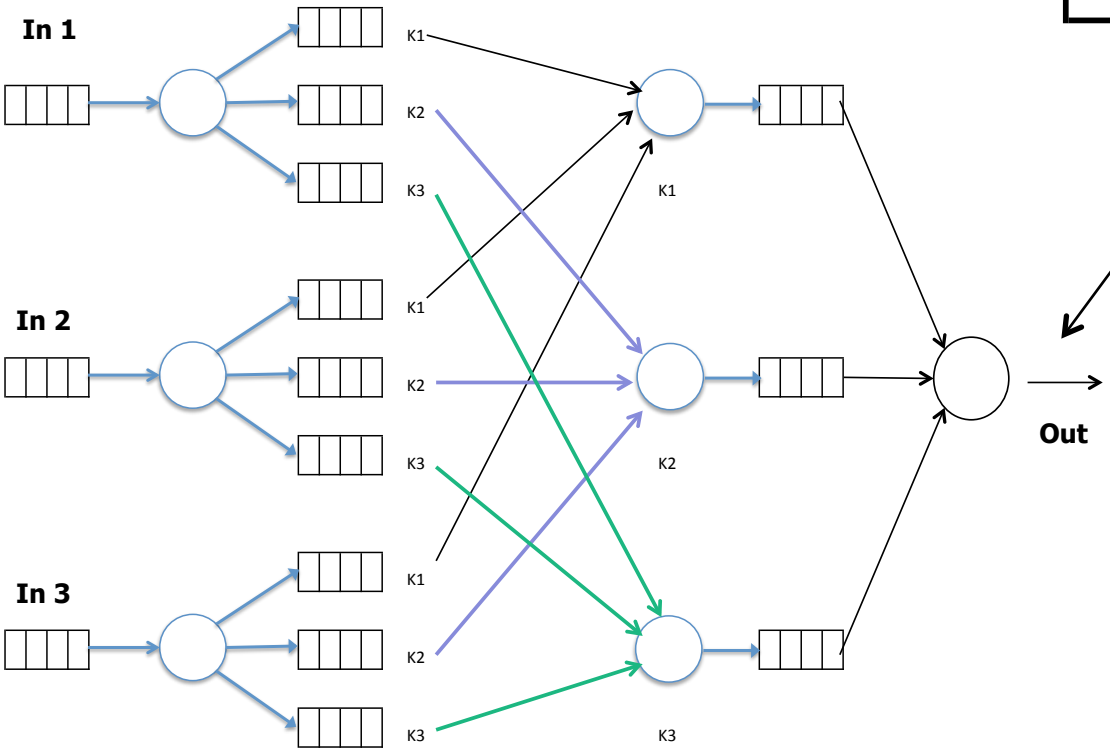


Output

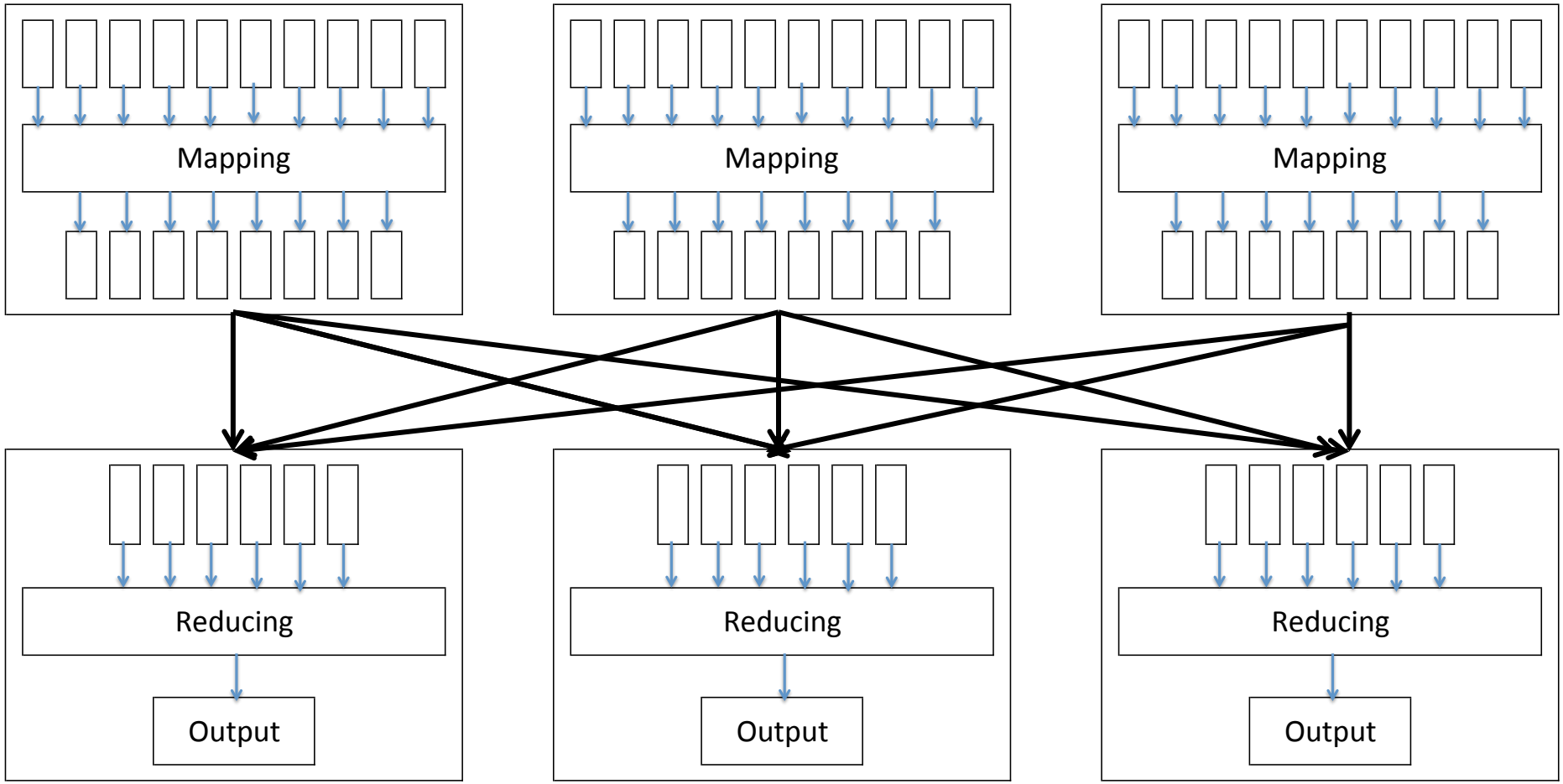
Dog	3
Fox	3
A	2
Like	2

Output

Dog	3
Fox	3
A	2
Like	2



Map Reduce Pipe Line



Open source version

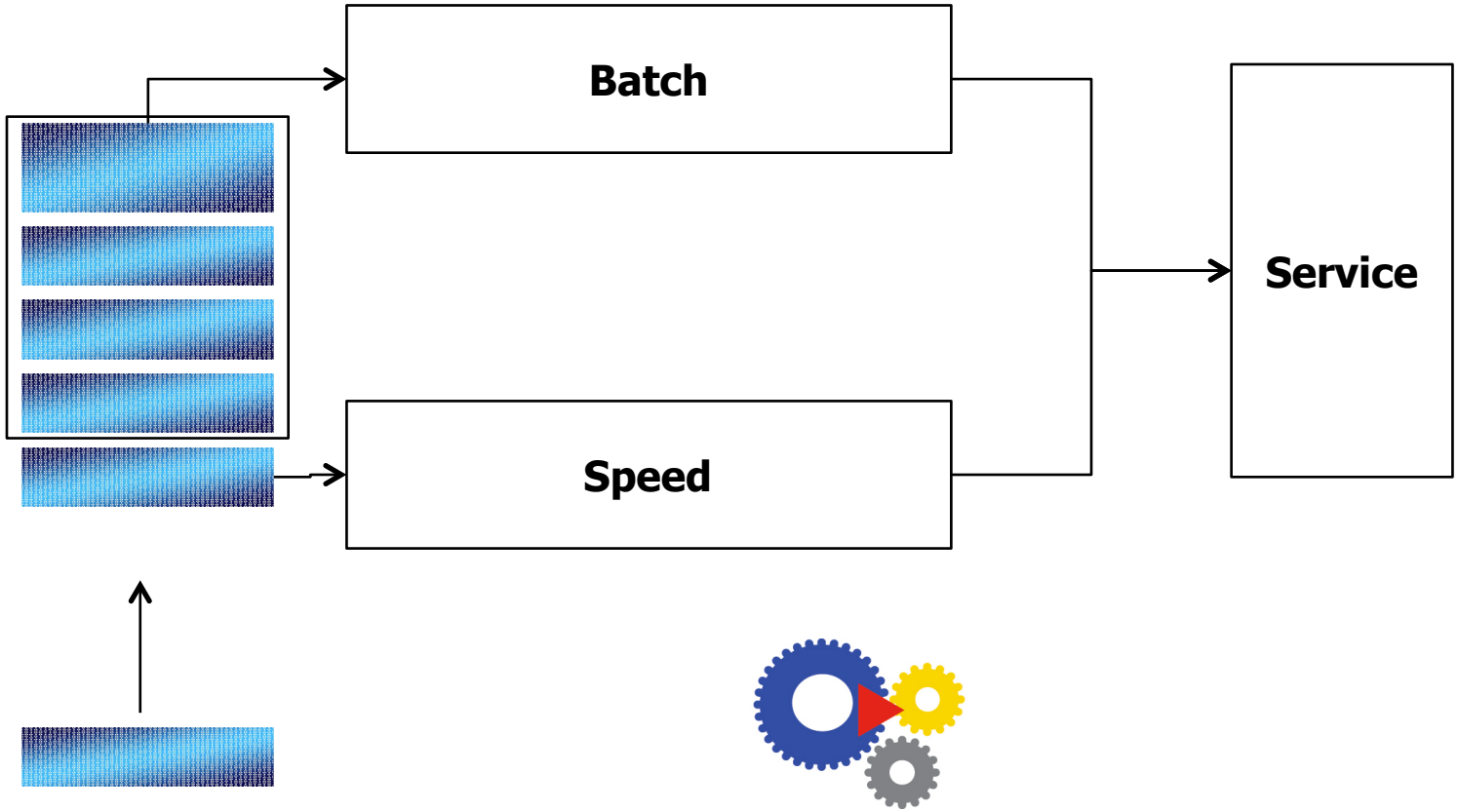
- http://hbase.apache.org/

- Apache HBase™ is the Hadoop database, a distributed, scalable, big data store.

Lambda Architecture

- Lambda architecture provides a combined solution of realtime (speed) data with batch data.
- The Mapreduce Latency for updates is up 2 to 3 hours.
- Intermediary updates are made during the Latency Time

Lambda Architecture



Lambda architecture

- 1) Speed/event layer :
  - Stream processing architecture.
  - The data are appended to the master data periodically over a time window.
  - Real time views
- 2) Batch layer :
  - Manages the master dataset:
    - Immutable, append-only raw data and pre-computing batch views.
  - Batch views
- 3) Serving layer:
  - Merges batch and real-time views.