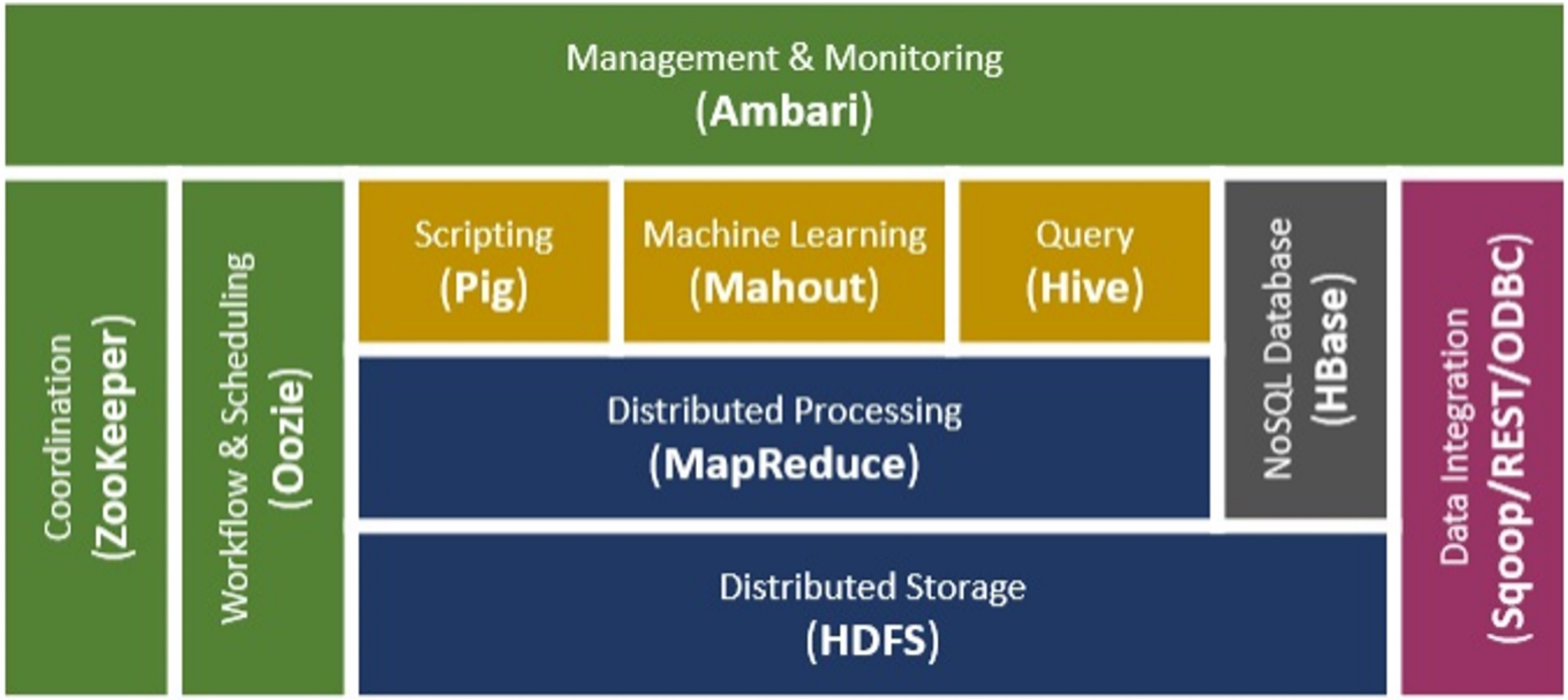
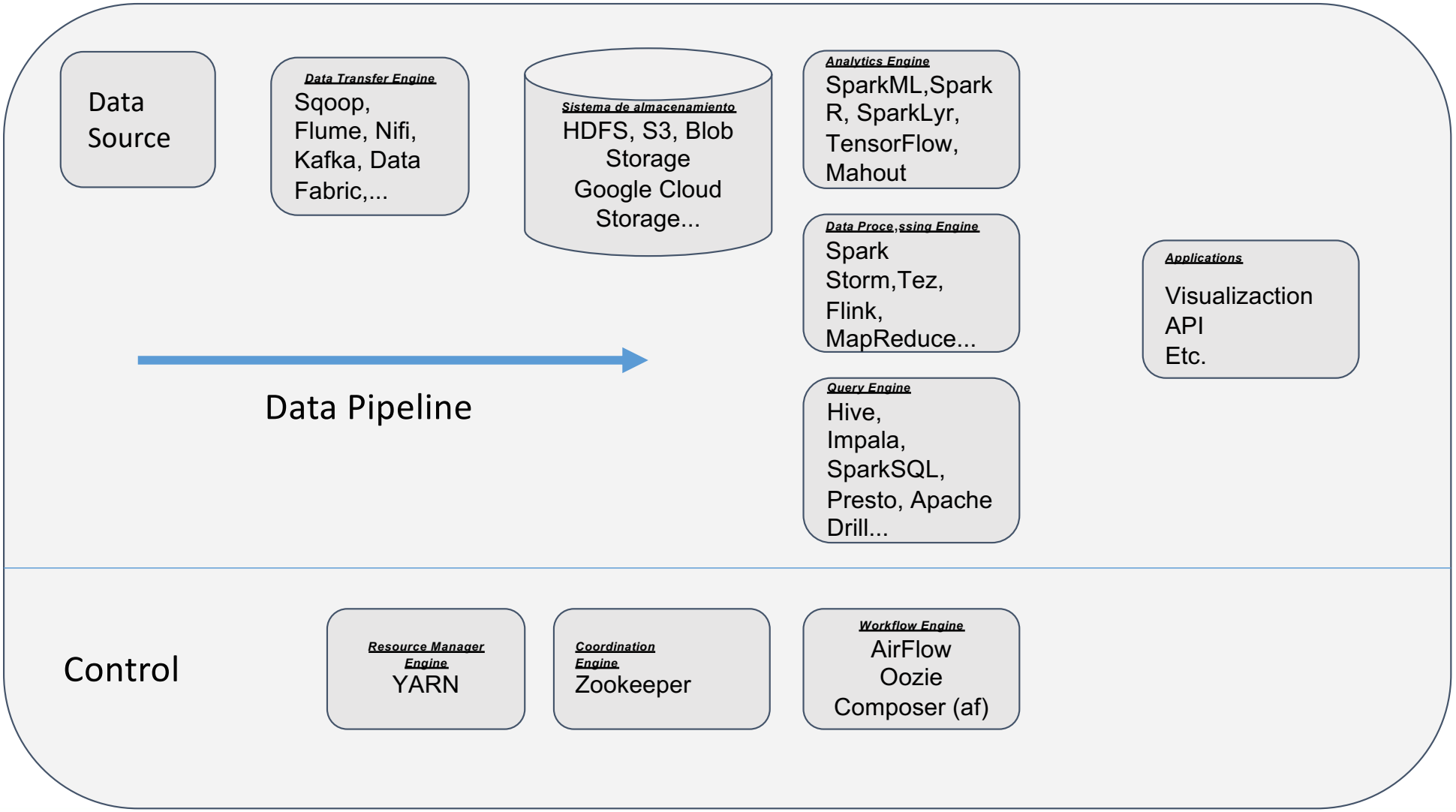
Procesamiento Distribuido

Hadoop MapReduce

Ecosistema Hadoop (estándar)



Ecosistema Big Data



Motores de procesamiento

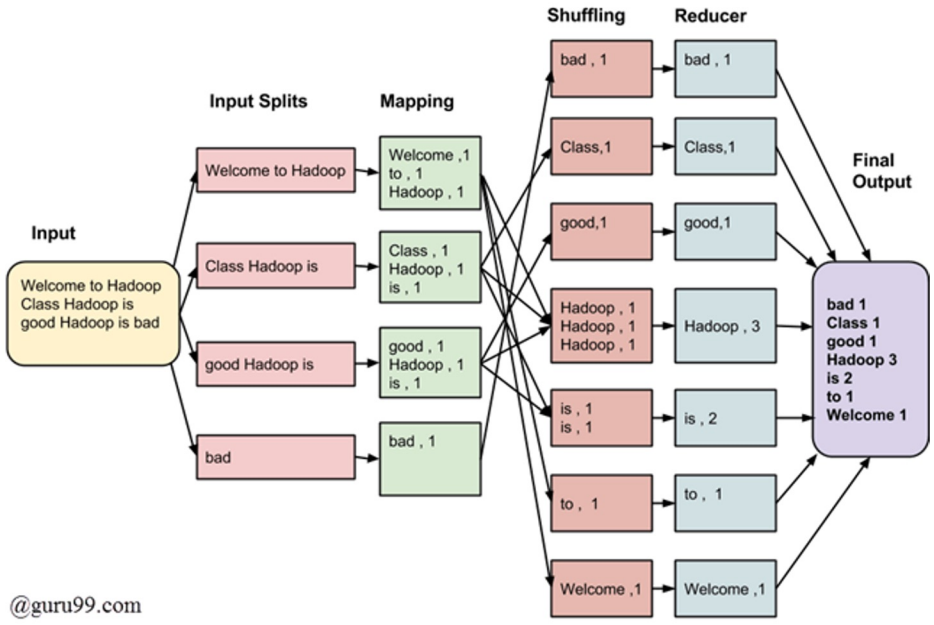
Paper base

- Jeffrey Dean and Sanjay Ghemawat. **MapReduce: Simplified Data Processing on Large Clusters.** jeff@google.com, sanjay@google.com. *Google, Inc.*

MapReduce: un marco para la programación paralela

- Minimiza el esfuerzo del programador para tareas simples de procesamiento en paralelo
- Funciones
  - Ocultar muchos detalles de bajo nivel (red, almacenamiento)
  - La tolerancia de fallas incorporada
  - Balanceo de carga automático

MapReduce



El cliente envía un tarea al JobTracker

El JobTraker interroga al NameNode sobre la ubicación de los datos.

Según la respuesta del NameNode el JobTracker solicita a los respectivos Task Trackers para que ejecuten una tarea en sus datos.

Los resultados son guardados en los DataNode y el NameNode es informado.

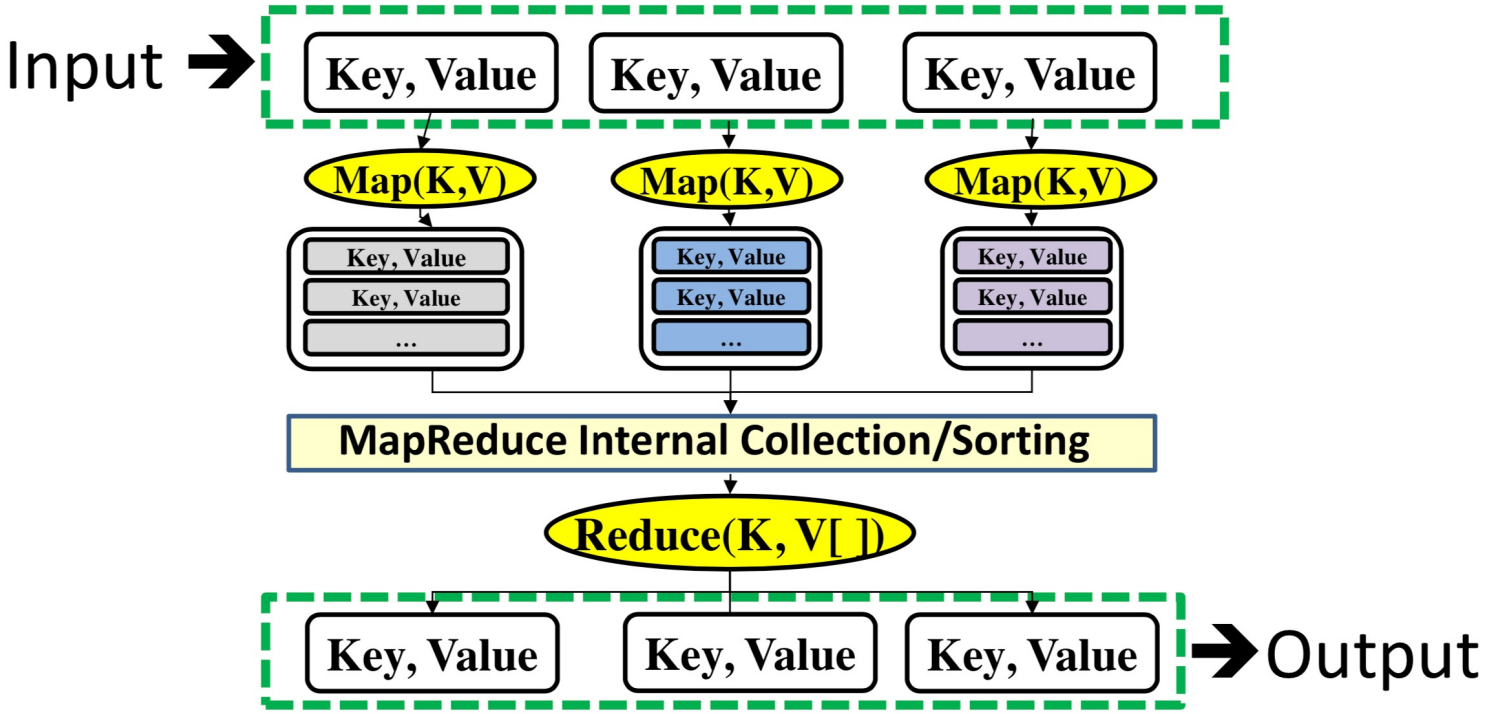
Base funcional

- Mapper
  - Los programas MapReduce se dividen en Mappers, tareas que se ejecutan en los nodos
  - Cada tarea Map ataca a un solo bloque de datos HDFS
  - Se ejecuta en el nodo donde reside el bloque
- Shuffle y Sort
  - Ordena y consolida los datos intermedios (temporales, k2) que generan los mappers
  - Se lanza después de que todos los mappers hayan terminado y antes de que lancen los procesos Reducer
- Reducer
  - Opera sobre los datos intermedios Shuffle/Sort
  - Genera la salida final

Base funcional

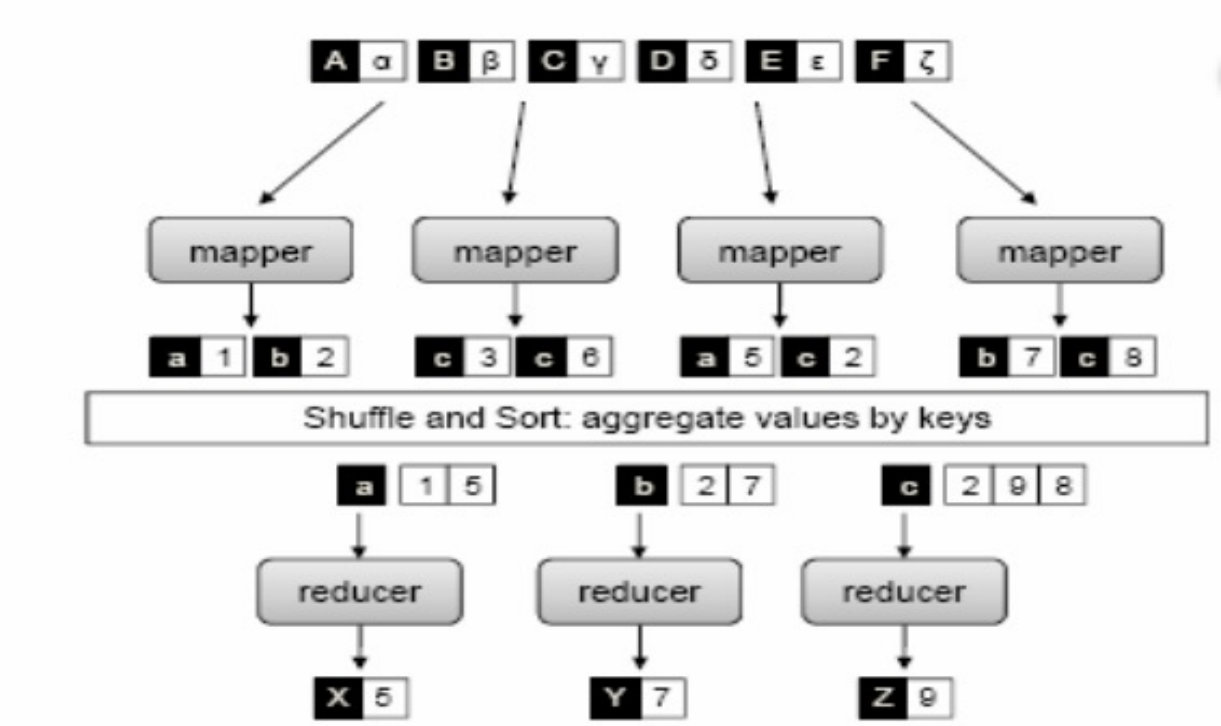
- Tuplas (clave,valor)
- Mapeo: Map
  - Toma Map(k1,v1) -> list(k2,v2)
    - Transforma dominio de datos
  - La lista (k2,v2) es agrupada por k2 asi:
    - (k2,[v2])
  - Es paralelizable
- Reducción: Reduce
  - Se aplica en paralelo por grupos
  - Reduce(k2, list(v2)) -> list(v3)
    - list(k1,v1) -> list(v3)

MapReduce: Computation Pipeline



Slide adapted from Alexander Behm & Ajey Shah's presentation (http://www.slideshare.net/gothicane/behm-shah-pagerank)

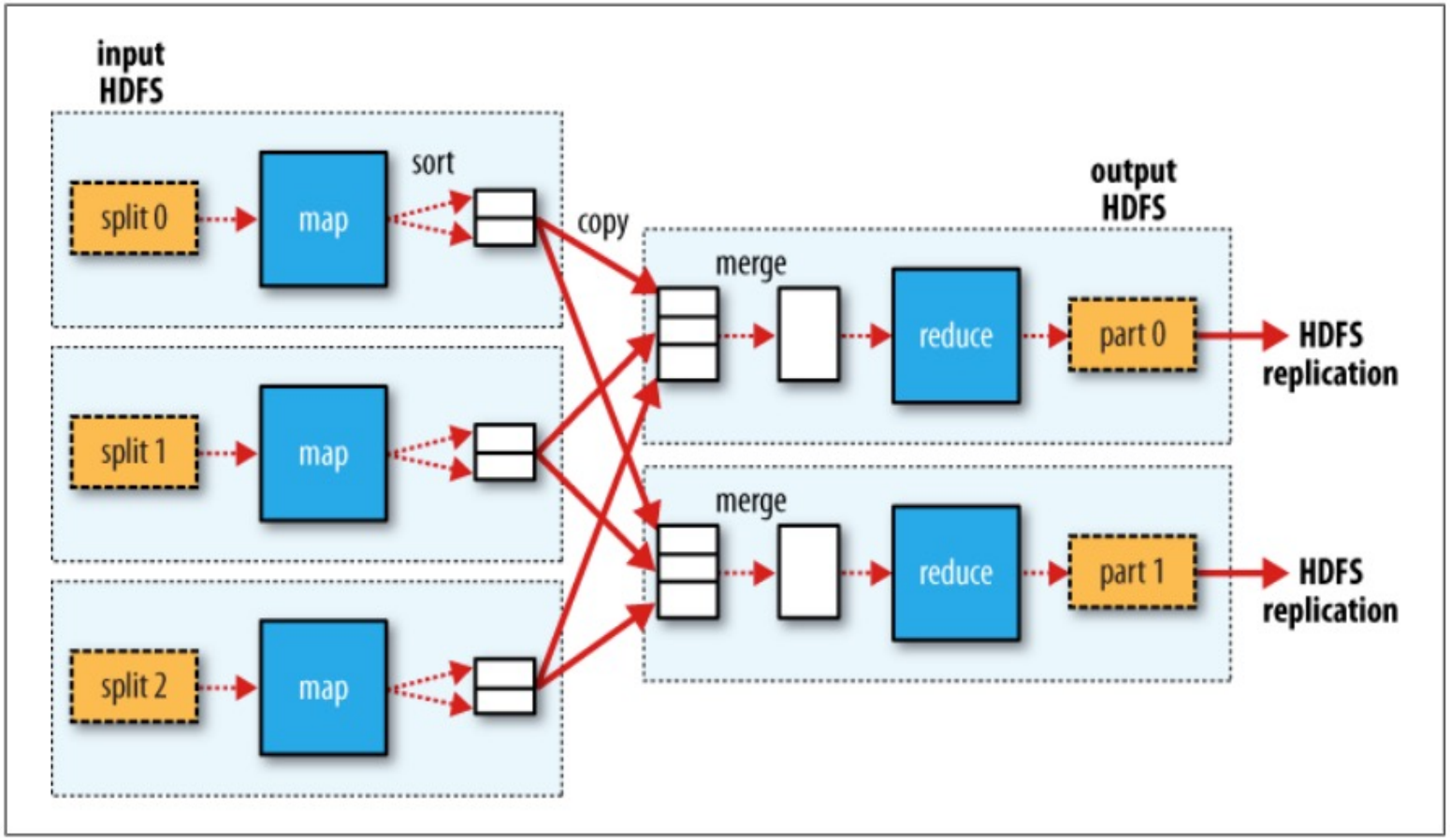
Visión lógica



MapReduce a través de un juego

https://www.youtube.com/watch?v=bcjSe0xCHbE

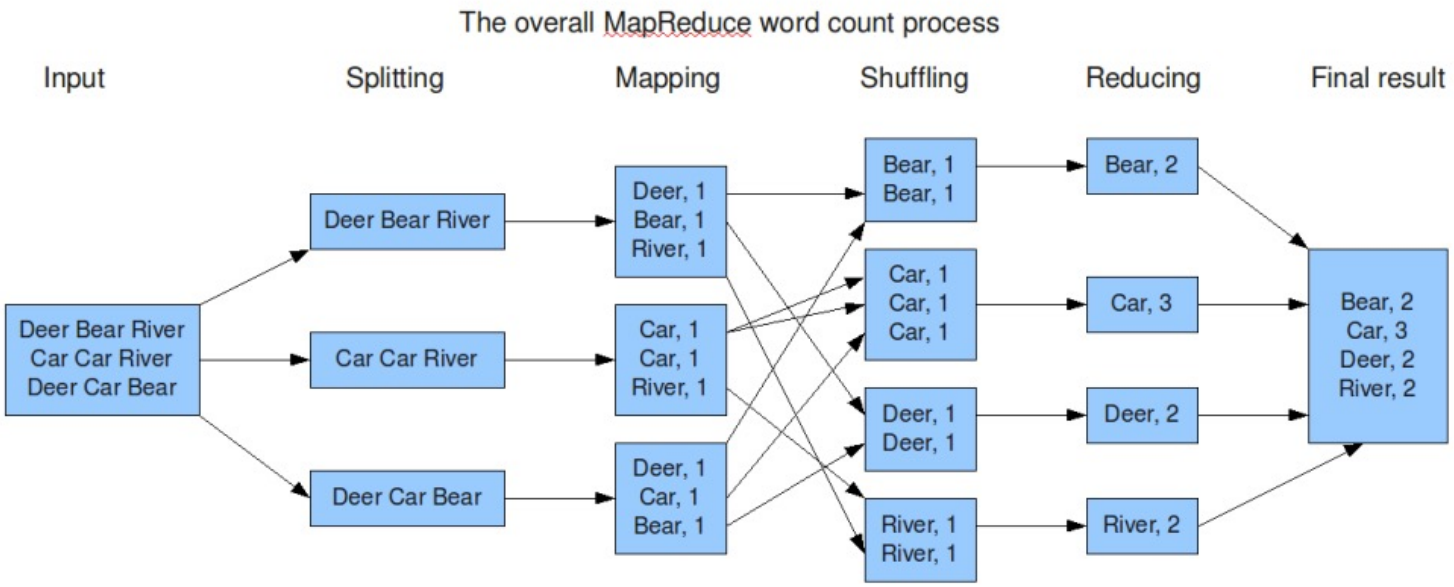
Map/Reduce


input
HDFS
split 0
map
sort
split 1
map
split 2
map
copy
merge
reduce
part 0
output
HDFS
HDFS
replication
merge
reduce
part 1
HDFS
replication

Ejemplo: Conteo de palabras

```
Map(String name, String content):
// name: nombre del documento
// content: contenido del documento
        for each word w in content:
                emit(w,1)
```

```
Reduce(String word, List V):
// word: una palabra
// List V: lista de valores que se emitieron en el Map
        int result = 0
        for each v1 in V:
                result = result + v1
        emit(word,result)
```


The overall MapReduce word count process
Input
Splitting
Mapping
Shuffling
Reducing
Final result
Deer Bear River
Car Car River
Deer Car Bear
Deer Bear River
Car Car River
Deer Car Bear
Deer, 1
Bear, 1
River, 1
Car, 1
Car, 1
River, 1
Deer, 1
Car, 1
Bear, 1
Bear, 1
Bear, 1
Car, 1
Car, 1
Car, 1
Deer, 1
Deer, 1
River, 1
River, 1
Bear, 2
Car, 3
Deer, 2
River, 2
Bear, 2
Car, 3
Deer, 2
River, 2

Word Counting

Input: Text Data

Hello World Bye World
Hello Hadoop Bye Hadoop
Bye Hadoop Hello Hadoop
… …

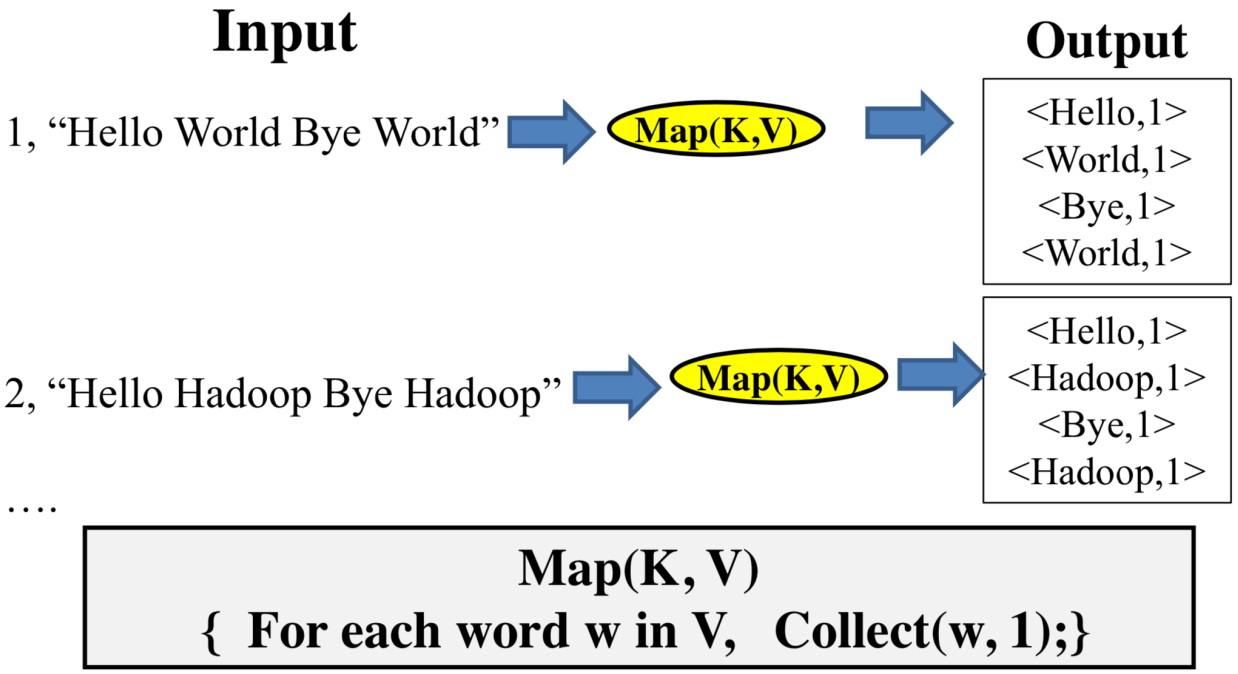
Output:
Count of each word

Bye 3
Hadoop 4
Hello 3
World 2
…

How can we do this within the MapReduce framework?

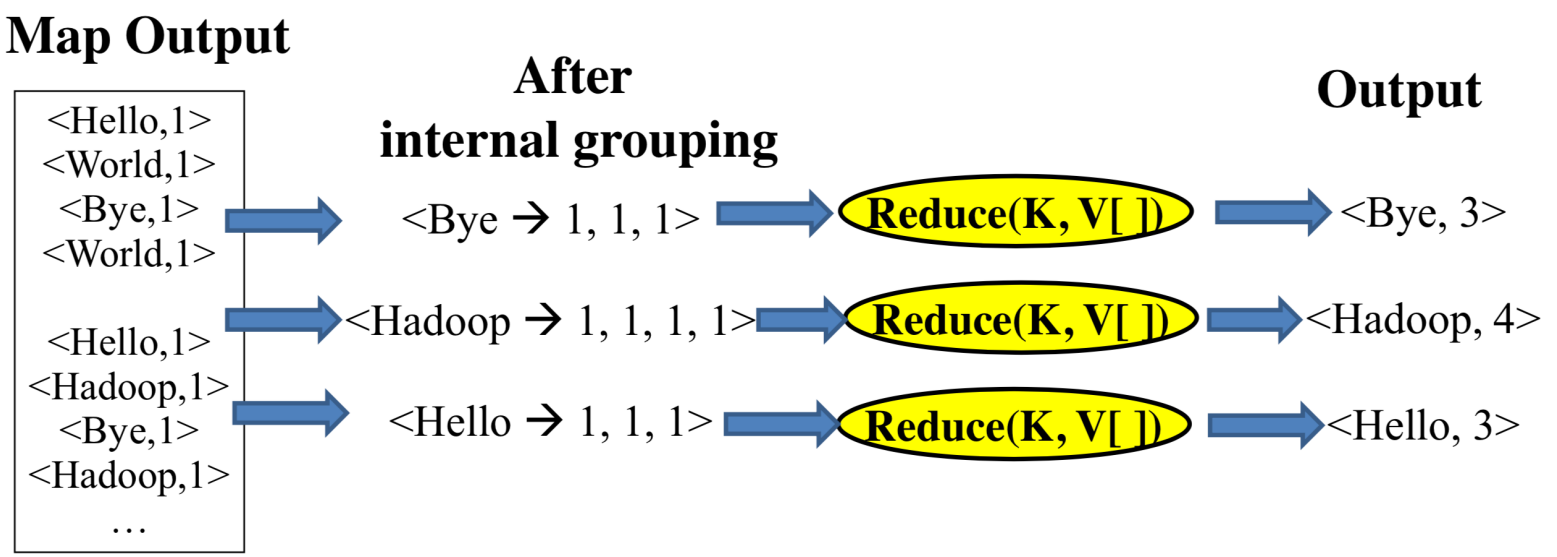
Slide adapted from Alexander Behm & Ajey Shah's presentation (http://www.slideshare.net/gothicane/behm-shah-pagerank)

Word Counting: Map Function



Slide adapted from Alexander Behm & Ajey Shah's presentation (http://www.slideshare.net/gothicane/behm-shah-pagerank)

Word Counting: Reduce Function



Slide adapted from Alexander Behm & Ajey Shah's presentation (http://www.slideshare.net/gothicane/behm-shah-pagerank)