Universidad del Rosario

Big Data

W2. Introducción al procesamiento en paralelo MapReduce

FERNEY ALBERTO BELTRAN MOLINA
Escuela de Ingeniería, Ciencia y Tecnología
Matemáticas Aplicadas y Ciencias de la Computación

Contenidos

- **Introducción a MapReduce**
- Patrones de programación MapReduce

Chapter 6. Erl, T. (2013). Big Data Fundamentals: Concepts, Drivers & Techniques. Prentice Hall.Big

THE WORD-COUNT PROBLEM

El problema de Word-Count es la tarea de contar el número de ocurrencias de cada palabra en un documento

“It was the best of times, it was the worst of times, it was the age of wisdom, it was the age of foolishness, it was the epoch of belief, it was the epoch of incredulity, it was the season of Light, it was the season of Darkness, it was the spring of hope, it was the winter of despair, we had everything before us, we had nothing before us, we were all going direct to Heaven, we were all going direct the other way—in short, the period was so far like the present period, that some of its noisiest authorities insisted on its being received, for good or for evil, in the superlative degree of comparison only.

There were a king with large jaw and a queen with a plain face…”

Charles Dickens, A Tale of Two Cities

THE WORD-COUNT PROBLEM

El problema de Word-Count es la tarea de contar el número de ocurrencias de cada palabra en un documento

El desafío, crear un programa de computador donde :

- La entrada, sea un documento de texto
- La salida, lista de pares (palabra, cuenta)

Usaremos el problema de Word-Count para ilustrar nuestro primer método Big Data: MapReduce.

El problema de Word-Count no es particularmente interesante, pero es posible que sea el mejor ejemplo disponible. Úselo para comprender MapReduce y reflexionar sobre Big Data.

THE WORD-COUNT PROBLEM

El problema de Word-Count es la tarea de contar el número de ocurrencias de cada palabra en un documento

*"**It was the best of times, it was the worst of times, it was the age of wisdom, it was the age of foolishness, it was the epoch of belief, it was the epoch of incredulity***, it was the season of Light, it was the season of Darkness, it was the spring of hope, it was the winter *of despair, we had everything before us, we had nothing before us, we were all going direct to Heaven, we were all going direct the other way—in short, the period was so far like the present period, that some of its noisiest authorities insisted on its being received, for good or for evil, in the superlative degree of comparison only.*

There were a king with large jaw and a queen with a plain face…"

Charles Dickens, A Tale of Two Cities

THE WORD-COUNT PROBLEM

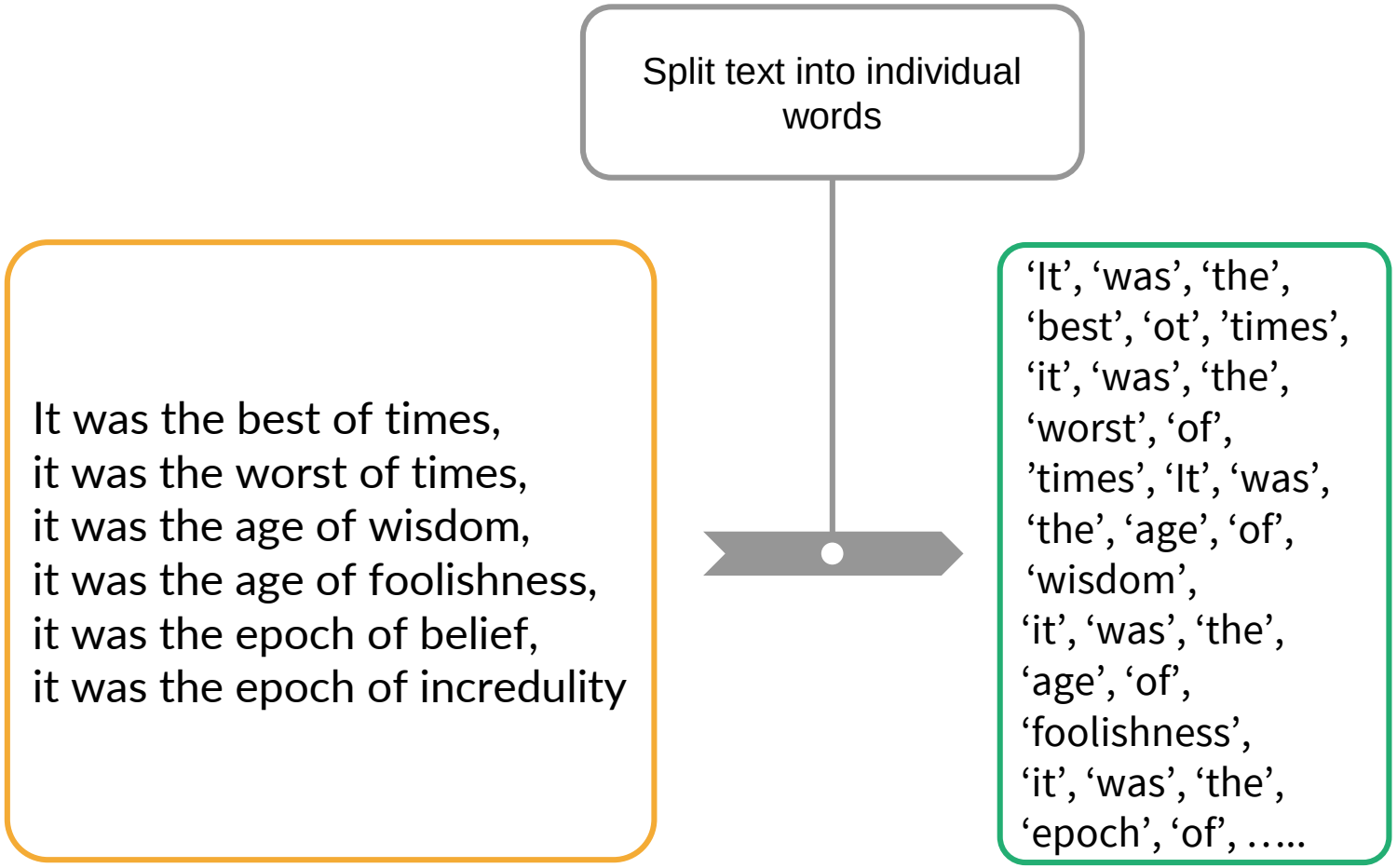
It was the best of times,
it was the worst of times,
it was the age of wisdom,
it was the age of foolishness,
it was the epoch of belief,
it was the epoch of incredulity

Mecanismo para contra palabras

(it, 6)
(was, 6)
(the, 6)
(best, 1)
(of, 6)
(times, 2)
(worst, 1)
(age, 2)
(epoch, 2)
(foolishness, 1)
(incredulity, 1)

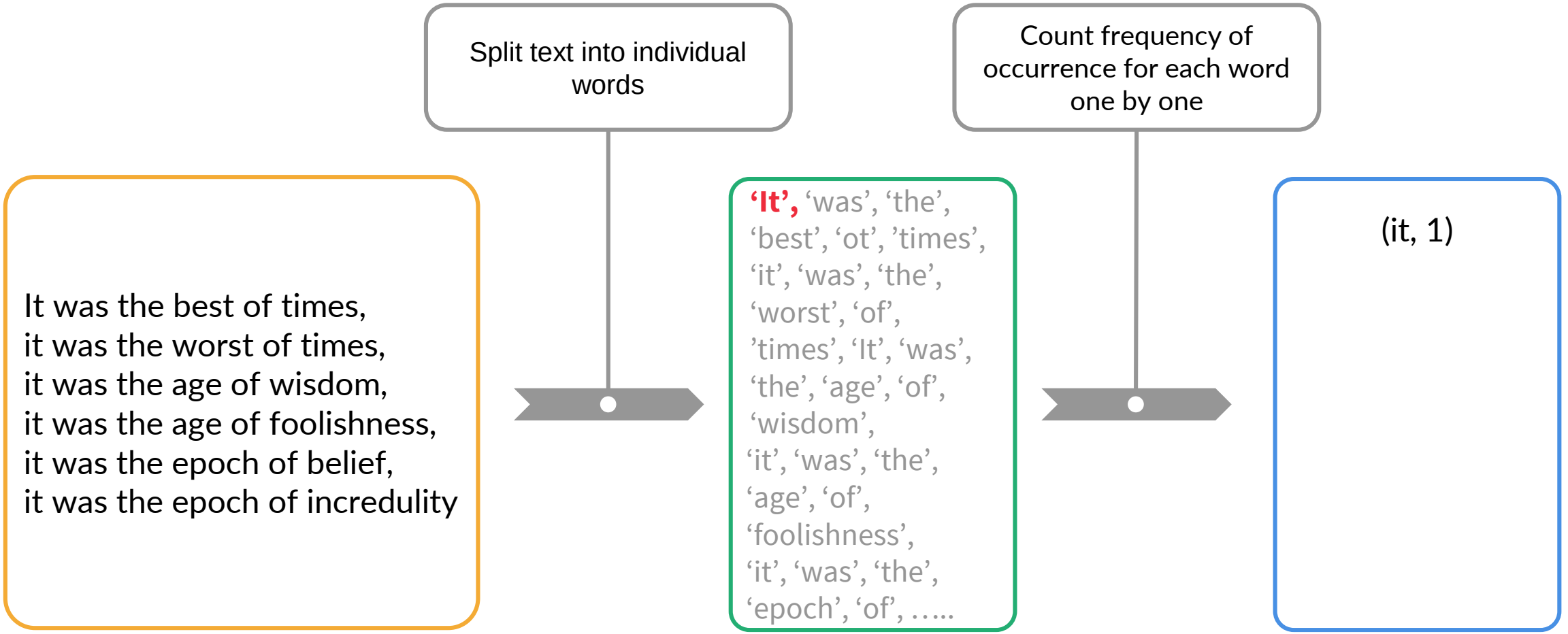
- **Input**: documento de texto
- **Output**: lista de pares (word, count)

Solución



- **Input**: documento de texto

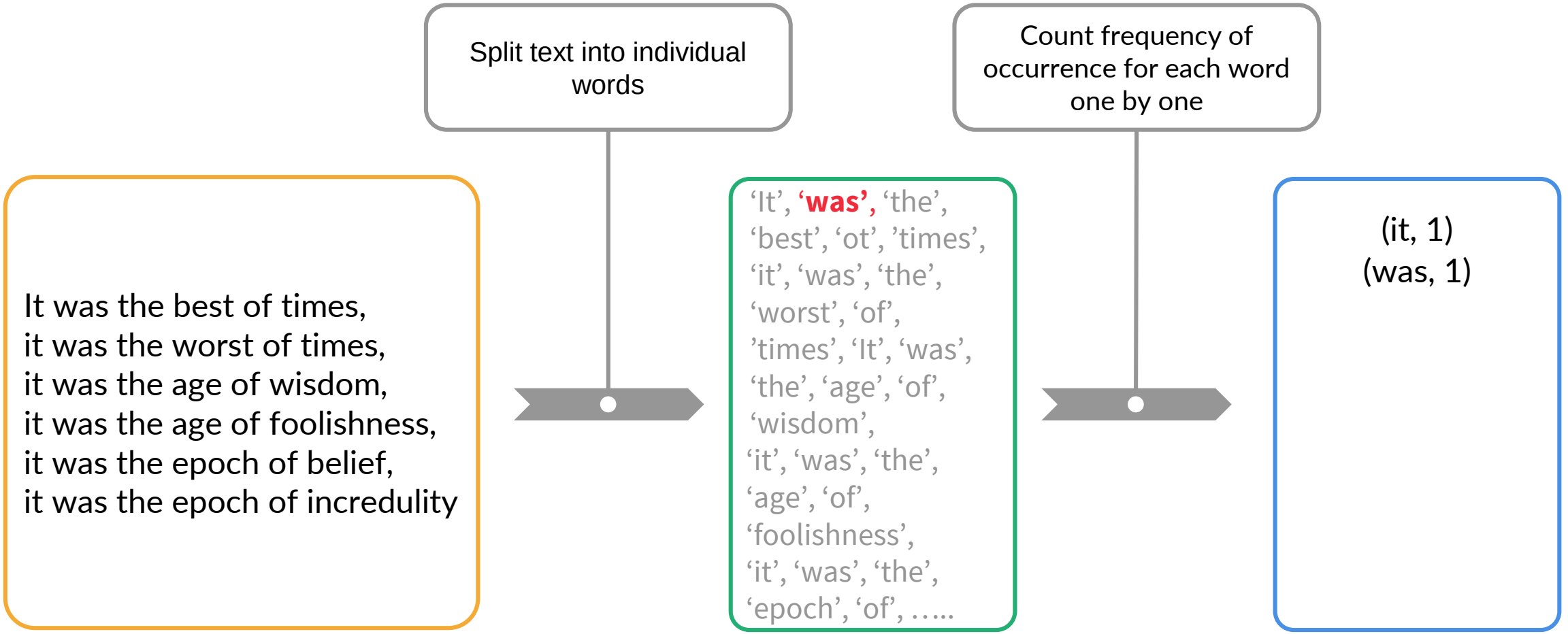
Solución



- **Input**: documento de texto

- **Output**:
list of pairs (word, count)

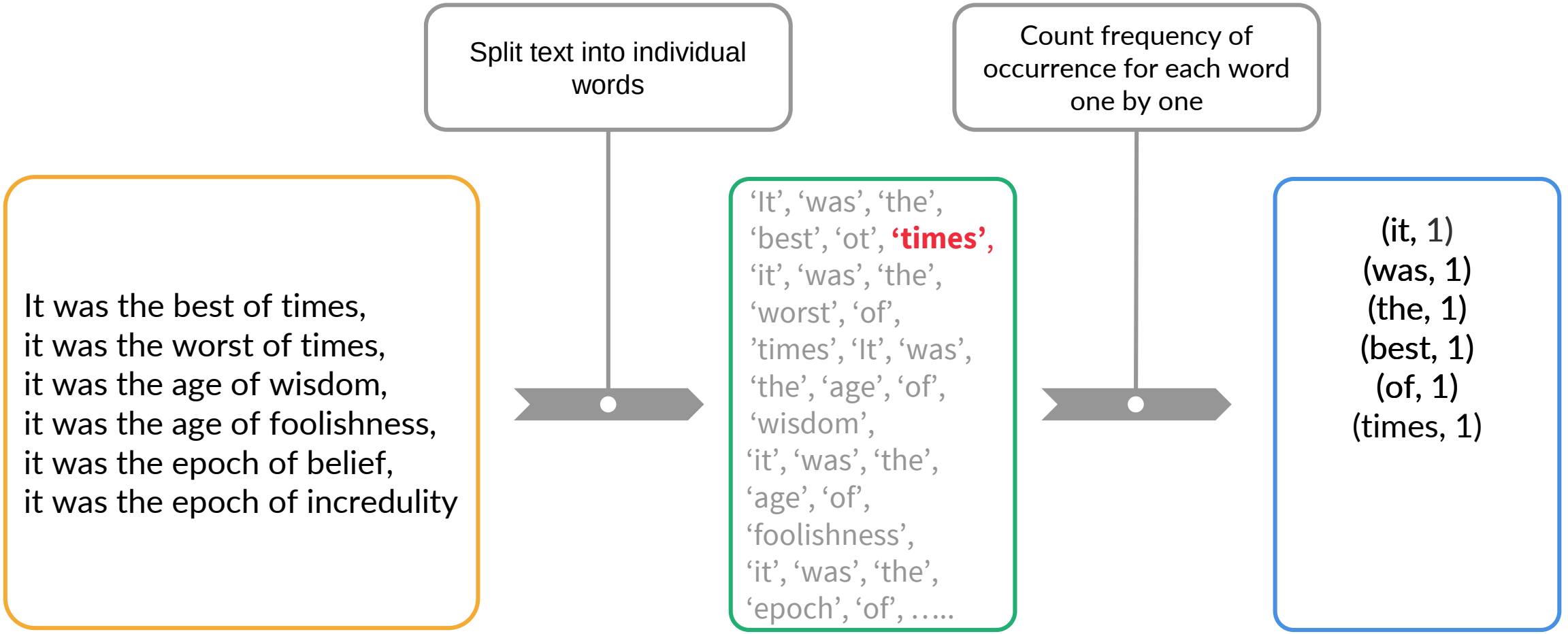
Solución



- **Input**: documento de texto

- **Output**:
list of pairs (word, count)

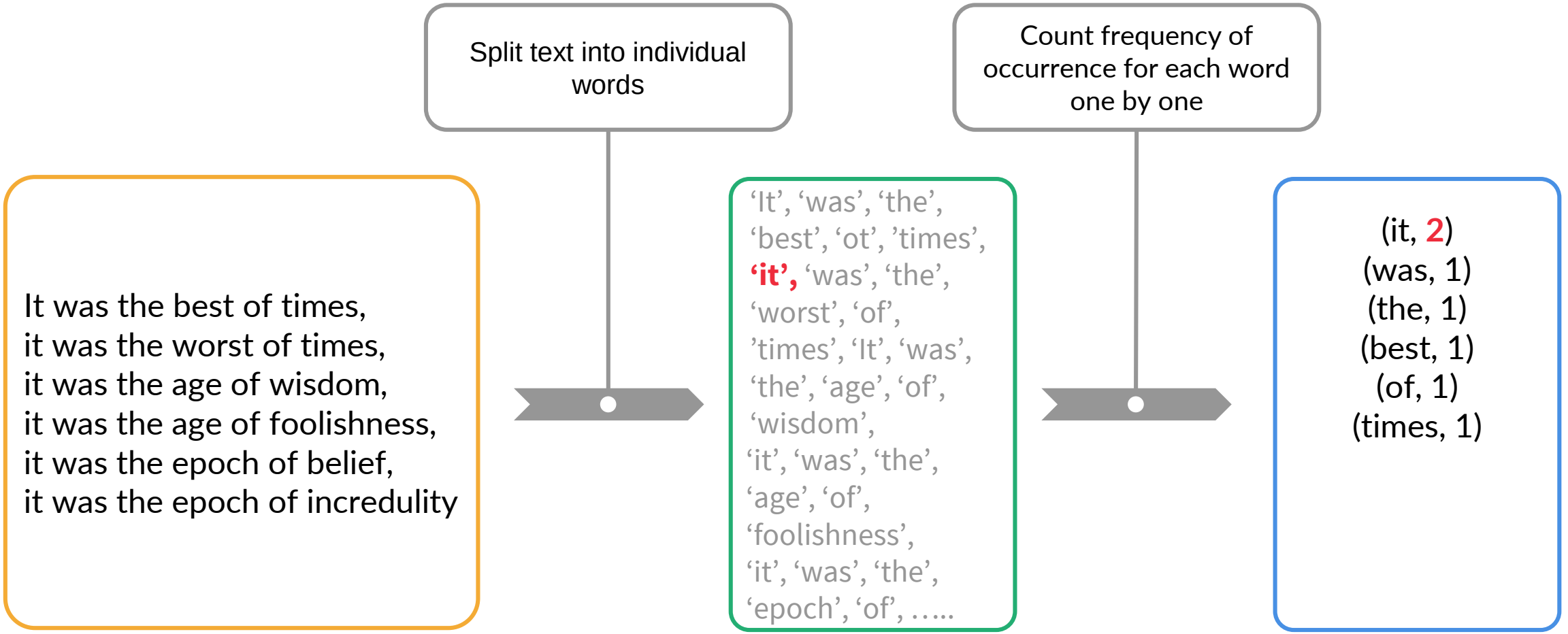
Solución



- **Input**: documento de texto

- **Output**:
list of pairs (word, count)

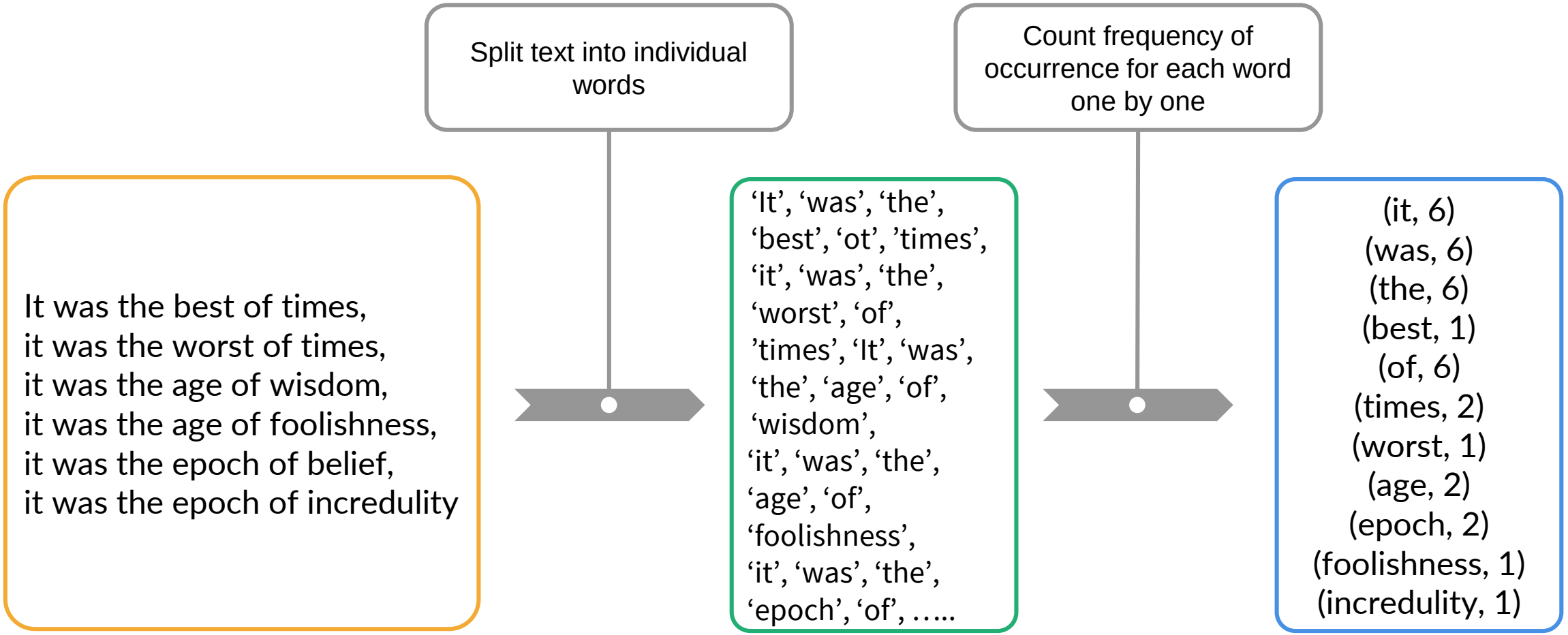
Solución



- **Input:** documento de texto

- **Output**:
list of pairs (word, count)

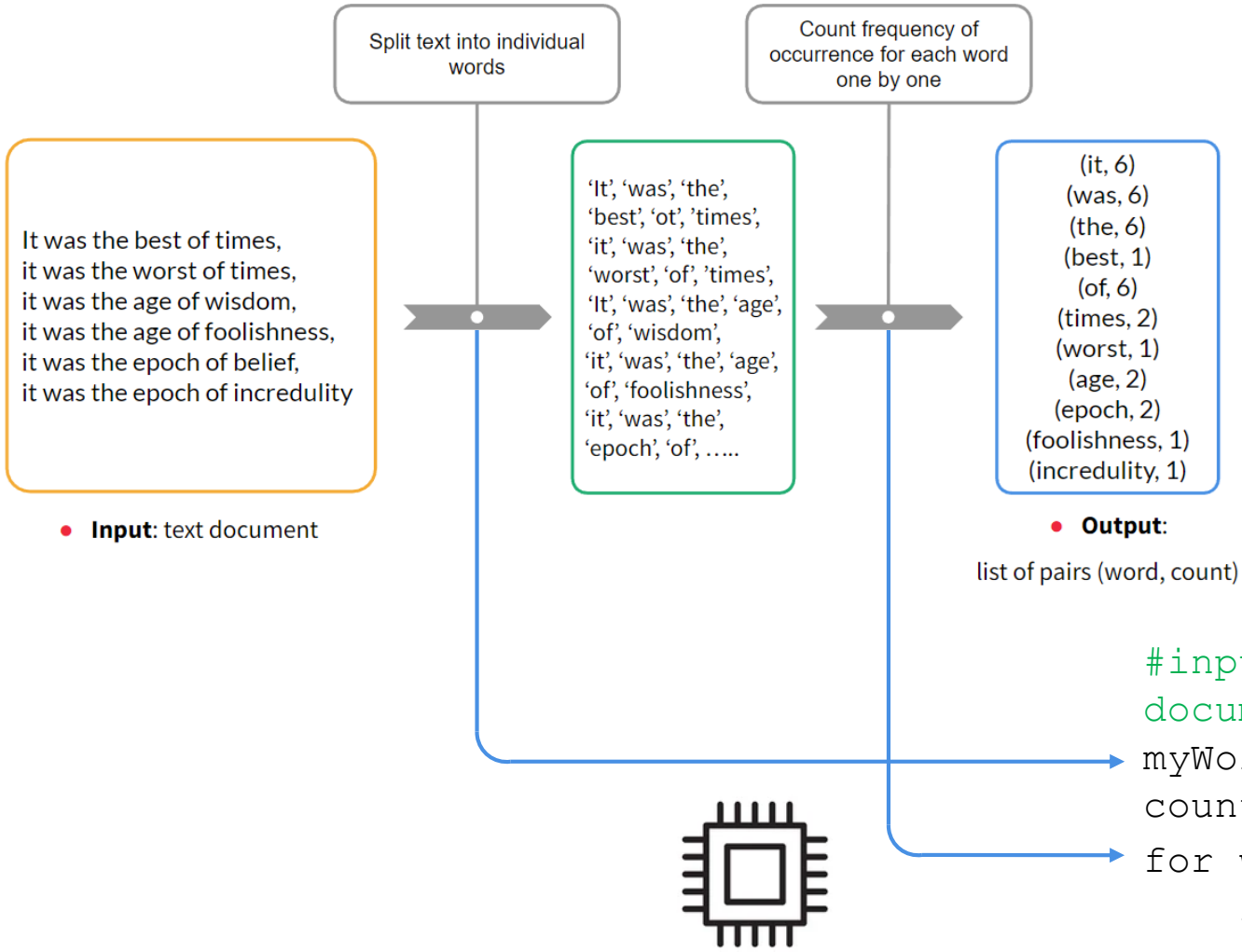
Solución



- **Input:** documento de texto

- **Output**:
list of pairs (word, count)

Solución en un único procesador



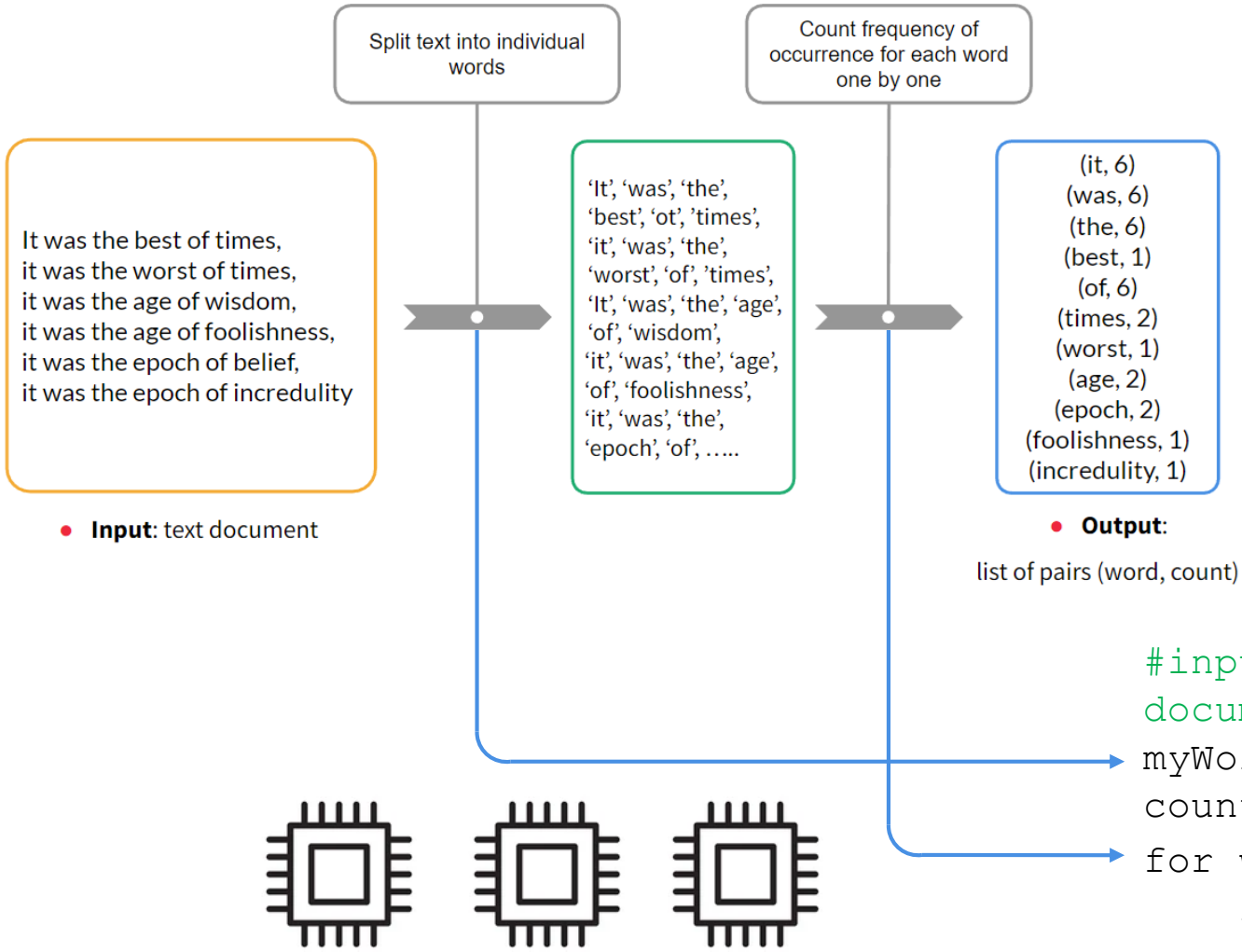
- **Input**: text document

- **Output**:

list of pairs (word, count)

```
#input: 'text' string with the complete
document
myWords = text.split()
count = dict()
for word in myWords:
    if word in count:
        count[word] = count[word] + 1
    else:
        count[word] = 1
```

Escalar la solución



- **Input**: text document
- **Output**:

list of pairs (word, count)

```
#input: 'text' string with the complete
document
myWords = text.split()
count = dict()
for word in myWords:
    if word in count:
        count[word] = count[word] + 1
    else:
        count[word] = 1
```

Pueden más núcleos de procesamiento aumentar la velocidad del proceso?

Paralelizar la solución

La solución en **un procesador** realizo dos tareas básicas:

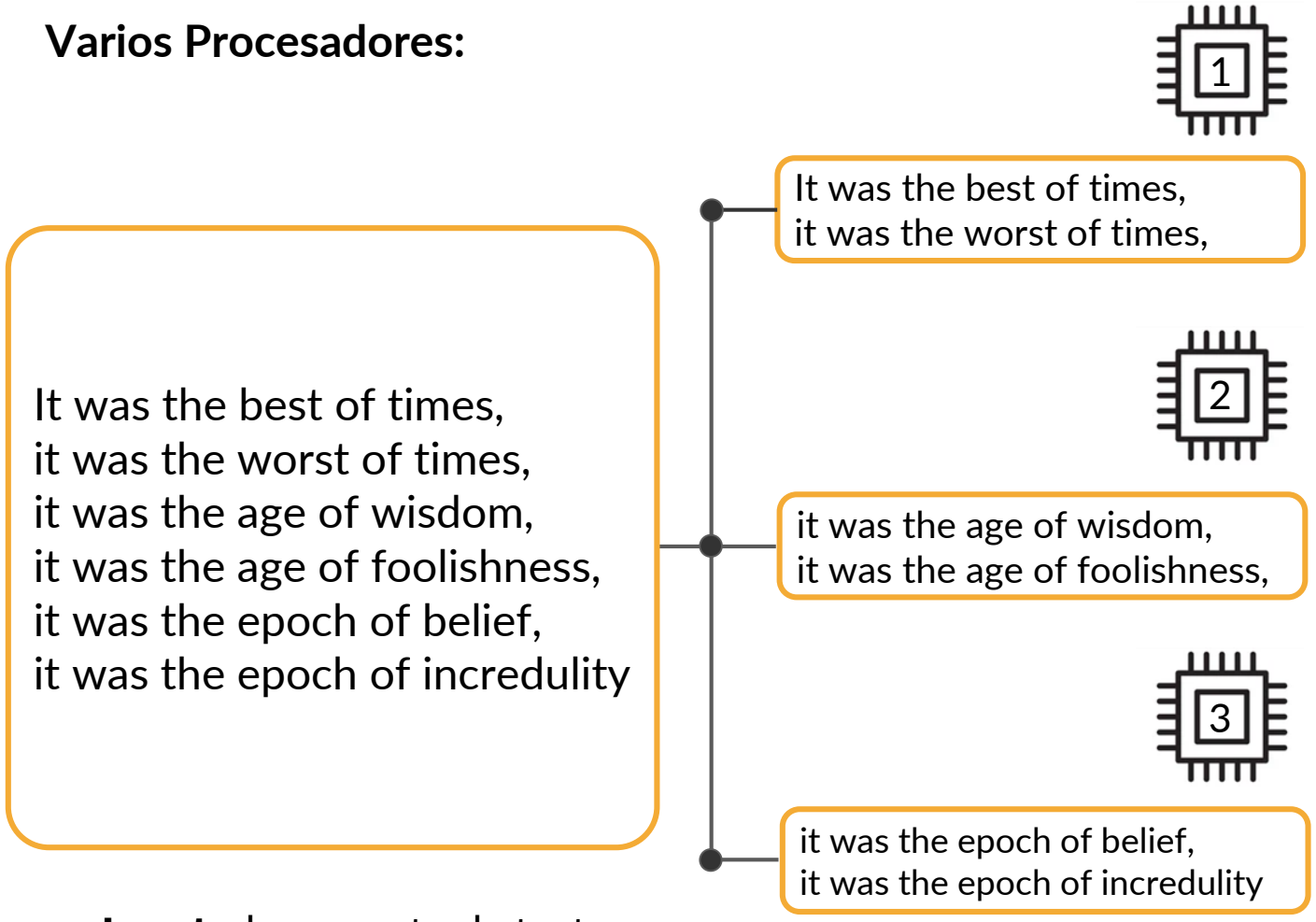
- Dividir el texto de entrada en una lista de palabras
- Leer cada palabra y aumentar el conteo respectivo

Si tenemos **varios procesadores** podemos paralelizar la solución de la siguiente manera:

- Asigne un **fragmento parcial** del texto de entrada a cada procesador
- Cada procesador resuelve el problema de Word-Count para su **propio fragmento** de texto
- Los resultados de cada procesador se **fusionan**.

Paralelizar la solución

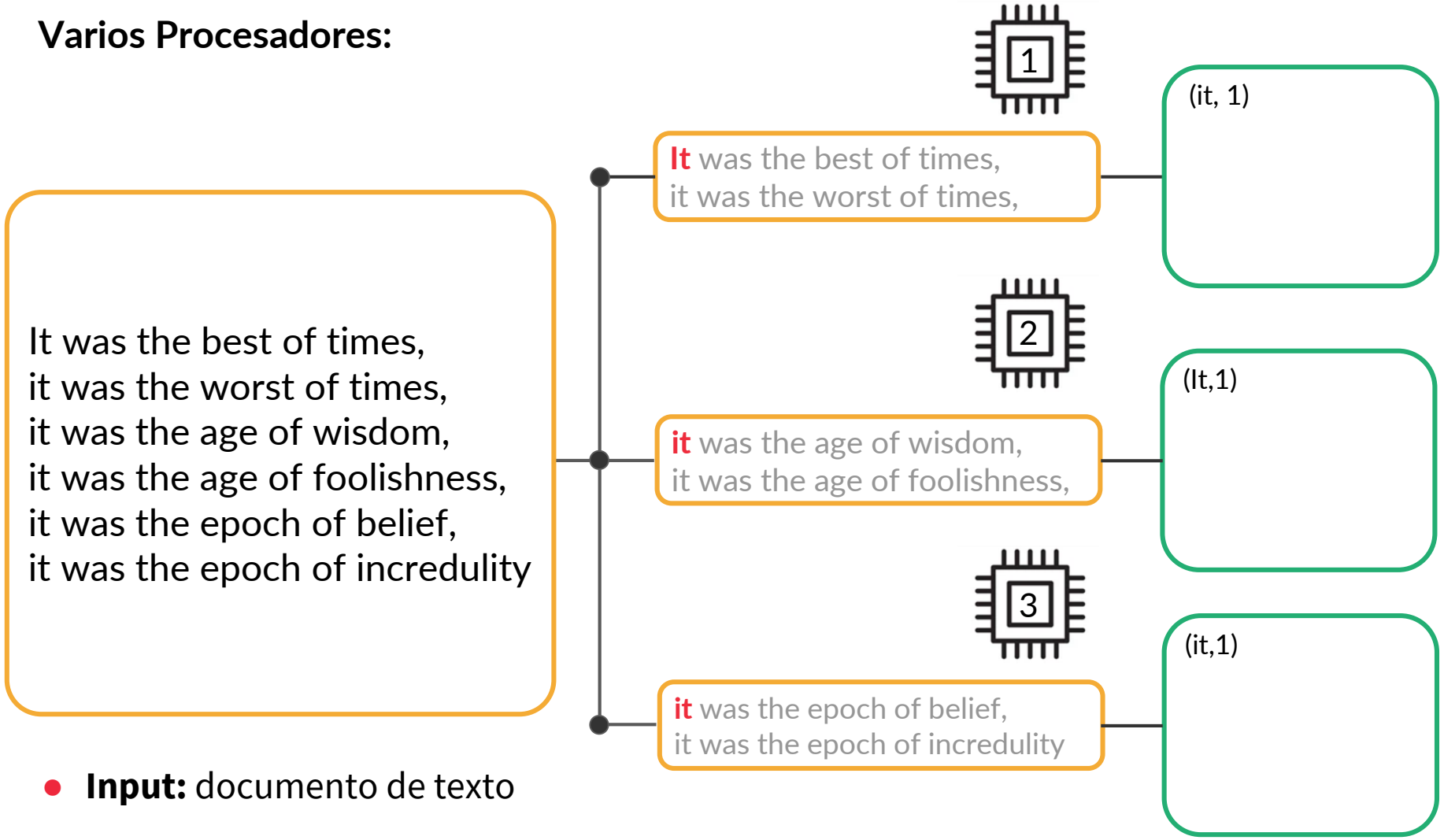
Varios Procesadores:



- **Input:** documento de texto

Paralelizar la solución

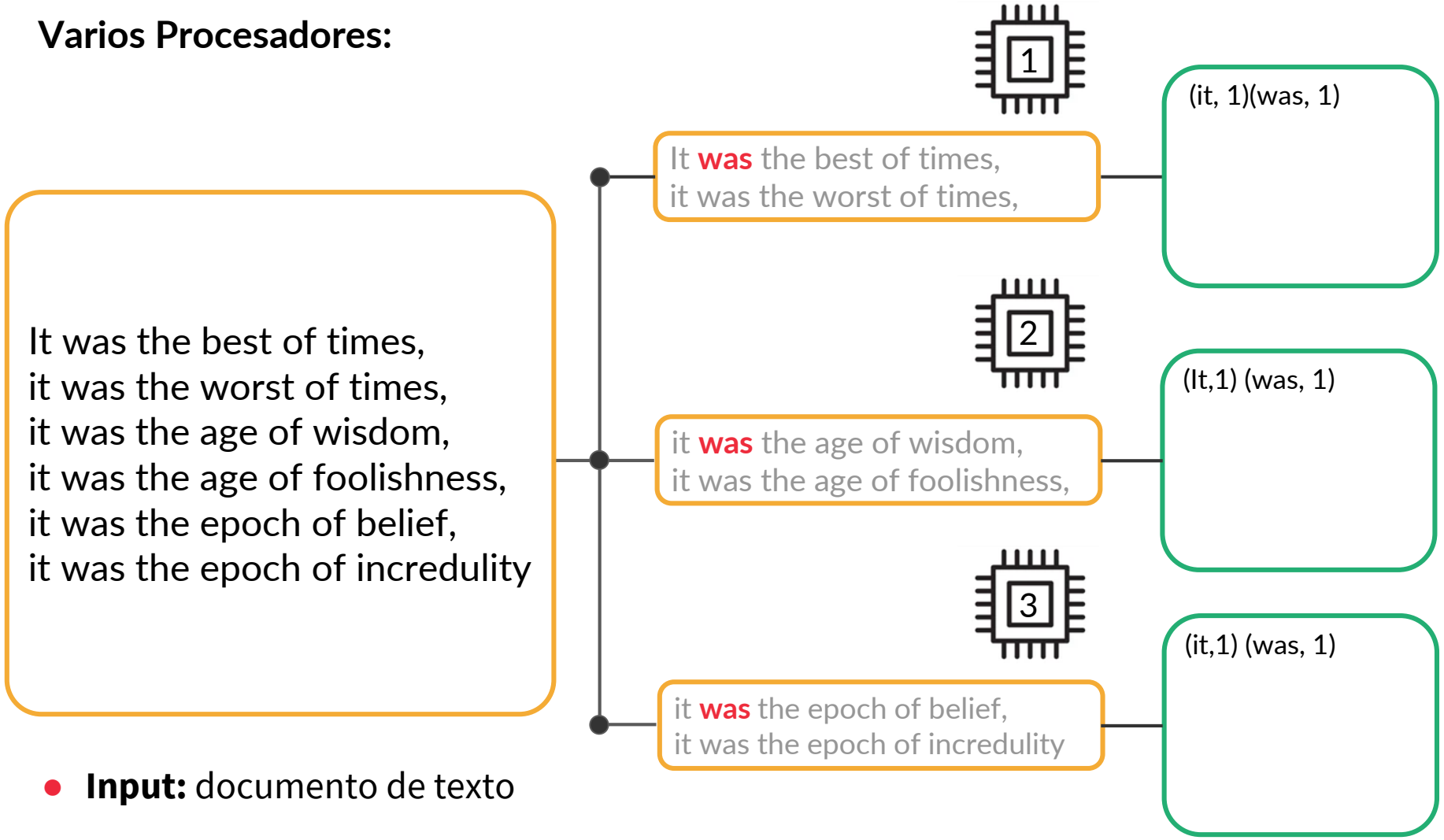
Varios Procesadores:



- **Input:** documento de texto

Paralelizar la solución

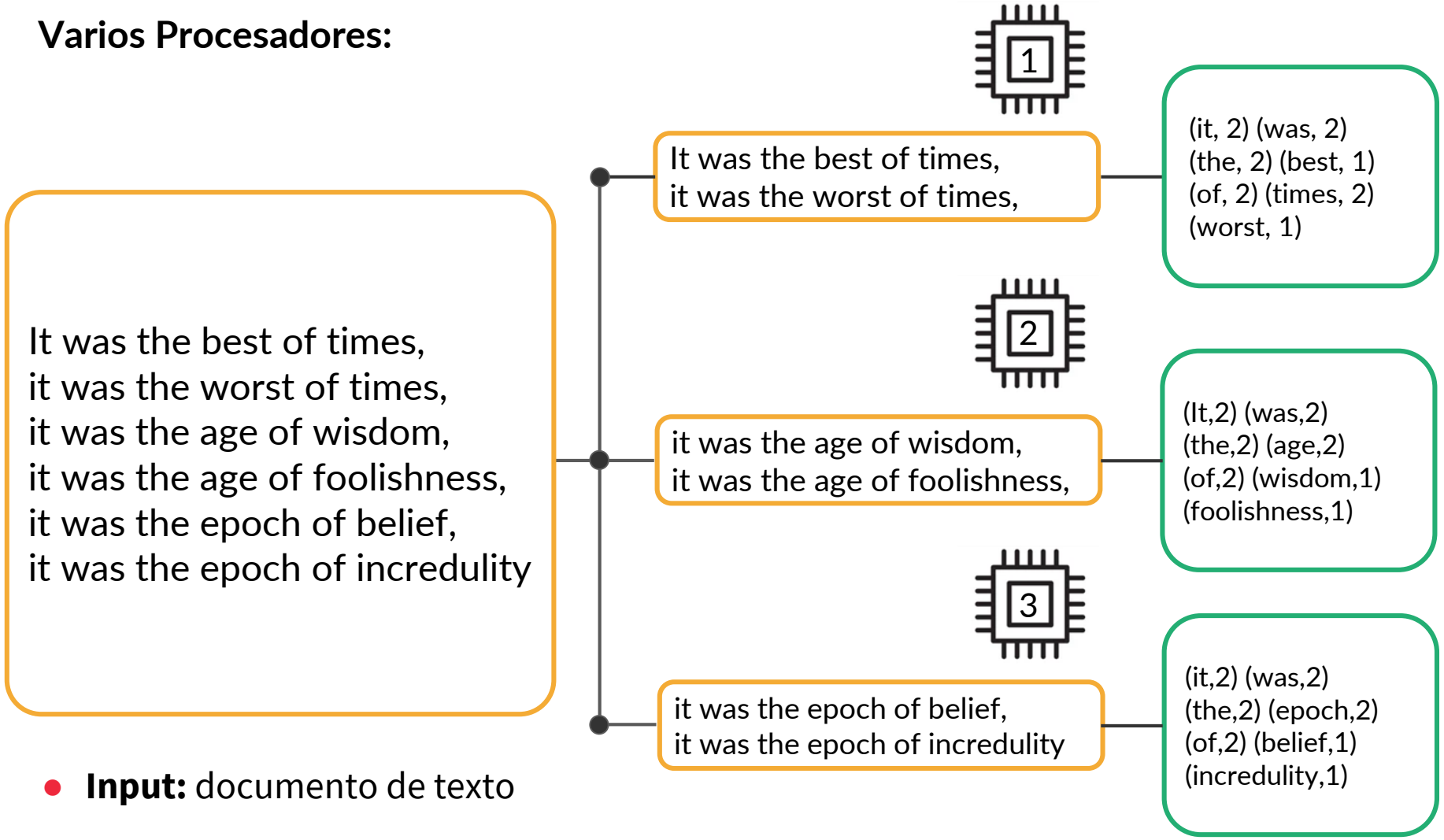
Varios Procesadores:



- **Input:** documento de texto

Paralelizar la solución

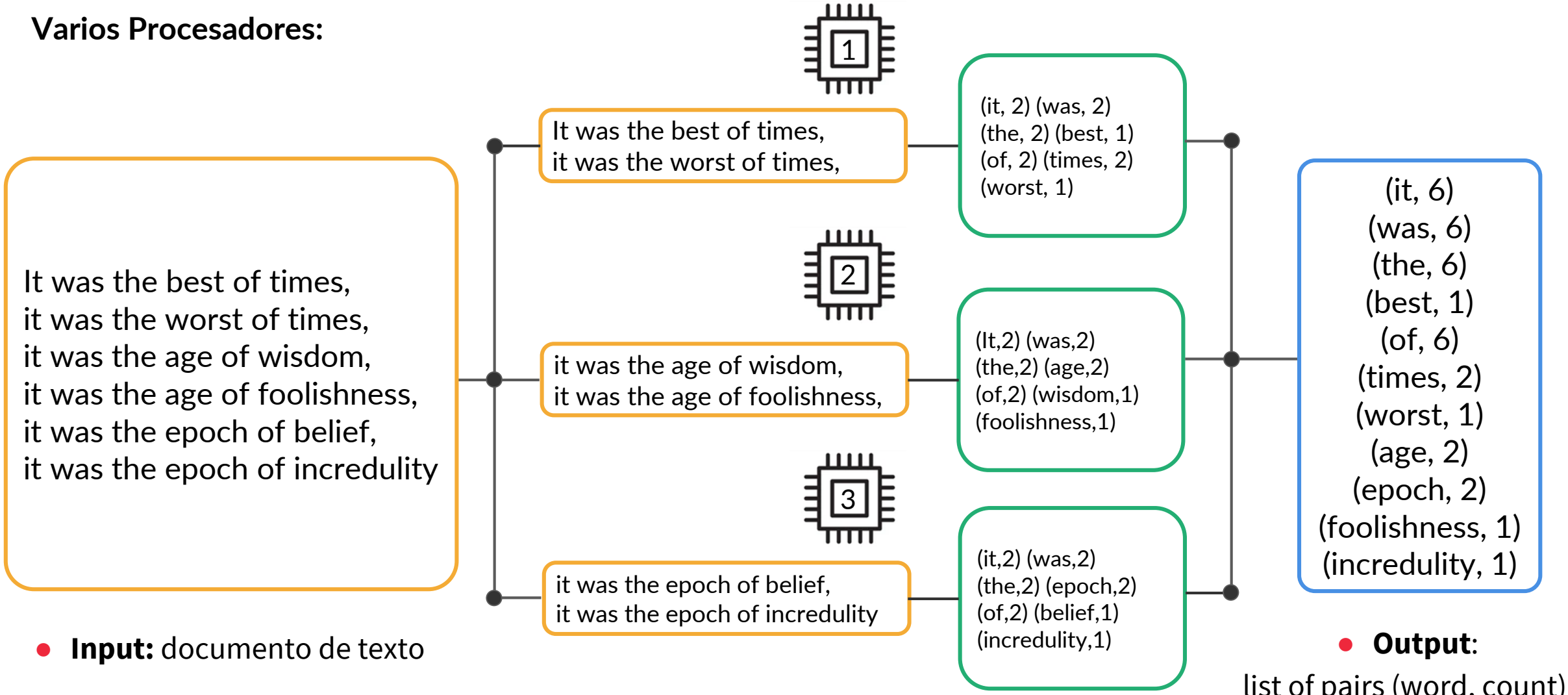
Varios Procesadores:



- **Input:** documento de texto

Paralelizar la solución

Varios Procesadores:



- **Input:** documento de texto

- **Output**:
list of pairs (word, count)

MapReduce

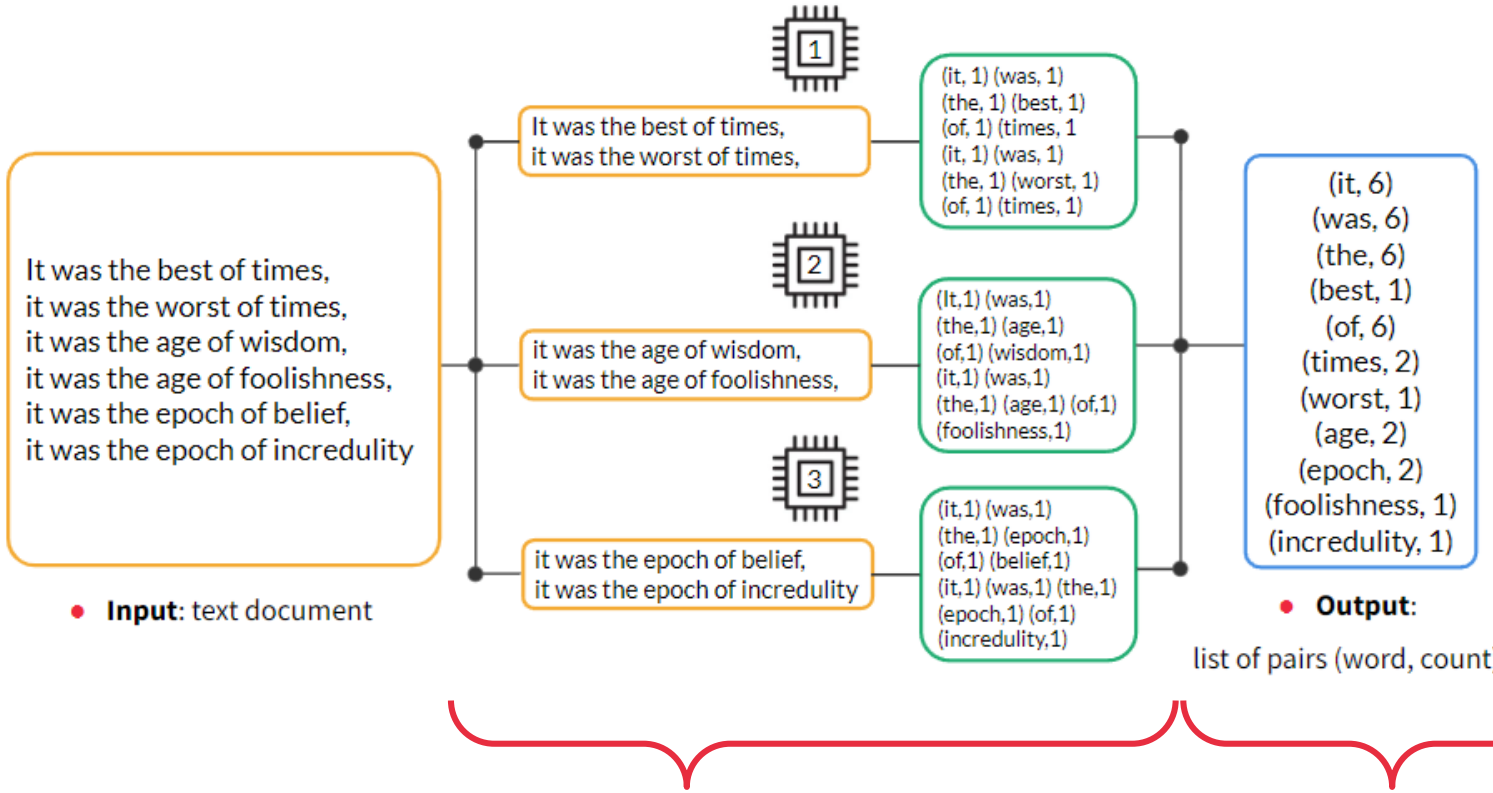
> “A simple and powerful **interface** that enables **automatic parallelization and distribution of large-scale computations**, combined with an **implementation** of this interface that achieves high performance on large **clusters of commodity PCs**.”
>
> **- Dean and Ghermawat, “MapReduce: Simplified Data Processing on Large Clusters”, Google Inc.**

MapReduce

- Modelo de programación paralela , diseñado para escalabilidad y tolerancias a fallos
- Popular implementación basado en Apache Hadoop.
- Usado por ejemplo por Facebook, twiter, linkedin, Aws, google.

  - Construcción de índices para buscadores
  - Clustering de artículos de noticias
  - Minería de datos
  - Detección de spam
  - Análisis astronómico bioinformática , simulaciones climáticas

MapReduce: modelo de programación



- **Input**: text document

- **Output**:

list of pairs (word, count)

Map Function:

- Es usada en la entrada y emite pares clave/valor intermedios.

Reduce Function:

- llama a grupos de pares con la misma clave y emite resultados para esta clave

MapReduce: modelo de programación

Las funciones Map y reduce functions siguen la estructura:

Ejemplo de Code

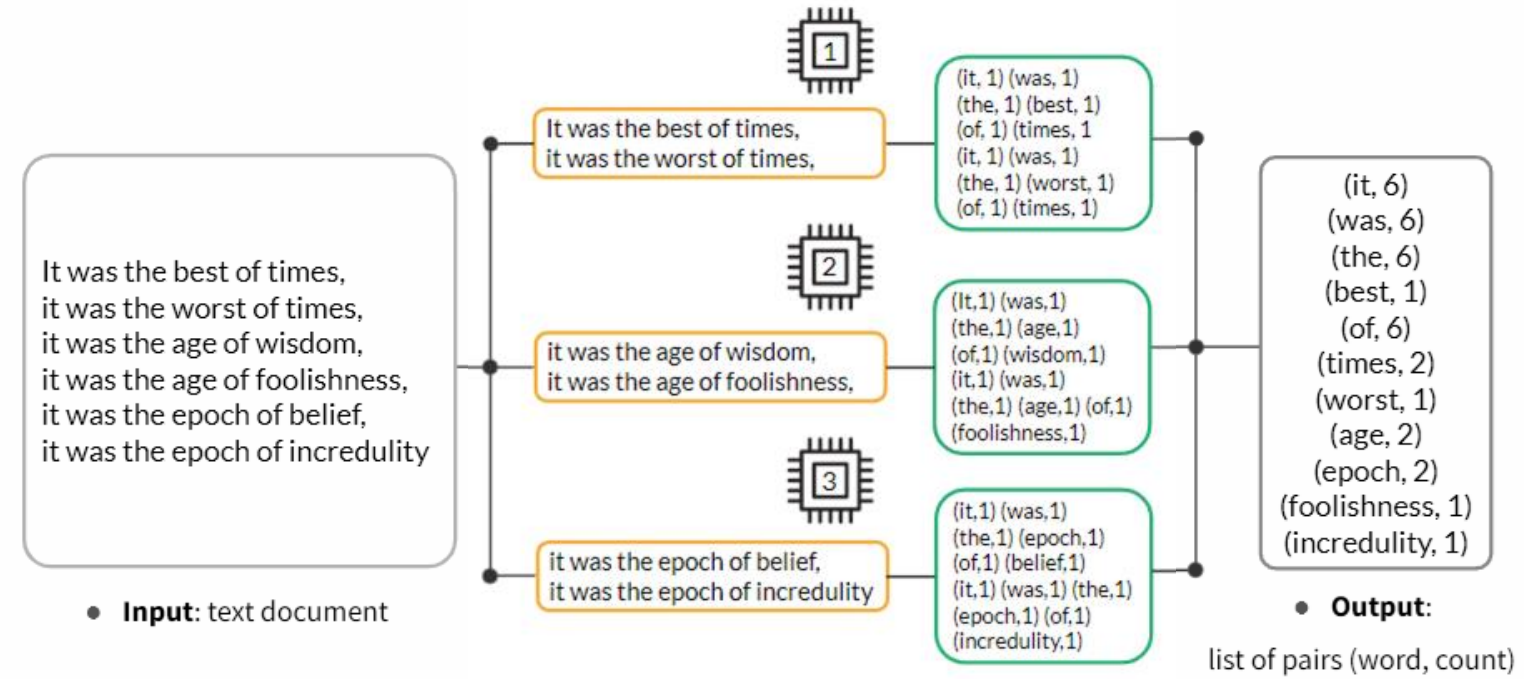
```
def funcname(self,key_in,value_in):
    # logic goes here
    yield(key_out, value_out)
```

- Toma los pares valor
- Realiza un proceso
- Genera nuevos pares clave/valor

La función MAP en el problema word count

En la solución de MapReduce parael problema de word-count, la función **map** :

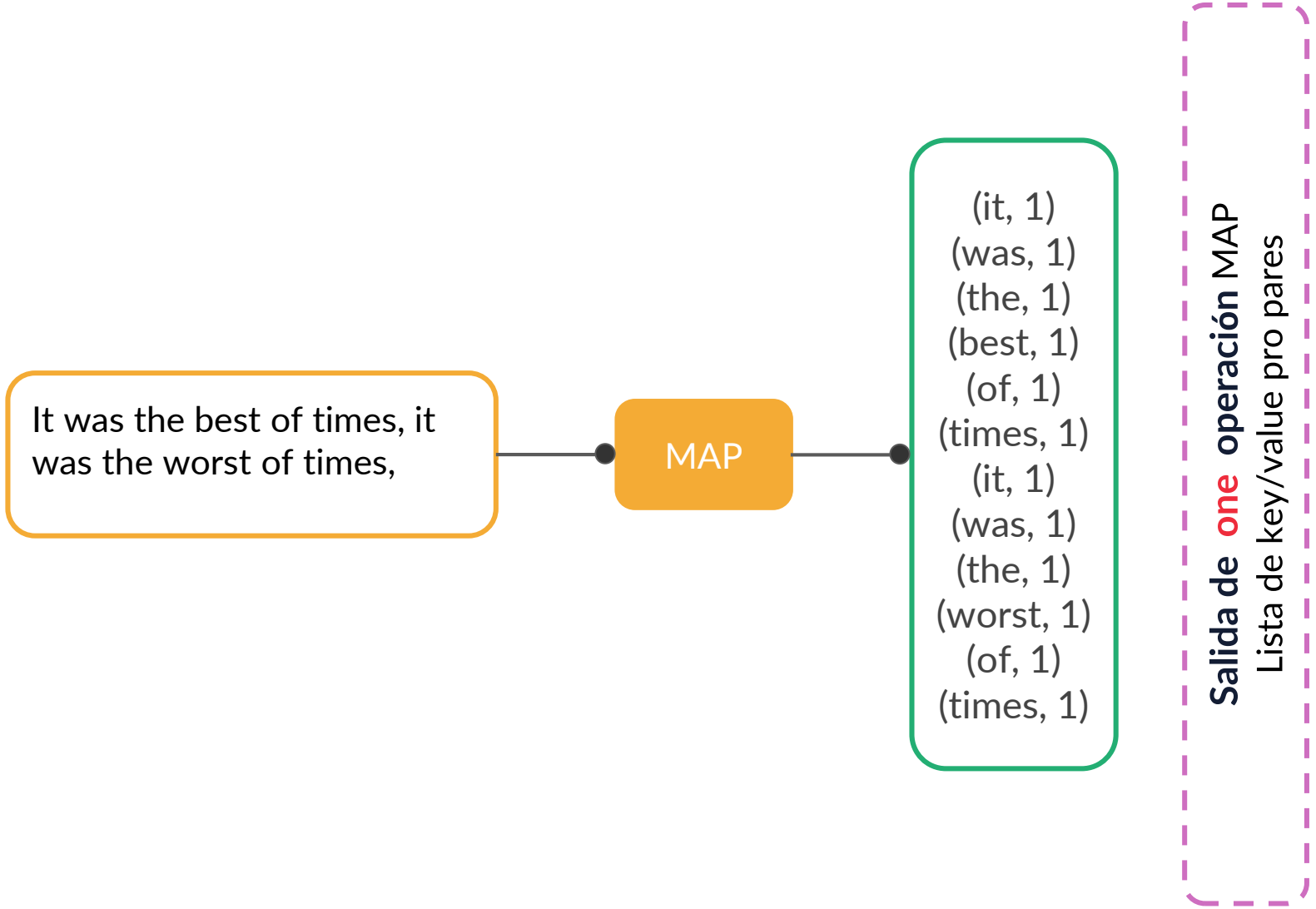
- Toma un elemento de entrada (por ejemplo, una línea) como un valor (la clave se puede ignorar)
- Produce pares clave/valor donde la clave corresponde a cada palabra del texto y el valor es el número 1.



Map code:

```
def mapper(self,_,line):
    for word in line.split():
        yield(word, 1)
```

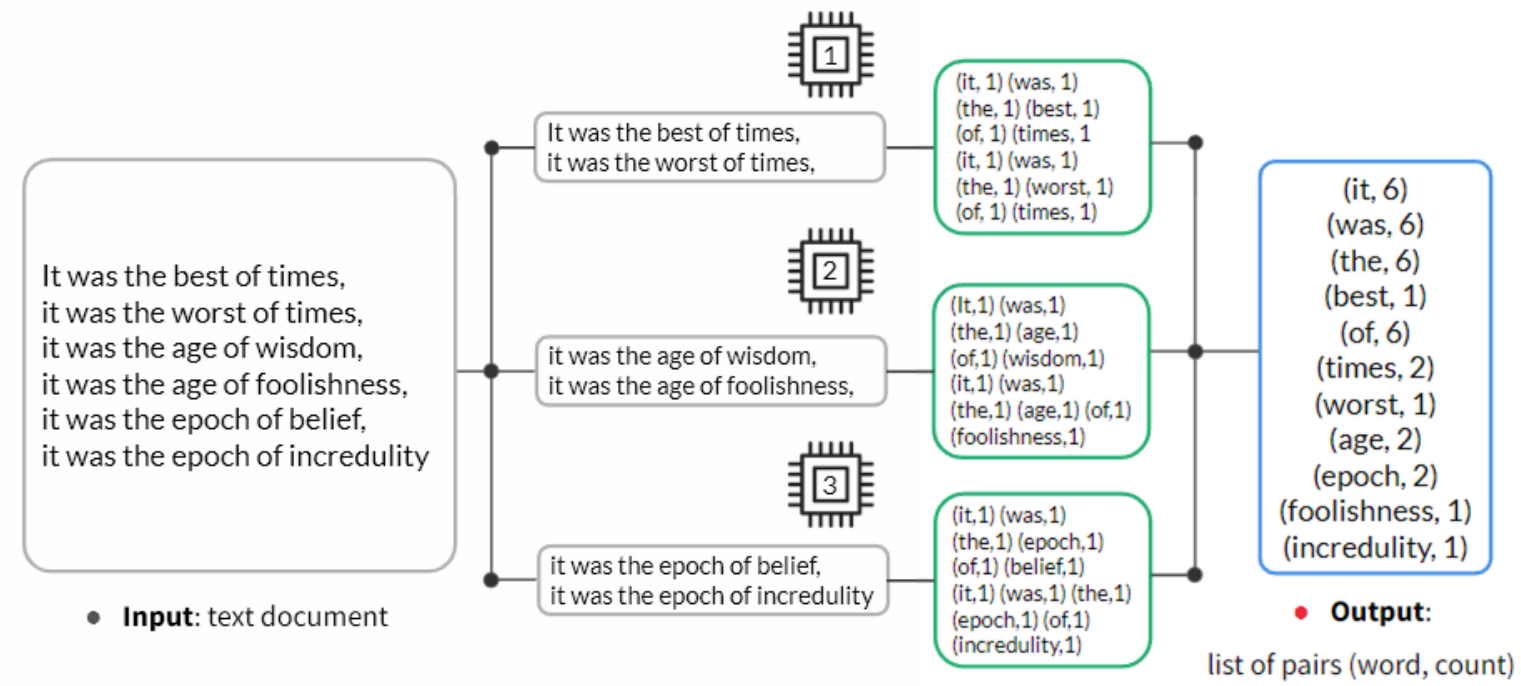
La función MAP en el problema word count



La función REDUCE en el problema word count

La función reduce :

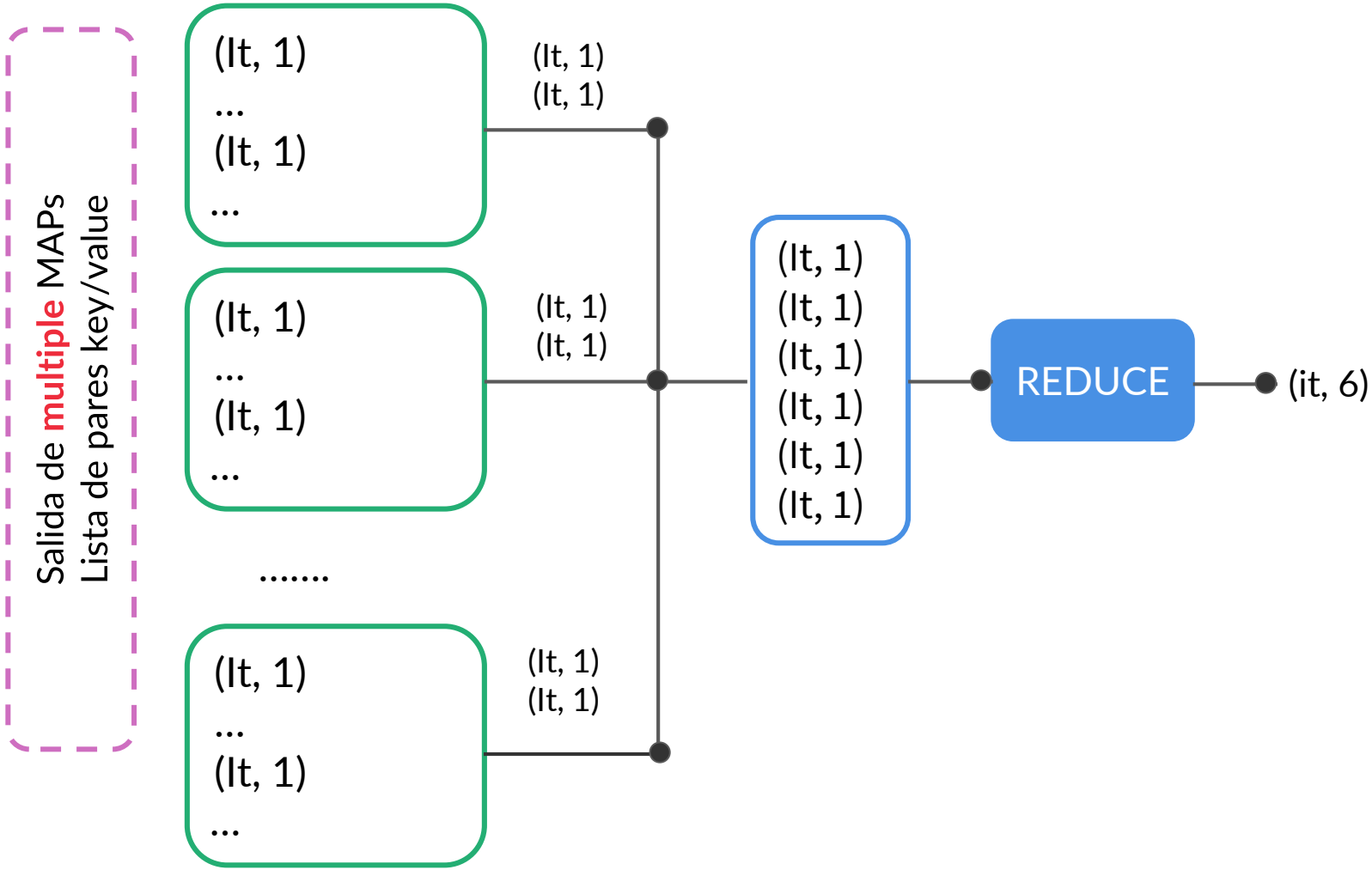
- Toma los pares key/value que tengan las mismas clave (key - word).
- Devuelve un solo par key/value, donde la clave es la palabra y el valor la suma de los valores de los pares recibidos



Reduce Code:

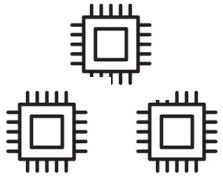
```
def reducer(self, word, values):
    total = sum(values)
    yield(word, total )
```

La función REDUCE en el problema word count



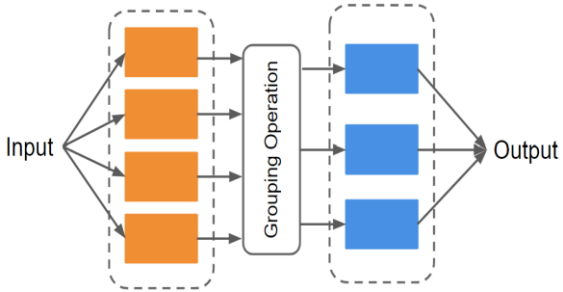
MAPREDUCE: Paralelización

MapReduce permite tener dos opciones para paralelizar el proceso



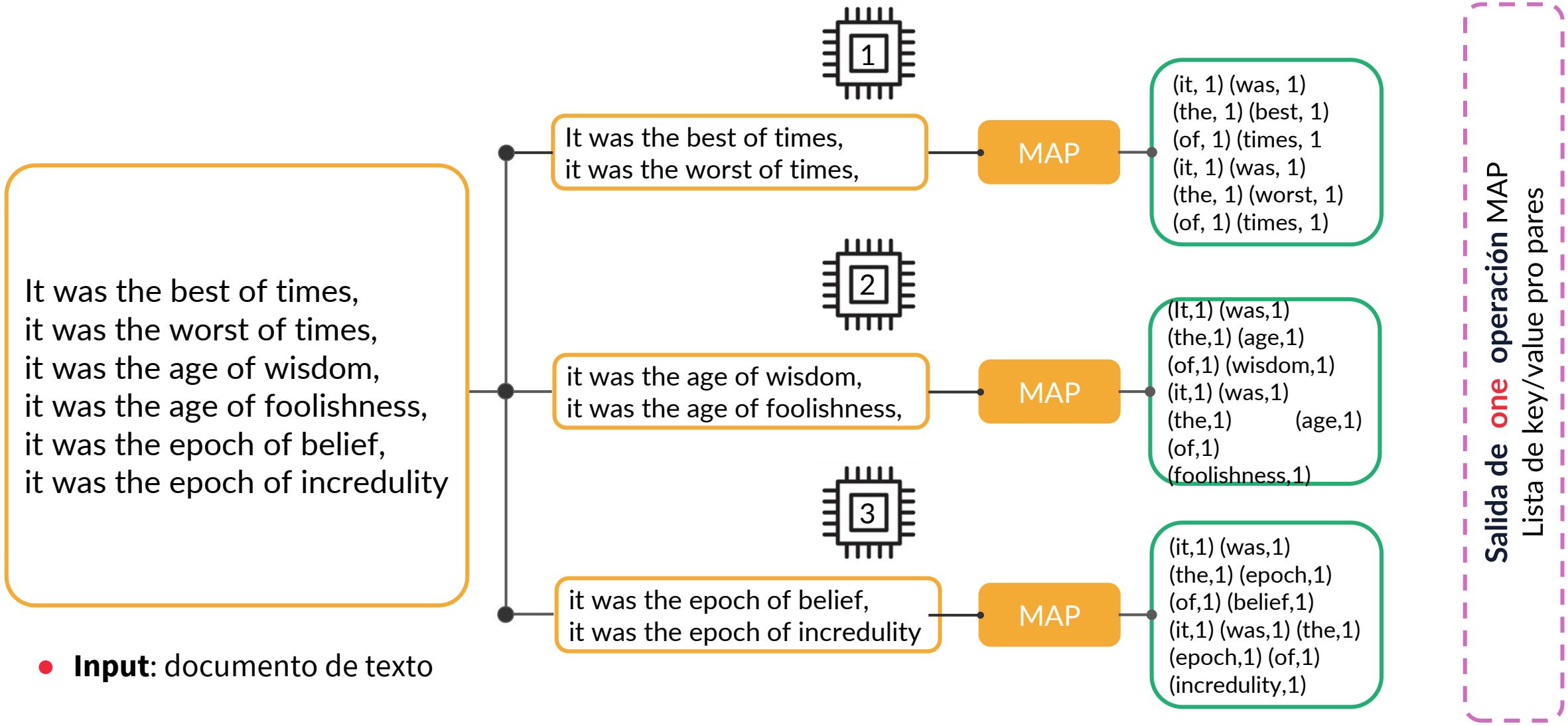
1. Los datos de entrada se pueden dividir en fragmentos
2. Los pares Key/value que comparten la misma clave se pueden agrupar

Por lo tanto, si tenemos una colección de procesadores, podemos paralelizar el problema asignando:



1. Un fragmento de los datos de entrada se asignan a un solo procesador (mapper).
2. Un grupo de pares Key/value para un procesador individual (reducer).

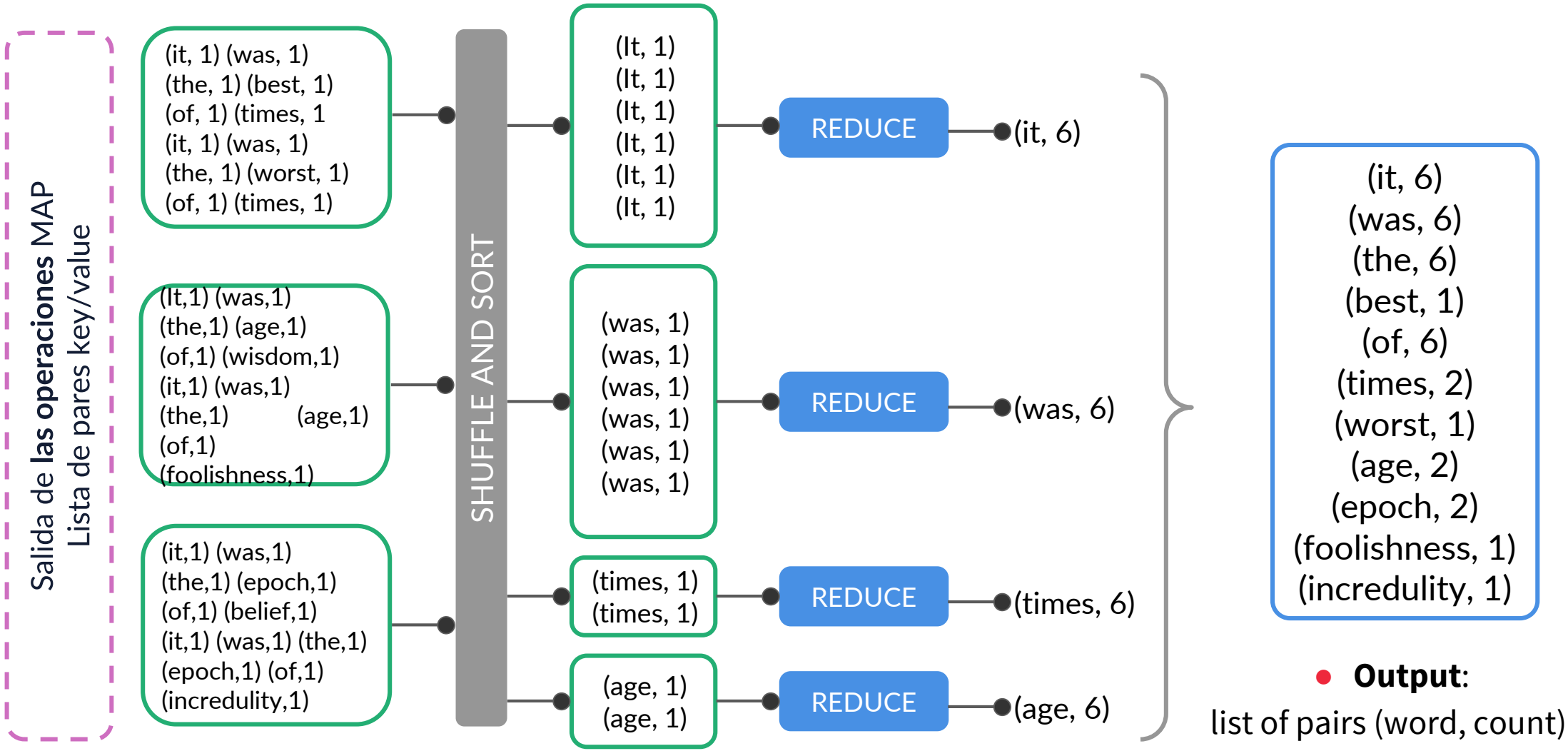
“WORD-COUNT”: Etapa MAP



- **Input**: documento de texto

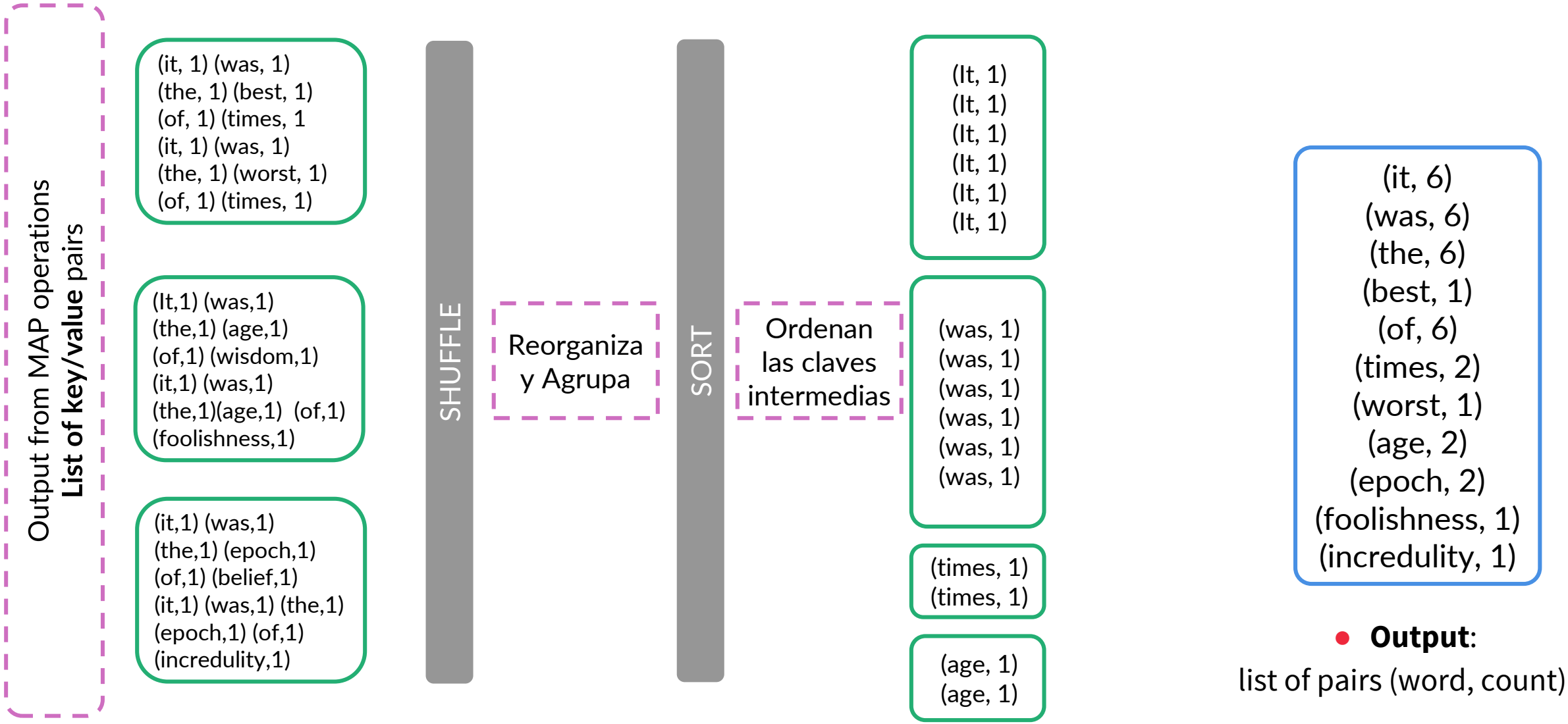
Salida de one operación MAP
Lista de key/value pro pares

“WORD-COUNT”: Etapa REDUCE



- **Output**:
list of pairs (word, count)

SHUFFLE AND SORT



- **Output**:

list of pairs (word, count)

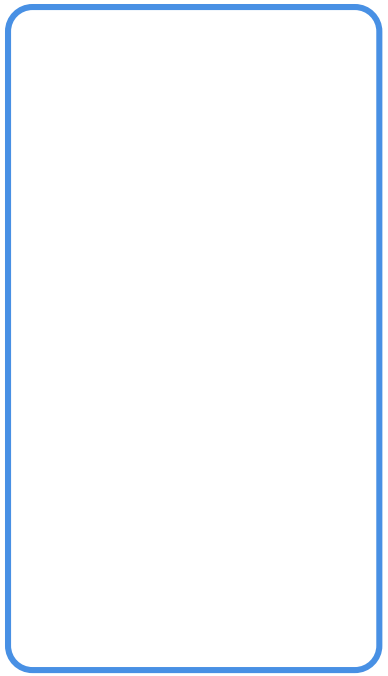
SHUFFLE AND SORT

Salida de **las operaciones** MAP
Lista de pares key/value

(it, 1) (was, 1)
(the, 1) (best, 1)
(of, 1) (times, 1
(it, 1) (was, 1)
(the, 1) (worst, 1)
(of, 1) (times, 1)

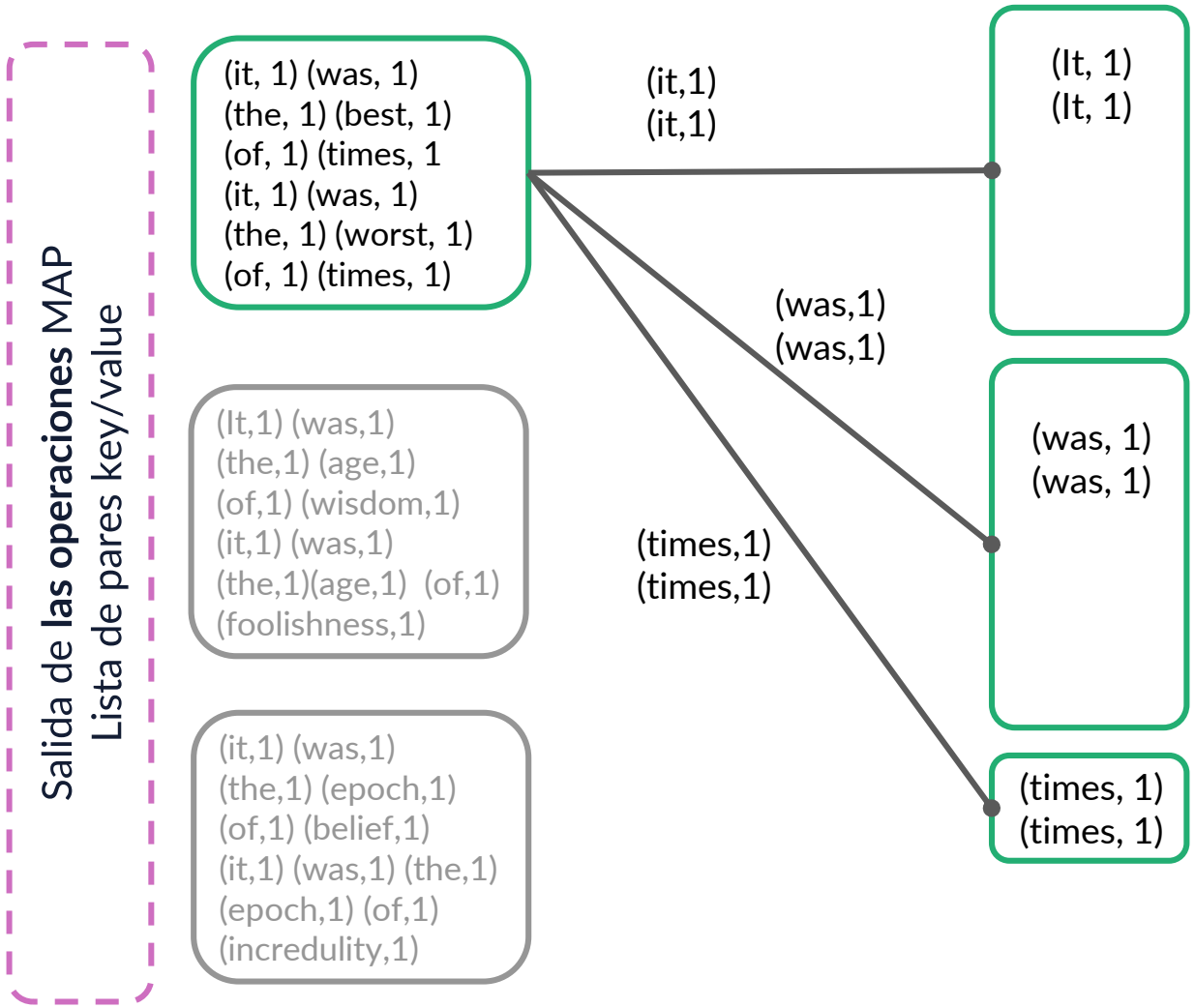
(It,1) (was,1)
(the,1) (age,1)
(of,1) (wisdom,1)
(it,1) (was,1)
(the,1)(age,1) (of,1)
(foolishness,1)

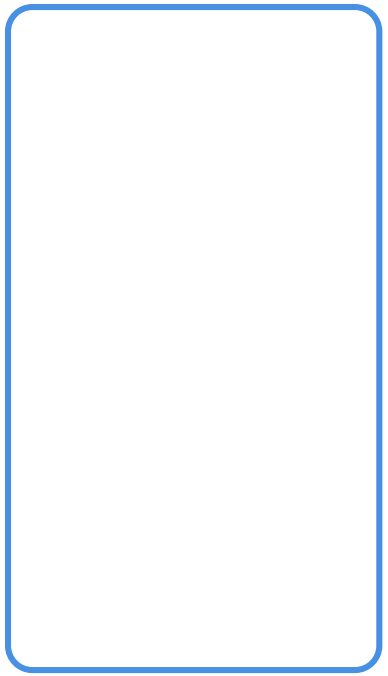
(it,1) (was,1)
(the,1) (epoch,1)
(of,1) (belief,1)
(it,1) (was,1) (the,1)
(epoch,1) (of,1)
(incredulity,1)



- **Output**:
list of pairs (word, count)

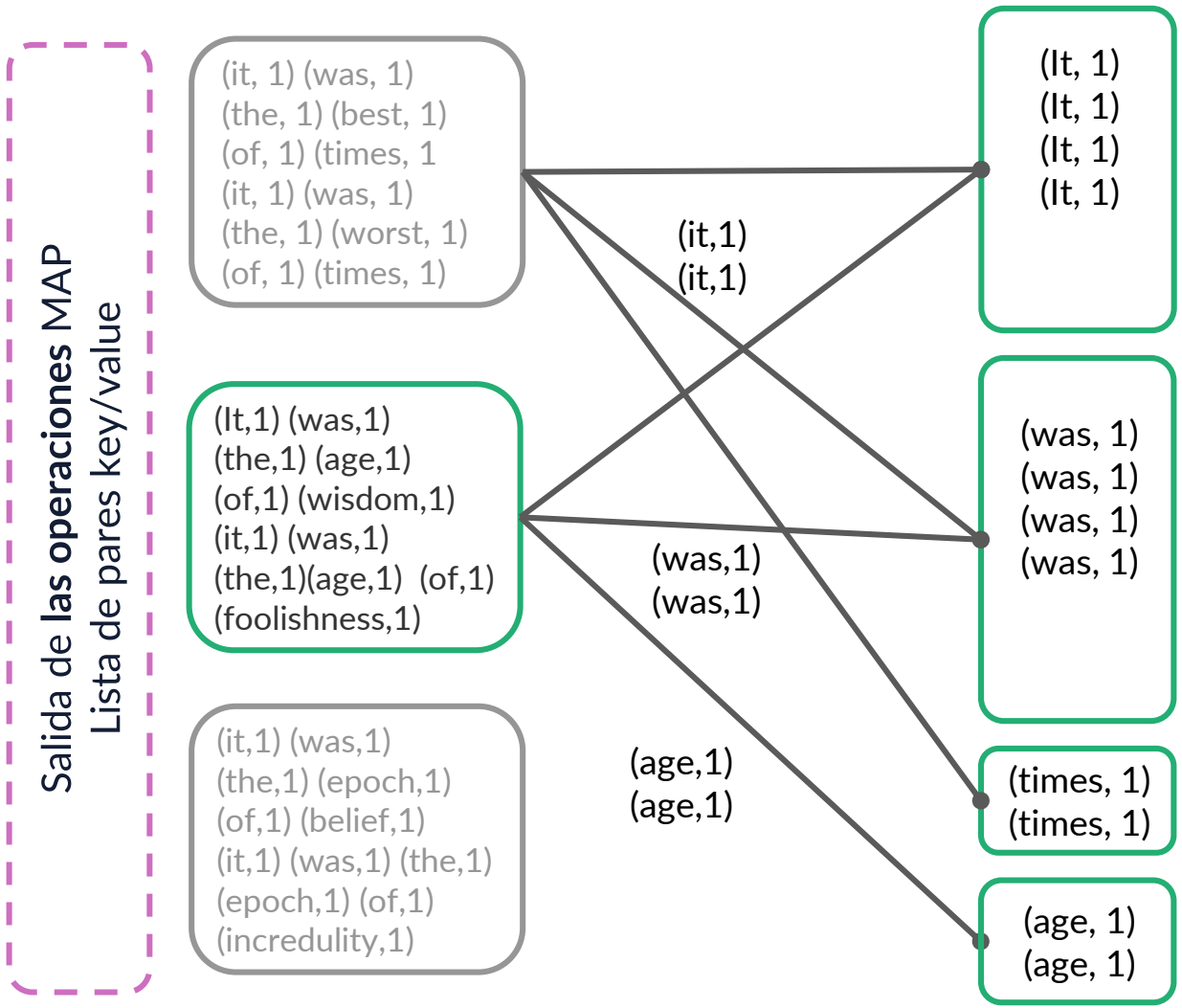
SHUFFLE AND SORT

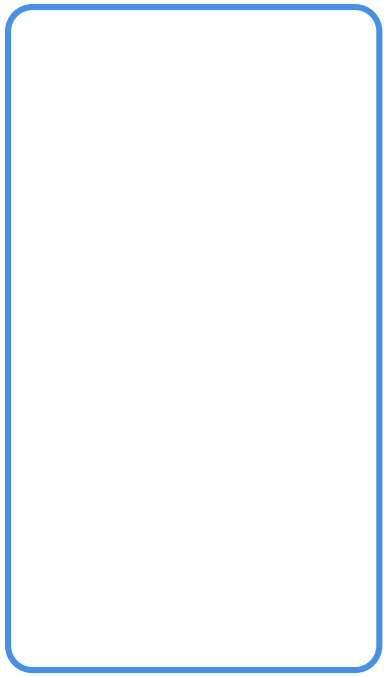




- **Output**:
list of pairs (word, count)

SHUFFLE AND SORT

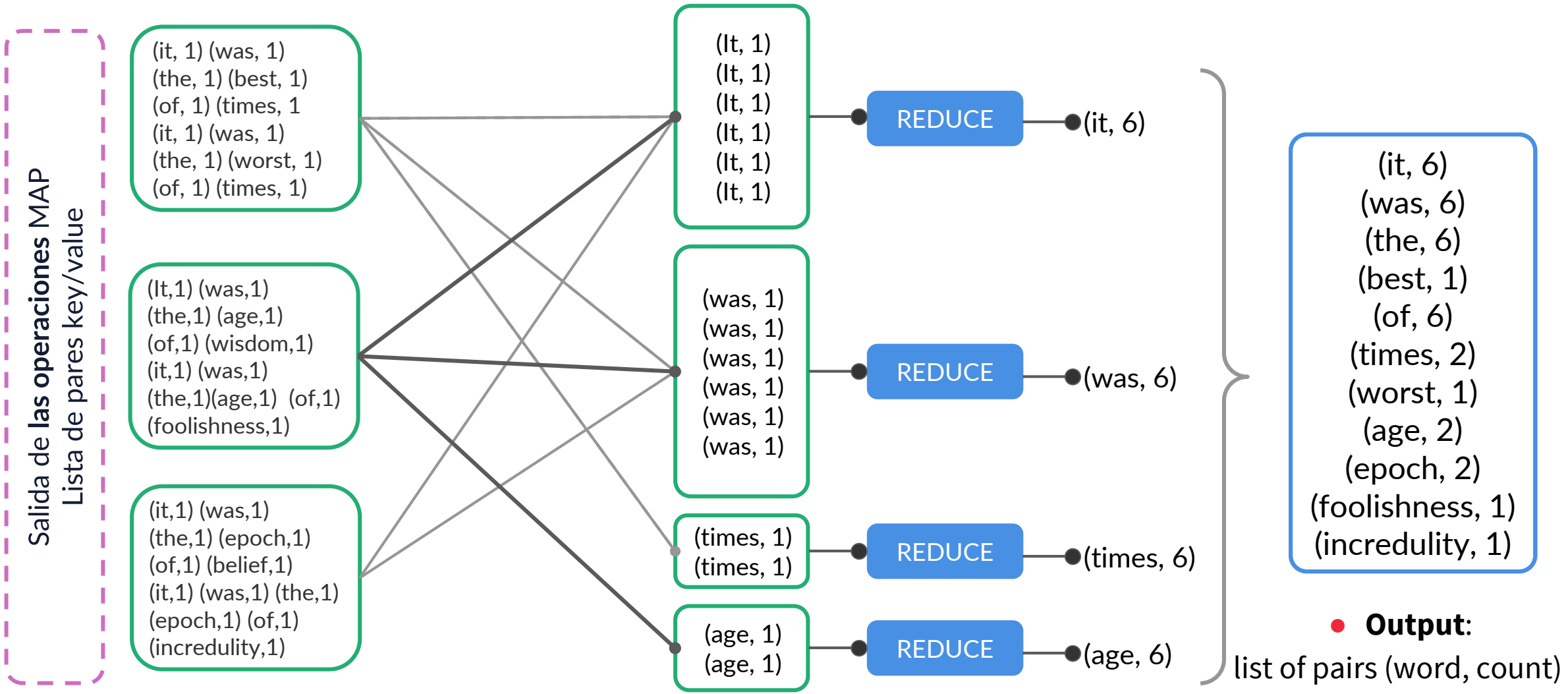




- **Output**:

list of pairs (word, count)

SHUFFLE AND SORT



- **Output**:

list of pairs (word, count)

MAPREDUCE: Implementación

01 El modelo de programación MapReduce define dos funciones, **map y reduce**, que expresan la lógica del problema.

02 Sin embargo, para que esta estrategia funcione, necesitamos **particionar** los datos de entrada y **particionar y mover** los pares **key/value** desde **mappers** a **reducers**.

03 MapReduce especifica esto cómo se implementa. El **framework MapReduce** hace esto por nosotros. Nosotros nos enfocamos en escribir bien las funciones de map y de reduce.

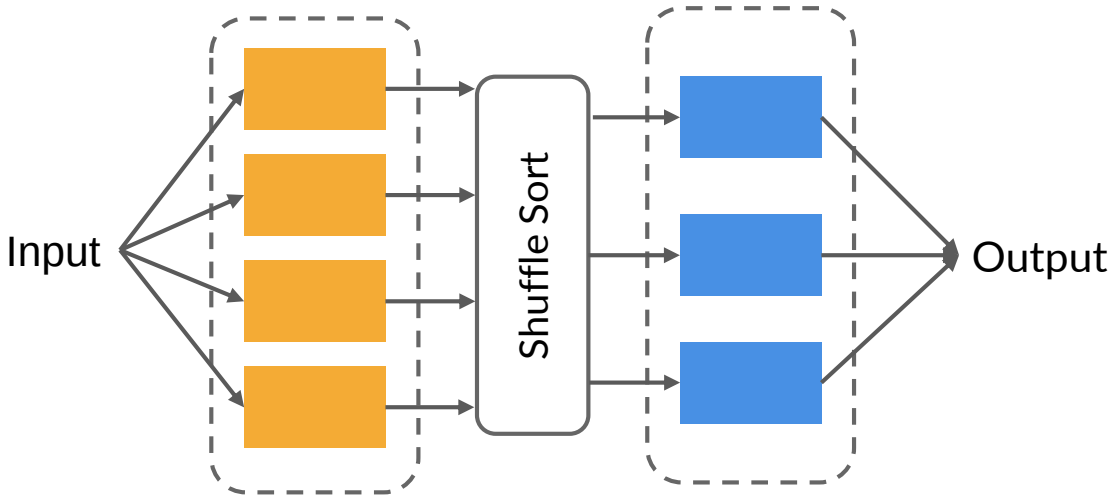
04 **Las misma keys se agrupan y se envian a la etapa de reduce**. Esta parte se realizar por medio del método **Shuffle and Sort**.

MAPREDUCE : Resumiendo

El problema Word-Count puede ser resuelto por :

- Dividir un fragmento de texto en fragmentos más pequeños, Ej. líneas
- Enviar cada linea a multiples funciones mappers.
- Agrupar pares key/value que comparten la misma key.
- Enviar cada grupo de pares key/value a diferentes reducers.

It was the best of times,
it was the worst of times,
it was the age of wisdom,
it was the age of foolishness,
it was the epoch of belief,
it was the epoch of incredulity



(it, 6)
(was, 6)
(the, 6)
(best, 1)
(of, 6)
(times, 2)
(worst, 1)
(age, 2)
(epoch, 2)
(foolishness, 1)
(incredulity, 1)

MAPREDUCE: Bottlenecks and the Combiner

BOTTLENECK
(cuello de botella)

El **cuello de botella** para MapReduce es el costeo de la operación de **shuffle and sort**, ya que, las claves emitidas en los mappers deben ser **copiadas a través de una red.** Además, ordenar (**sorting)** grandes cantidades de elementos individuales es muy **costoso**

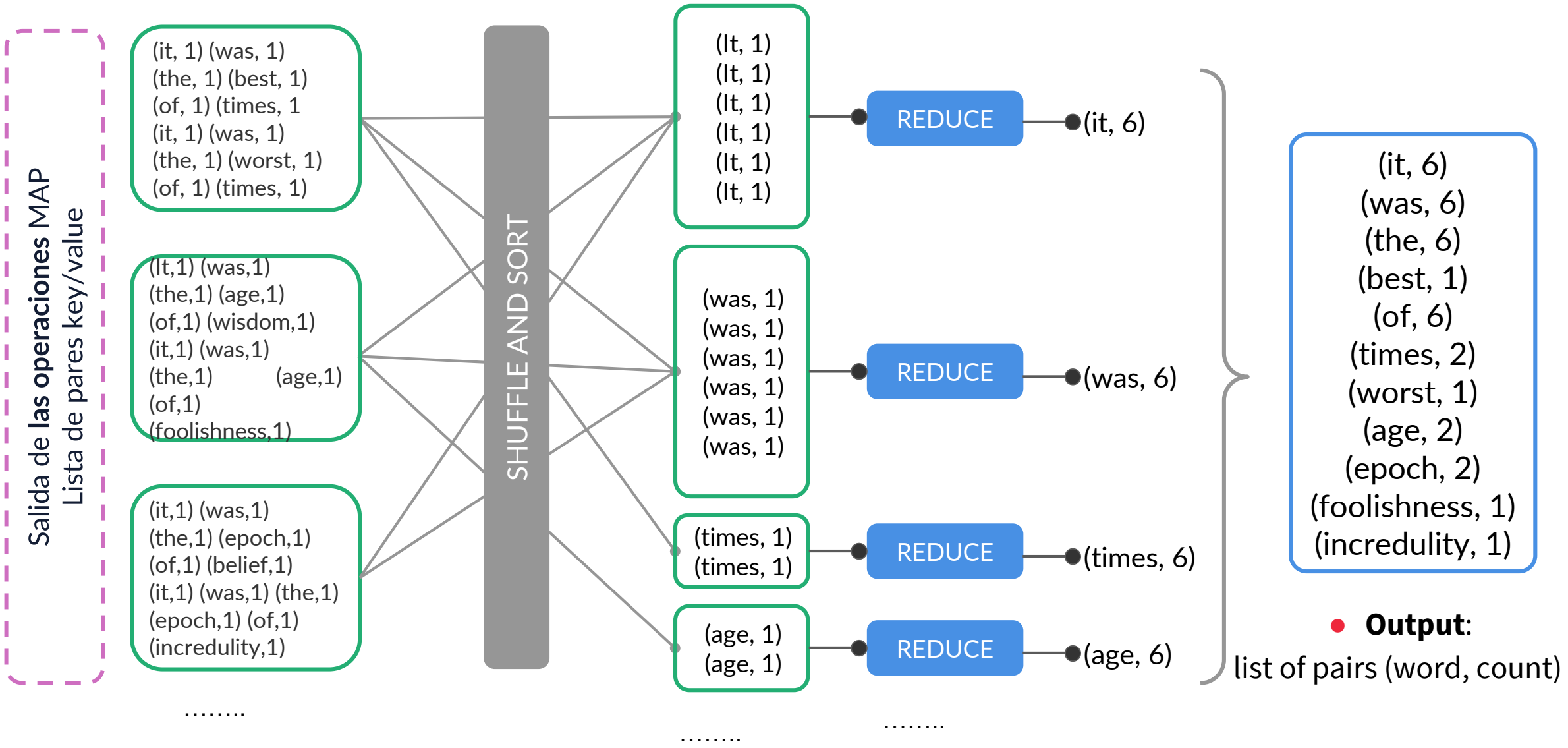
COMBINER

El **combinador** es un paso adicional opcional realizado por un mapper que puede reducir el número de elementos individuales copiados a través de la red y ordenados. Esencialmente, **actúa como un reductor antes de particionar** y copiar a través de la red todas las parejas clave/valor

Combinador para Word count

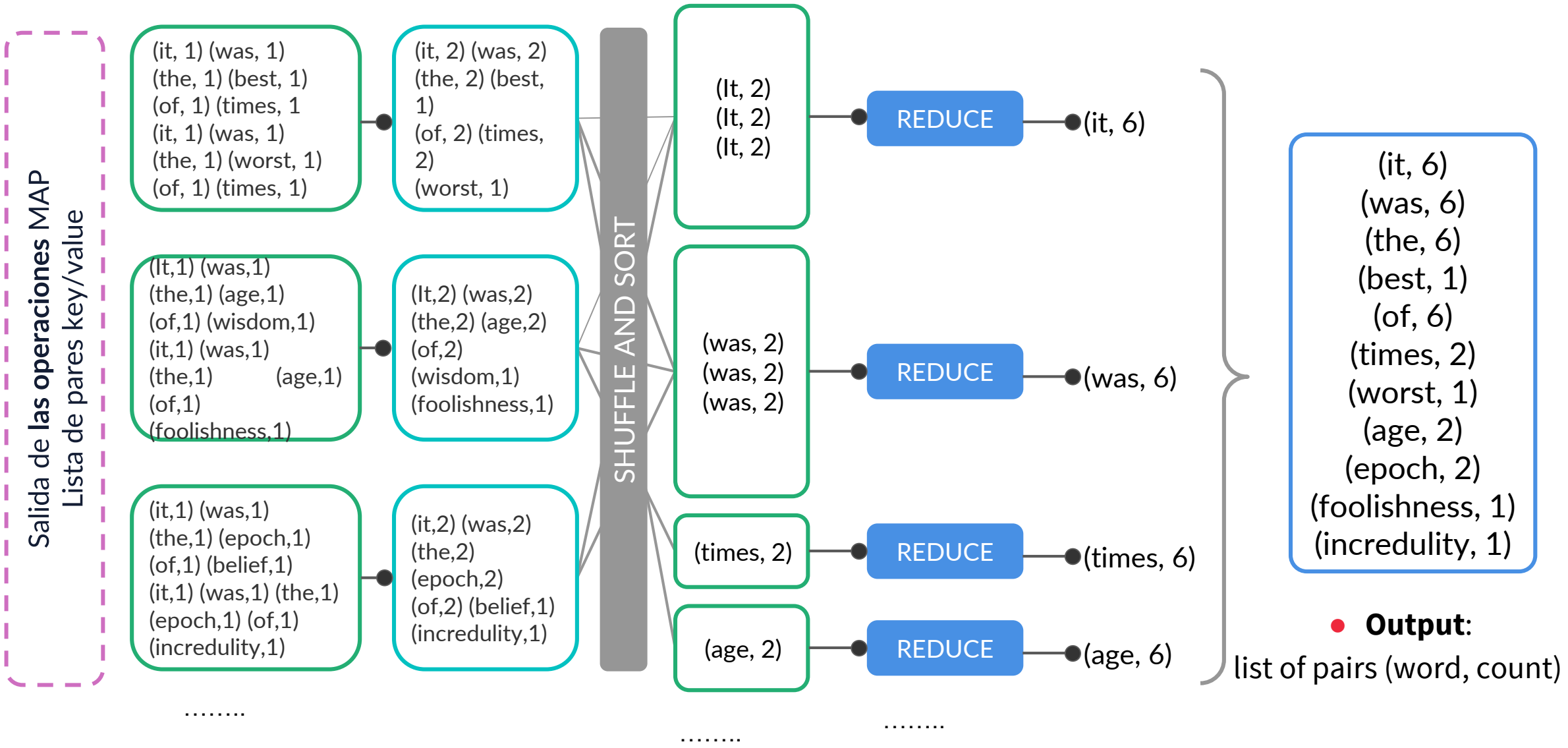
```
def mapper(self,_,line):
    for word in line.split():
        yield(word, 1)

def reducer(self, word, values):
    total = sum(values)
    yield(word, total)

def combiner(self, word, values):
    total = sum(values)
    yield(key, total)
```

WORD-COUNT SIN Combinador

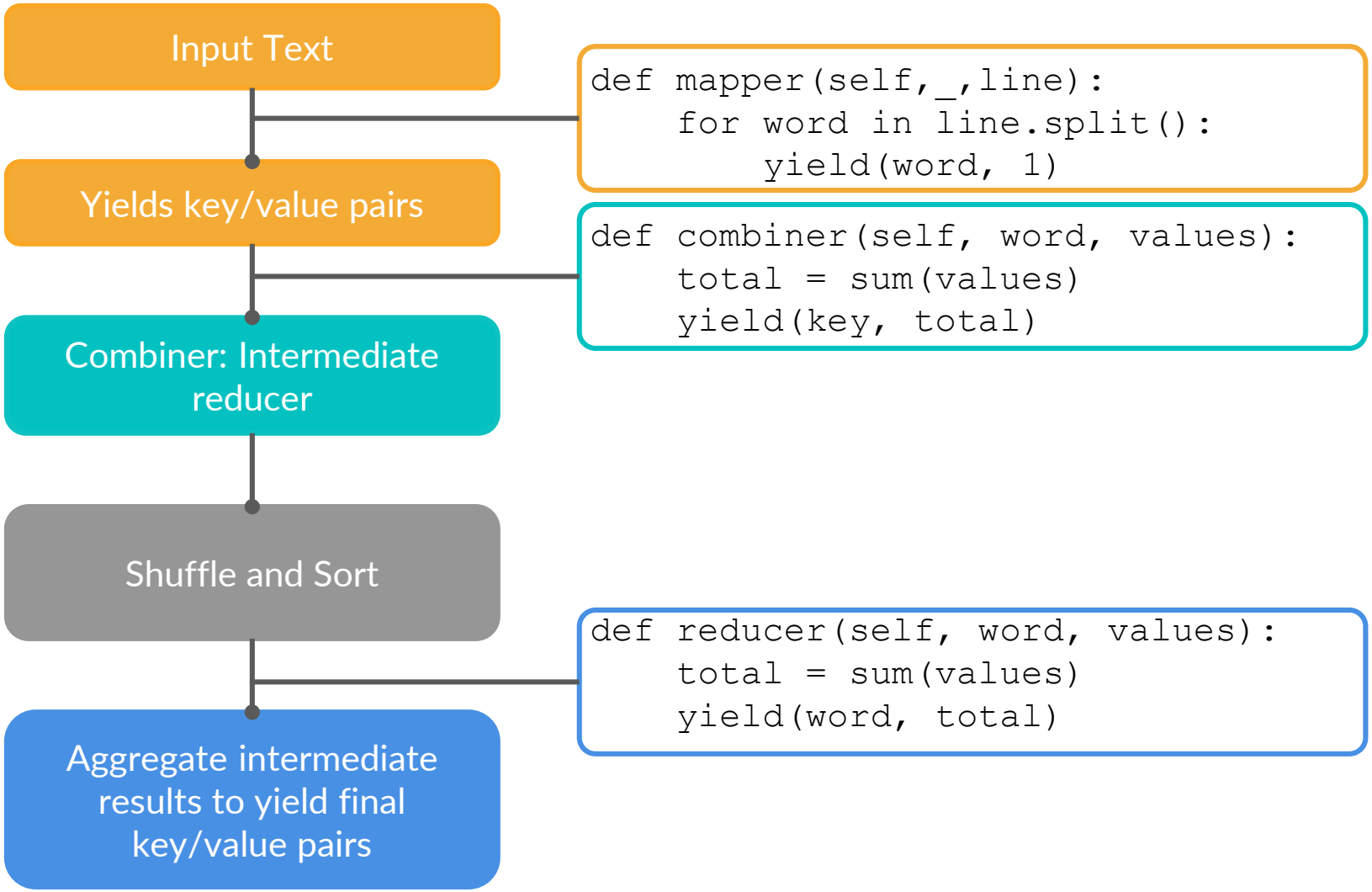


- **Output**:
list of pairs (word, count)

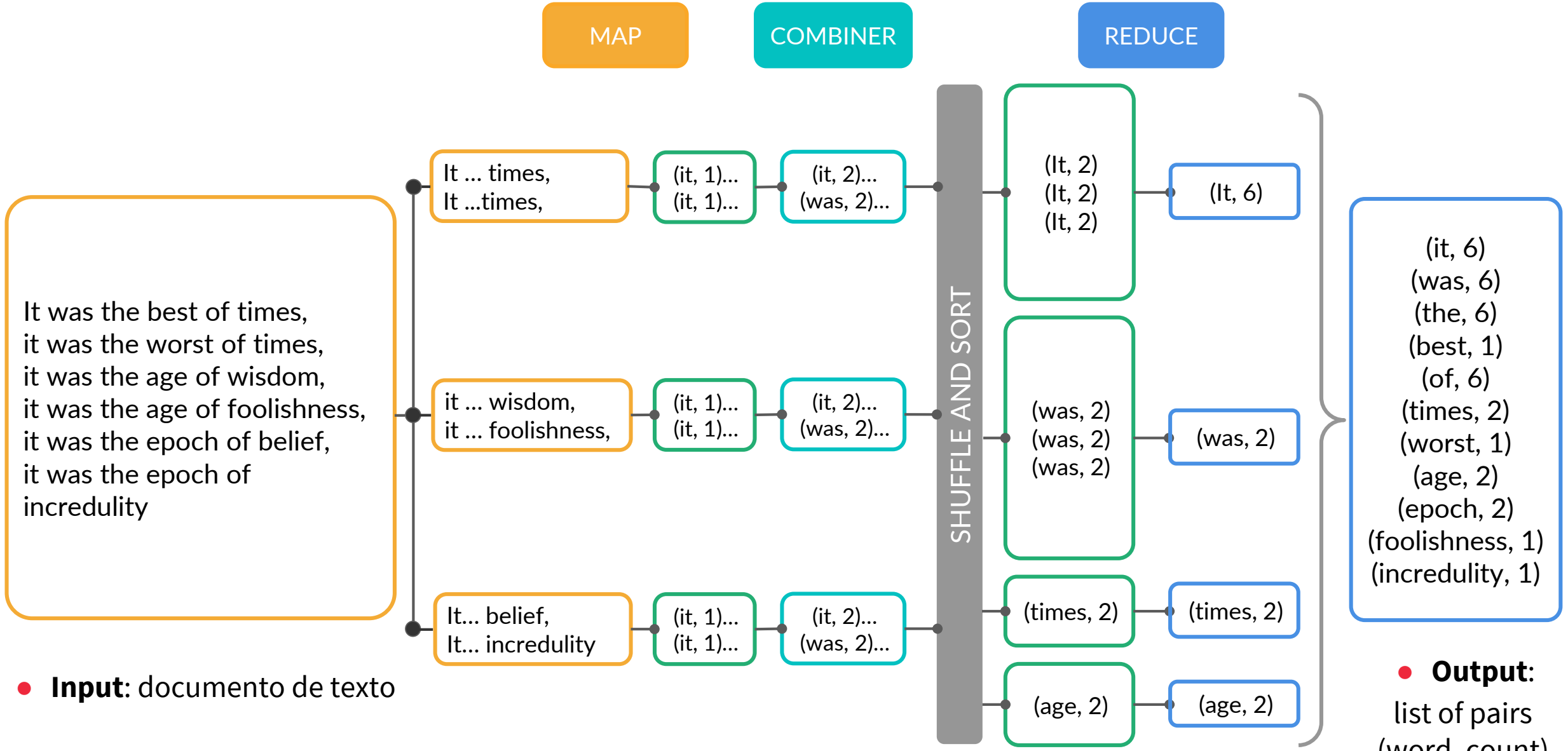
WORD-COUNT CON Combinador



WORD-COUNT CON Combinador



MAPREDUCE : WORD-COUNT



- **Input**: documento de texto

- **Output**: list of pairs (word, count)

Contenidos

- Introducción a MapReduce
- **Patrones de programación MapReduce**

Chapter 6. Erl, T. (2013). Big Data Fundamentals: Concepts, Drivers & Techniques. Prentice Hall.Big

Patrones de programación

MapReduce es una abstracción conveniente para resolver problemas de Big Data.

Como científicos de datos, escribimos las funciones map y reduce de manera simple, mientras que la implementación de MapReduce lleva a cabo el trabajo de bajo nivel, incluyendo la distribución de tareas en los nodos de un clúster.

- Para expresar nuestro problema con el framework de MapReduce, es importante conocer los patrones de programación de MapReduce más comunes.
- Algunos problemas son aptos para el modelo MapReduce, mientras que otros no lo son.
- Una familia de problemas cuyas soluciones siguen la misma estrategia para ser implementadas en MapReduce se llama patrón de programación de MapReduce

Patrón 1: index invertido

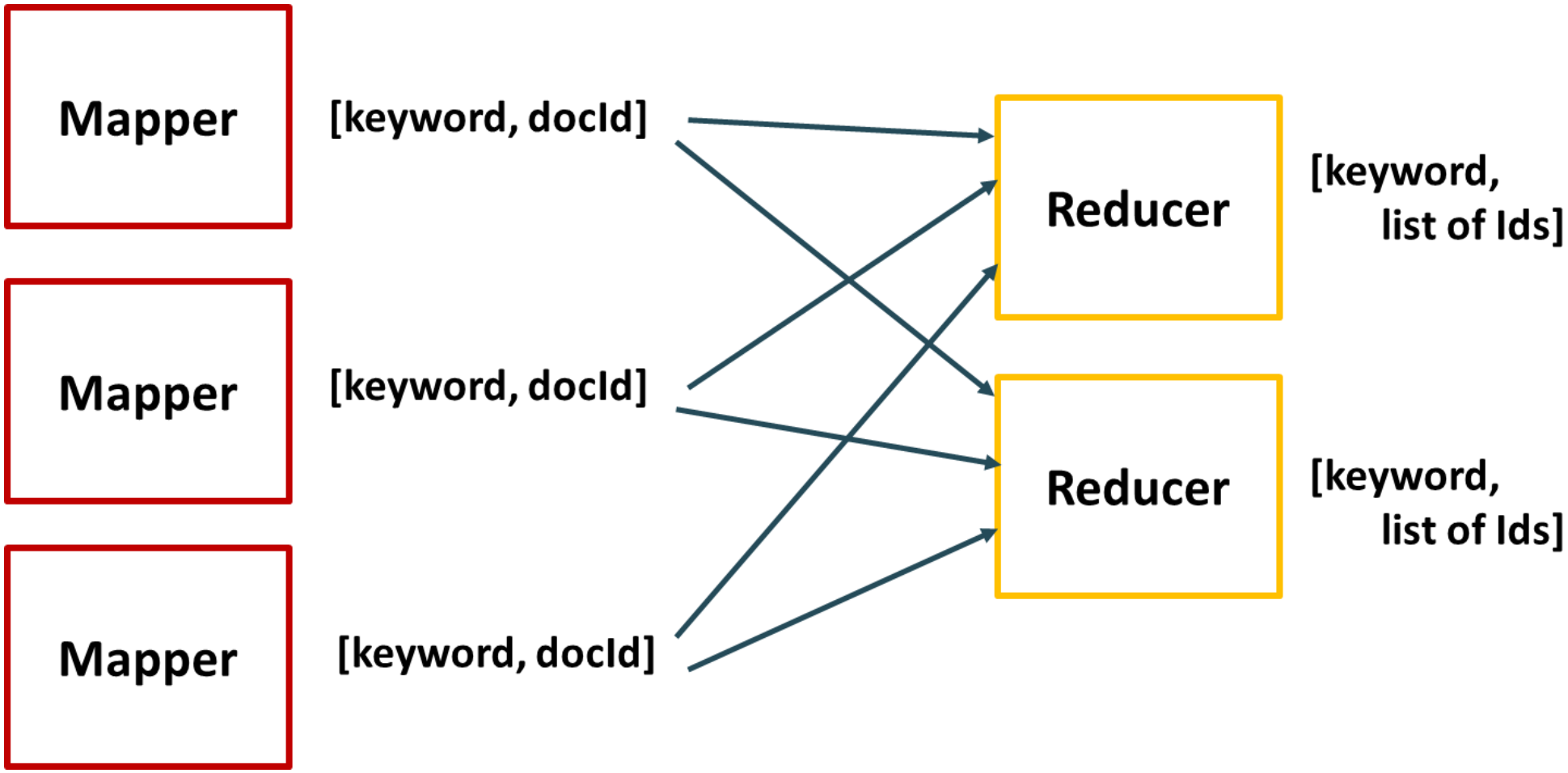
Objetivo: generar un índice a partir de una colección de elementos para permitir búsquedas más rápidas.

La búsqueda identifica los registros que contienen una característica específica.

Ejemplos:

- Construir un índice a partir de un libro de texto.
- Encuentra todos los sitios web que coincidan con un término de búsqueda

Patrón 1: Estructura index invertido



Patrón 1: index invertido

Mapper

1. Buscar las características en los input
2. Retornar |el listado (características, doicd)

Combiner???

Reducer

1. Identificar función (arroja las listas de resultados recibidos)

Patrón 1: index invertido, pseudocode

```
def mapper(docId, text):
    features = find_features(text)
    for(feature in features):
        emit(feature, docId)


def reducer(feature, docIds):
    emit(feature, formatNicely(docIds))
```

Patrón 2: Filtros

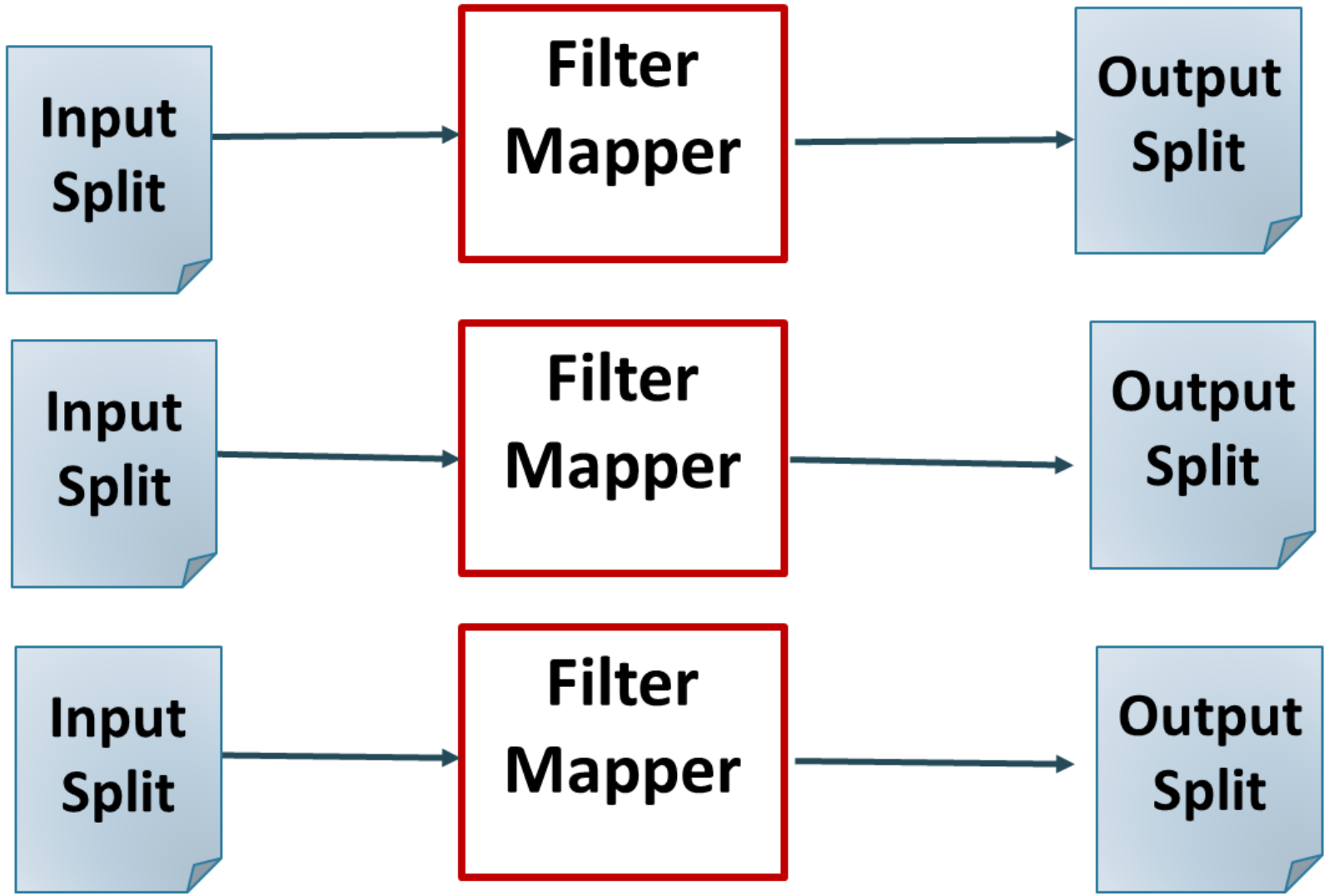
Objetivo: filtrar los elementos que no se necesitan para realizar más cálculos. Acelera los cálculos posteriores, ya que reduce el tamaño de la colección de elementos a procesar.

Examples:

- Tracking a thread de eventos (Registros del mismo usuario)
- Limpieza de Data

El filtrado es un trabajo solo para el mapeador: no produce una agregación.

Patrón 2: Filtros Estructura



Patrón 3: Top Ten

Objetivo: Recupere una pequeña cantidad de registros, en relación con una métrica de clasificación

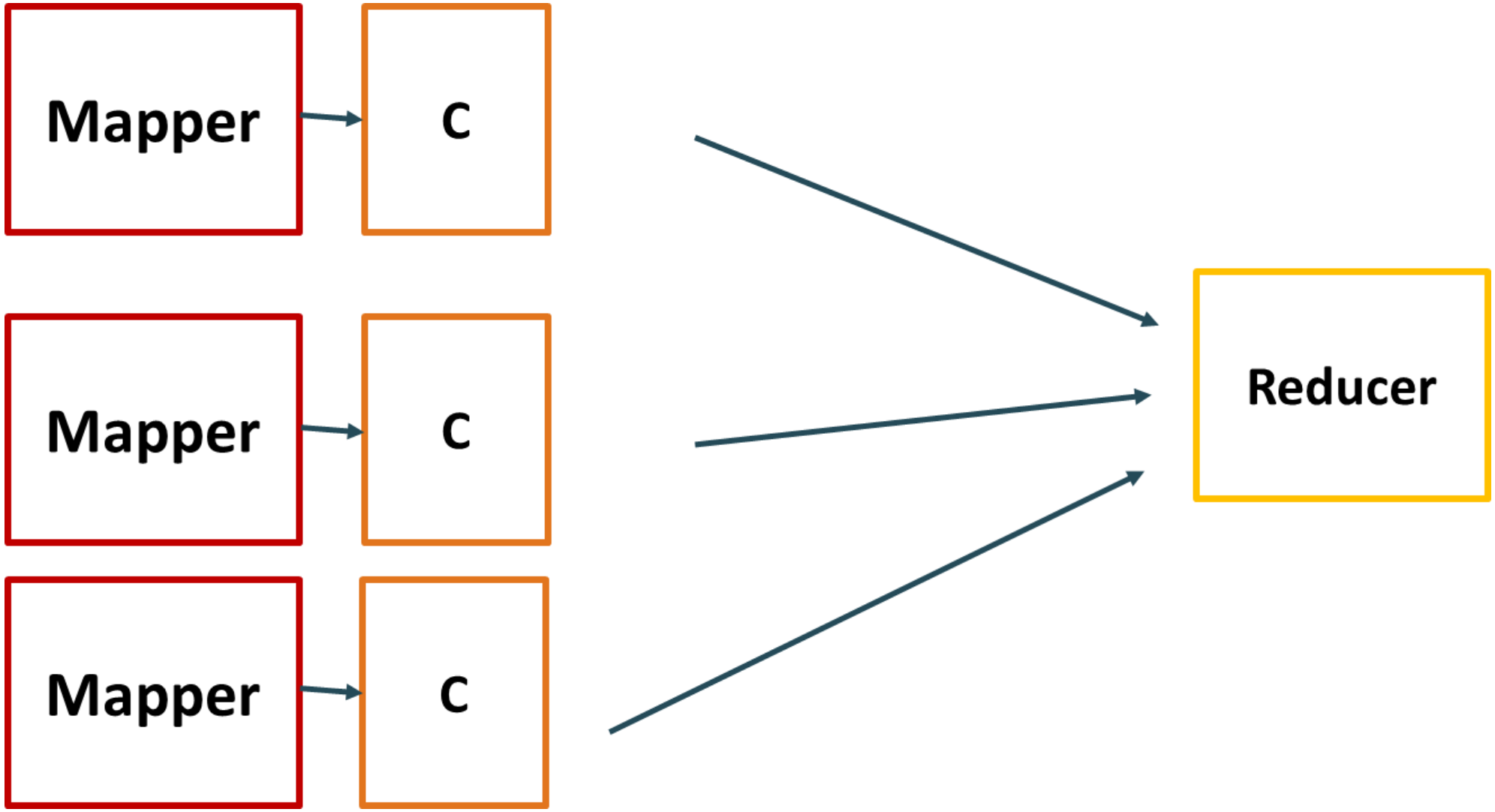
Ejemplo:

- Crear lista de los mejores vendedores
- Buscar valores atipicos en los datos

En este patrón solo se utilizará un reductor. Por lo tanto, no hay necesidad de particionar y barajar.

La clasificación será parte de la función de reducción.

Patrón 3: Top Ten



Patrón 3: Top Ten

Mapper

1. Analiza las características de los pares
2. Retornar (None, pair)

Combiner

Igual que el reductor

Reducer

1. Ordenar todos los pares de acuerdo a su valor
2. Obtener los diez primeros pares
3. Generar cada par top

Patrón 3: Top Ten pseudocode

```
def mapper(_, row):
     studentId = parseId(row)
     grade = parseGrade(row)
     pair = (studentId, grade)
     emit(None, pair)

def reducer(_, pairs):
      top10 = pairs.sort().getTop(10)
       rank = 1
       for(student in top10):
            emit(rank, student[0])
             rank += 1
```

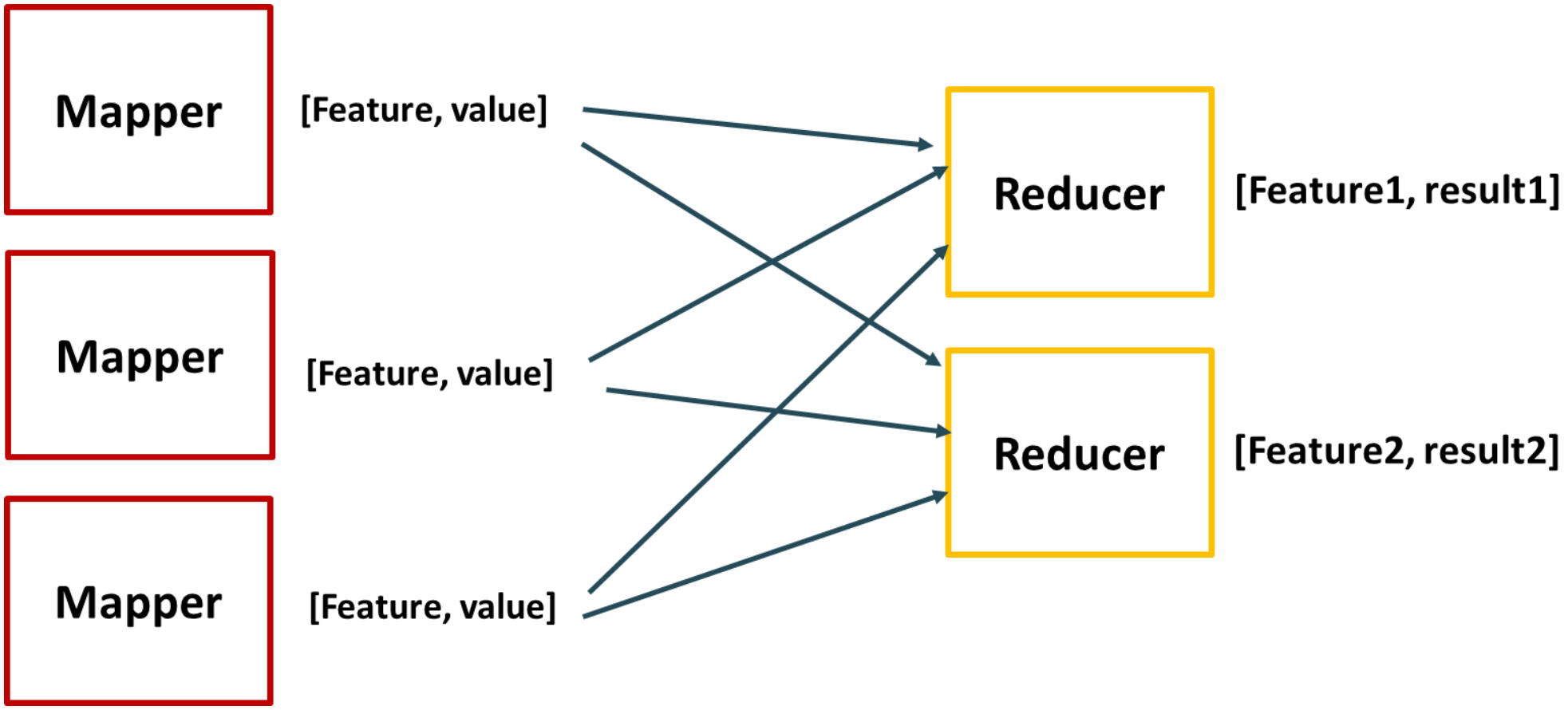
Patrón 4: Resumen numérico

Objetivo: Calcular valores estadísticos agregados sobre un conjunto de datos.

Ejemplos:

- Contador de ocurrencias
- Valores máximos y mínimos
- Promedio / media/ desviación estándar

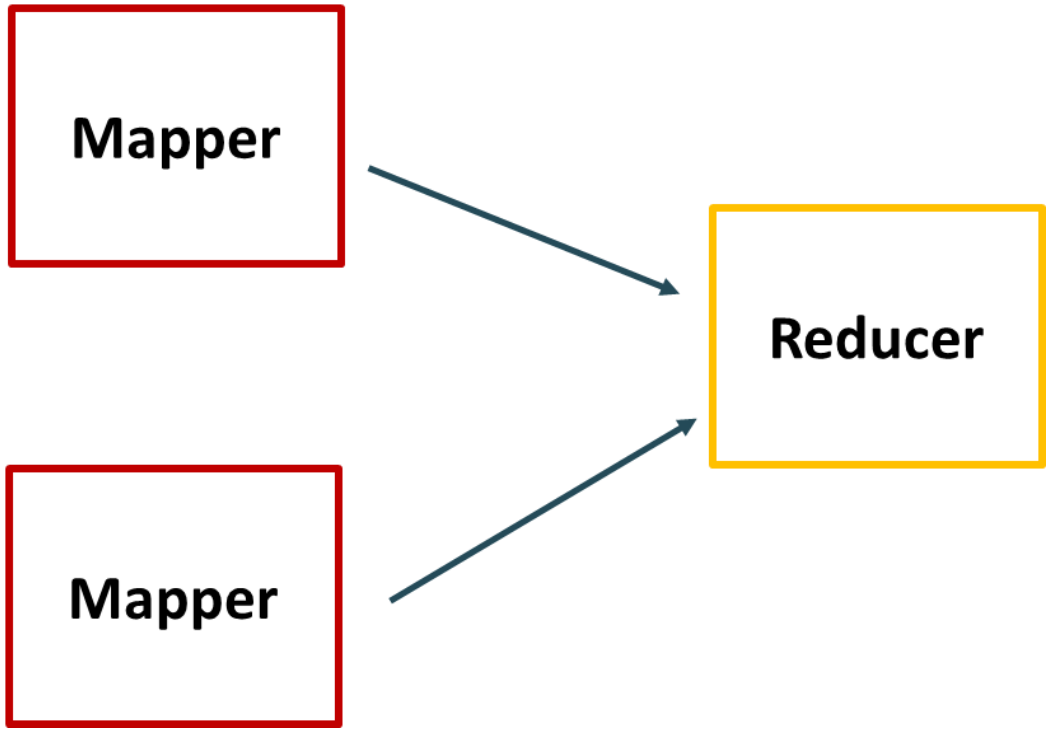
Patrón 4: Resumen numérico



Calculo de promedios

Input: Datos con ID del curso, ID de estudiante, valor
Objetivo: calcular el promedio del cursos

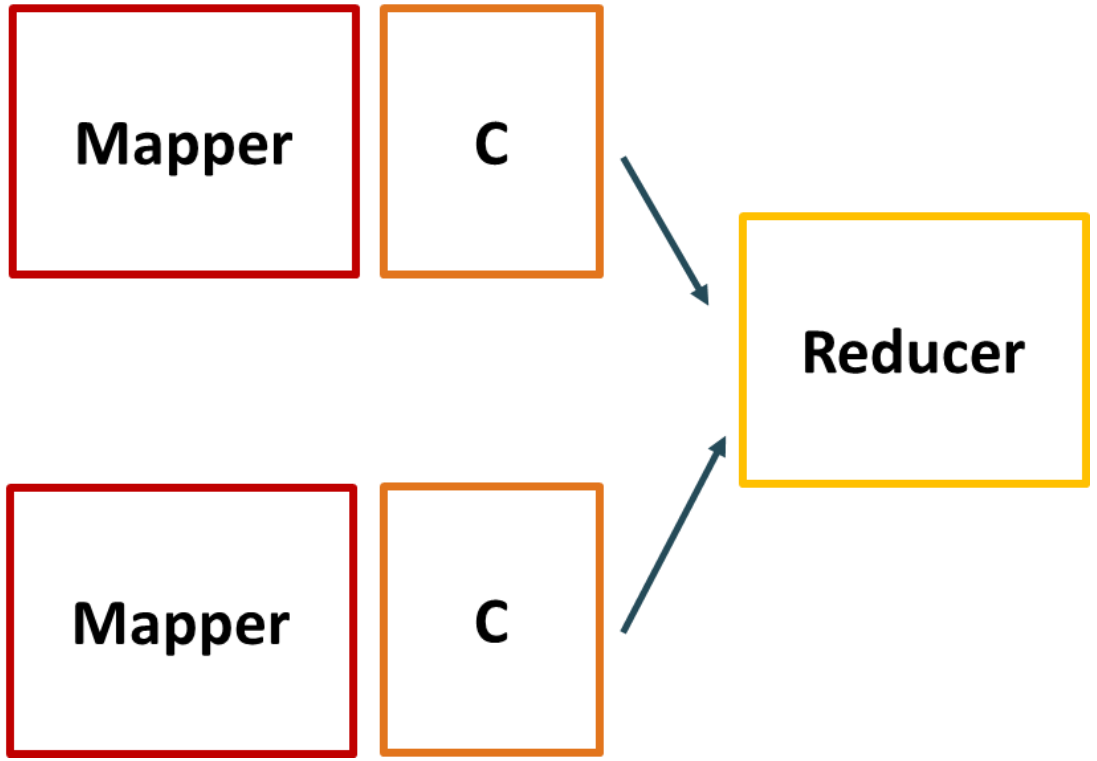
BigDataProc	ec03847293847	100
DataMining	ec29347298347	100
BigDataProc	ec23894283472	100
BigDataProc	ec23489209348	100
BigDataProc	ec23492834343	100
BigDataProc	ec34948758493	0
BigDataProc	ec56456456545	100
BigDataProc	ec73453435434	100



Calculo de promedios ¿Con combiner?

Input: Datos con ID del curso, ID de estudiante, valor
Objetivo: calcular el promedio del cursos

BigDataProc	ec03847293847	100
DataMining	ec29347298347	100
BigDataProc	ec23894283472	100
BigDataProc	ec23489209348	100
BigDataProc	ec23492834343	100
BigDataProc	ec34948758493	0
BigDataProc	ec56456456545	100
BigDataProc	ec73453435434	100



Calculo de promedios

El promedio **NO es una opereación asociativa** y no se puede ejecutar parcialmente con un combinador..

Una solución es cambiar los valores en los pares key/value:

- Funcion Map: emite cantidades y número de elementos
- Función Combiner : produce cantidades agrupadas y número de elementos
- Función Reducer: considera tanto las cantidades agrupadas como el número de elementos para calcular la media final.

Resumiendo

Q: Hoy hemos aprendido sobre MapReduce. MapReduce es un método utilizado para contar palabras en documentos de texto, ¿verdad?

A: MapReduce puede utilizarse para contar palabras en documentos de texto, pero no es un método diseñado específicamente para esa tarea. Es un framework general de procesamiento de datos que puede utilizarse para resolver una amplia variedad de problemas, y el conteo de palabras es solo un ejemplo utilizado para ilustrar

Resumiendo

Q: Cierto, pero los problemas que podemos resolver usando MapReduce parecen un poco... aburridos? Quiero decir, ni siquiera estoy seguro de que calcular un promedio sea ciencia de datos.

A: Recuerda que esto no se trata de la computación que estamos tratando de implementar, sino de ser capaces de implementarla en sí misma. No importa lo simple que sea tu calculo, si tu conjunto de datos es enorme, no podrás procesarlo usando un PC convencional. Luego, te enfrentarás a una situación en la que tu problema no será qué procesar la info, sino cómo asegurarte de que puedas procesarla. Además, antes de pasar a un complejo algoritmo de ciencia de datos que requiera tecnología de big data, es importante comprender algoritmos más simples y útiles.

Resumiendo

Q: Dijiste que MapReduce no se puede utilizar para resolver todos los problemas existentes, ¿por qué no aprendemos algo que sea más general?

A: Mira a MapReduce como un Framework, no como una herramienta. Es una estrategia para expresar la solución para un problema dado. En algunos casos será la mejor opción, en otros no lo será. Es importante entender las fortalezas y limitaciones de MapReduce y aprender a reconocer cuándo puede ser la mejor opción para solucionar un problema en particular

Resumiendo

Q: Ok, De acuerdo, digamos que puedo expresar mi solución utilizando una función map y una función reduce. Pero luego también tengo que hacer otras cosas, partitioning, shuffling, sorting, ¿y cómo hago para que mi cluster funcione? ¿Cómo hablarán entre sí los nodos de mi cluster? ¡Y necesito hacer todo esto para cada problema! Eso es locura.

A: Este es precisamente el punto: esencialmente, deberá preocuparse por la lógica, es decir, la función map y la función de reduce. La plataforma Big Data se encargará de todas las cosas de bajo nivel, por ejemplo, Apache Hadoop, ese es el tema de la próxima semana.
En cualquier caso, incluso si no implementa estas operaciones, debemos comprenderlas.

Adicional

- Leer el capitulo 6 Big Data Fundamentals: Concepts, Drivers & Techniques. Prentice Hall.
- Leer los artículos complementarios en e-aulas