

# PROGRAMACIÓN SOBRE GRANDES VOLUMENES DE DATOS

**Map Reduce**

Magister - Efraín Alberto Oviedo  
eaoc46@gmail.com

**UNIVERSIDAD DE ANTIOQUIA  
FACULTAD DE INGENIERÍA**

**ESPECIALIZACIÓN EN ANALÍTICA Y CIENCIA DE DATOS**



**UNIVERSIDAD  
DE ANTIOQUIA**

1 8 0 3

# AGENDA

## **1. Map Reduce en Spark**

2. Ejemplos

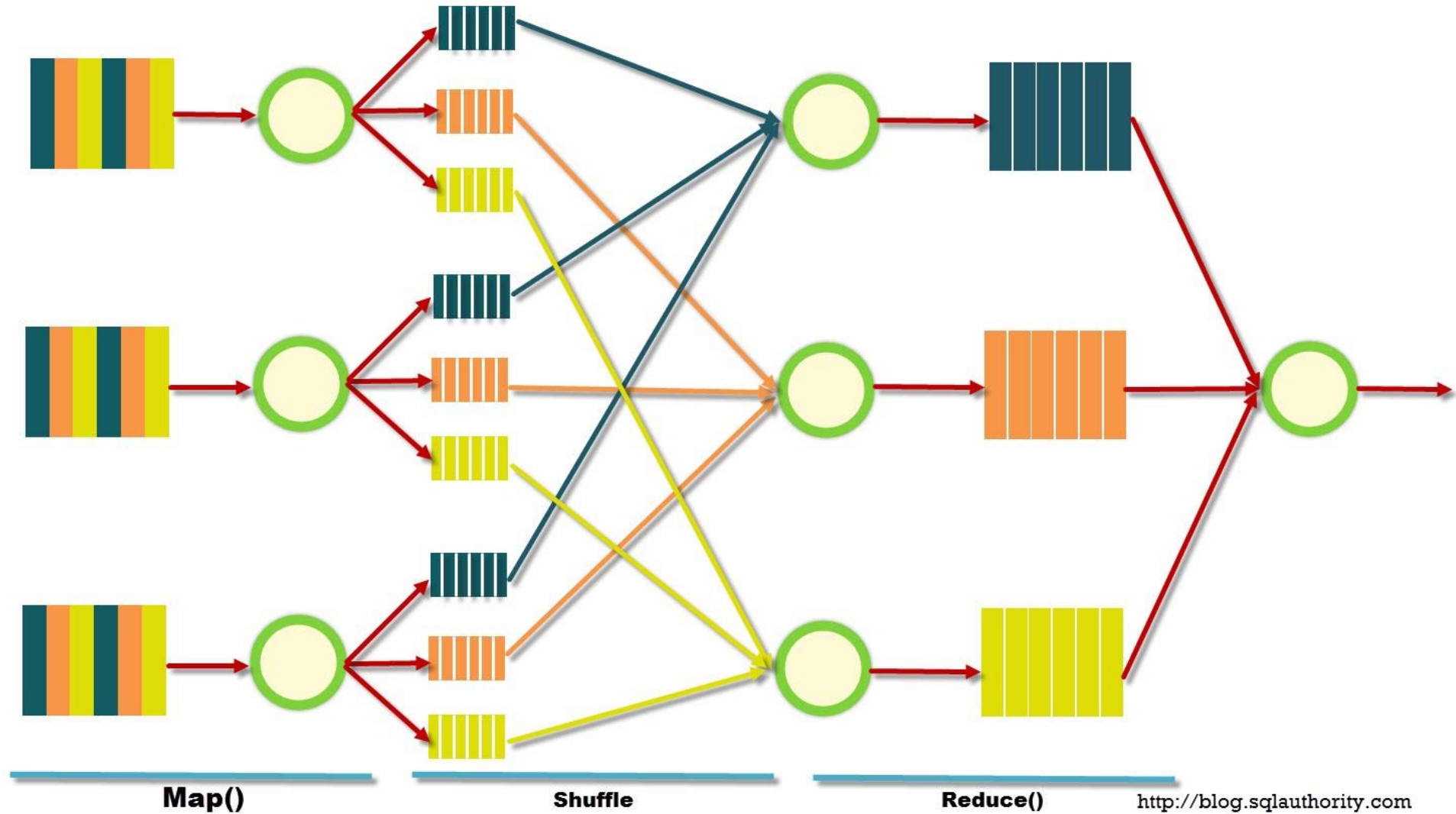
3. Ejercicio

# Qué es Map Reduce?

**Paradigma de programación** que permite trabajar en **ambientes distribuidos** usando una gran cantidad de servidores que conforman un **clúster**.

- **Map:** recibe un conjunto de datos y lo transforma en un segundo conjunto de datos cuyos elementos se representan en tuplas (clave-valor)
- **Reduce:** Utiliza como entrada las tuplas generadas por el map y genera un conjunto de datos de salida reducido a partir de la combinación de los datos recibidos

## How MapReduce Works?

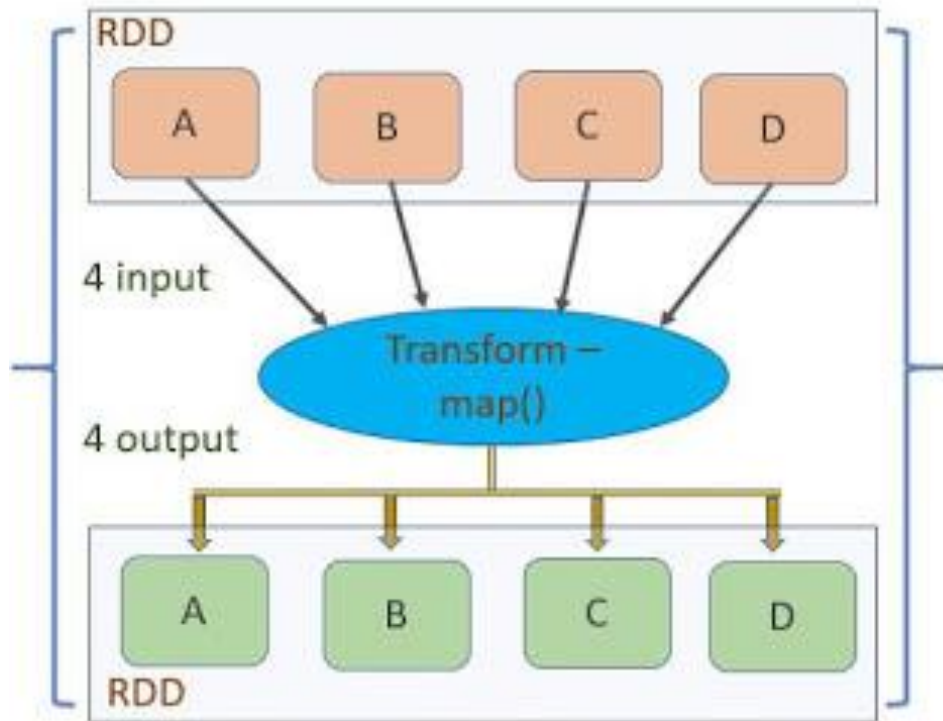



# Map

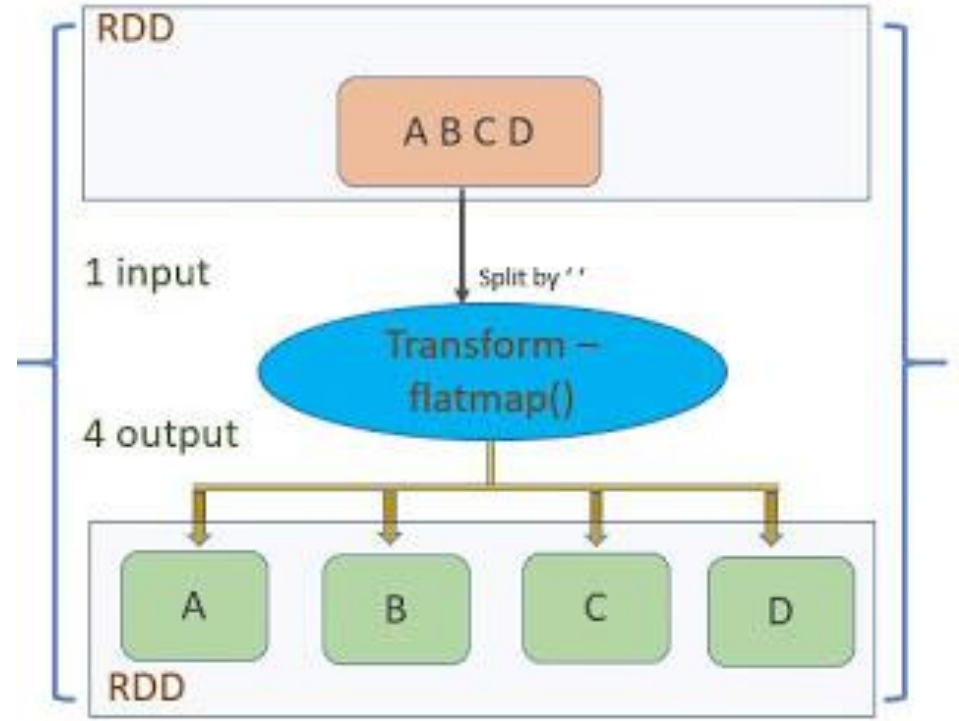
Recibe un conjunto de datos y lo transforma en un segundo conjunto de datos cuyos elementos se representan en tuplas (clave-valor)

Transformaciones	Descripción
<b>map(fcn)</b>	Devuelve un nuevo conjunto de datos distribuido formado al pasar cada elemento de la fuente a través de una función.
<b>flatMap(fcn)</b>	Similar al map, pero cada elemento de entrada se puede asignar a 0 o más elementos de salida. Esto significa que devuelve una secuencia en lugar de un solo elemento.

# Map vs FlatMap



 Spark Map() Operation



 Spark FlatMap() Operation

# Función map

Apliquemos la función map a una secuencia de números

```
num = sc.parallelize([0, 1, 2, 3, 4, 5])
```

```
▶ numMap=num.map(lambda x:(x,2*x))  
numMap.collect()
```

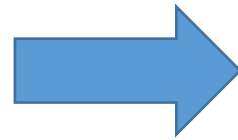
```
📄 [(0, 0), (1, 2), (2, 4), (3, 6), (4, 8), (5, 10)]
```

# Función map

Apliquemos la función map a un archivo de texto

animales.txt X

```
1 Gato Perro Conejo  
2 Conejo Gato Conejo  
3 Perro Conejo Gato
```



```
words = text_file.map(lambda line: line.split(" "))  
words.collect()
```



```
[[ 'Gato', 'Perro', 'Conejo'],  
 [ 'Conejo', 'Gato', 'Conejo'],  
 [ 'Perro', 'Conejo', 'Gato']]
```



# Función flatMap

Apliquemos la función flatMap a una secuencia de números

```
num = sc.parallelize([0, 1, 2, 3, 4, 5])
```



```
numFlatMap=num.flatMap(lambda x:(x,2*x))  
numFlatMap.collect()
```



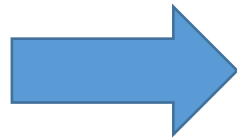
```
[0, 0, 1, 2, 2, 4, 3, 6, 4, 8, 5, 10]
```

# Función flatMap

Apliquemos la función flatMap a un archivo de texto

animales.txt X

```
1 Gato Perro Conejo  
2 Conejo Gato Conejo  
3 Perro Conejo Gato
```



```
▶ text_file = sc.textFile("local/data/animales.txt")  
  words = text_file.flatMap(lambda line: line.split(" "))  
  words.collect()
```

```
↳ ['Gato',  
   'Perro',  
   'Conejo',  
   'Conejo',  
   'Gato',  
   'Conejo',  
   'Perro',  
   'Conejo',  
   'Gato']
```

# Reduce

Utiliza como entrada las tuplas generadas por el map y genera un conjunto de datos de salida reducido a partir de la combinación de los datos recibidos

Transformaciones	Descripción
<b>reduce(fcn)</b>	Reduce los elementos de utilizando el operador binario conmutativo y asociativo especificado
<b>reduceByKey(fcn)</b>	Cuando se llama a un conjunto de datos de pares (K, V), devuelve un conjunto de datos de pares (K, V) donde los valores de cada clave se agregan usando la función de reducción dada, que debe ser de tipo $(V, V) \Rightarrow V$ .

# Función reduce

Apliquemos la función reduce a una secuencia de números

```
num = sc.parallelize([0, 1, 2, 3, 4, 5])
```

```
▶ numReduce=num.reduce(lambda a,b:a+b)  
print("La suma es: ", numReduce)
```

```
↳ La suma es: 15
```

```
▶ numReduce=num.reduce(lambda a,b: a if a<b else b)  
print("El menor es: ", numReduce)
```

```
↳ El menor es: 0
```

# Función reduceByKey

Apliquemos la función reduceByKey a tuplas de números

```
tuplas = sc.parallelize([[0, 1], [0, 2], [0, 3], [1, 5], [1, 6], [2, 9]])
```



```
tuplasReduce=tuplas.reduceByKey(lambda a,b:a+b)  
tuplasReduce.collect()
```



```
[(0, 6), (2, 9), (1, 11)]
```



```
tuplasReduce=tuplas.reduceByKey(lambda a,b: a if a<b else b)  
tuplasReduce.collect()
```



```
[(0, 1), (2, 9), (1, 5)]
```

# AGENDA

1. Map Reduce en Spark
- 2. Ejemplos**
3. Ejercicio

# Ejemplo - Fibonacci

## Obtener el cuadrado de los elementos de la sucesión de Fibonacci

```
[6] fibo = sc.parallelize([0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89])  
    fibo.collect()
```

```
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
```

Ahora utilizemos una función map que nos permite tomar cada elemento de la sucesión y multiplicarlo por sí mismo. El resultado es un nuevo conjunto de datos que contiene el cuadrado de los elementos iniciales

```
[7] fibo2 = fibo.map(lambda x: x*x)  
    fibo2.collect()
```

```
[0, 1, 1, 4, 9, 25, 64, 169, 441, 1156, 3025, 7921]
```

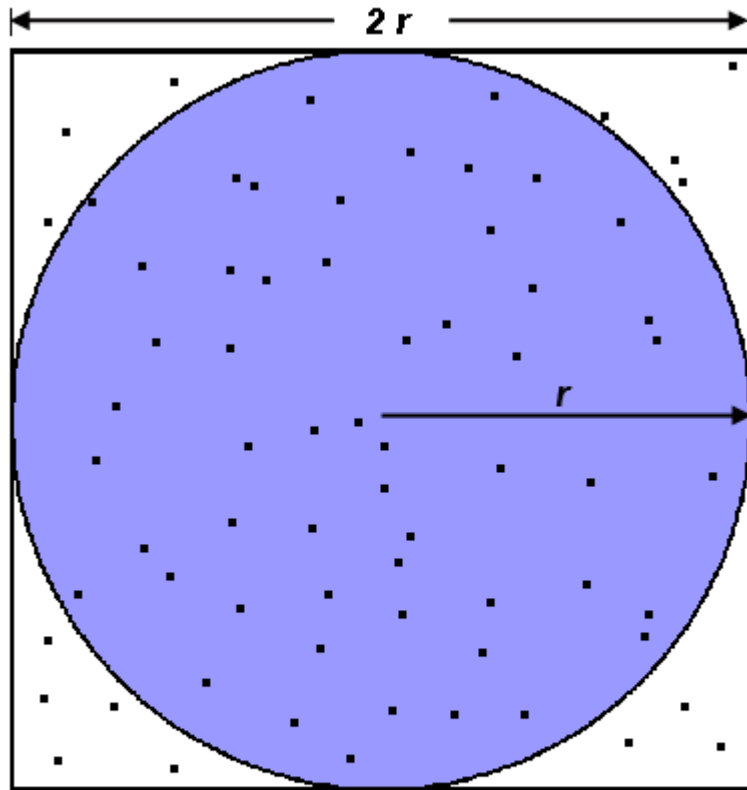
Ahora utilizemos la función reduce para sumar el resultado que obtuvimos en la función map anterior

```
[8] from operator import add  
    fibo2Sum = fibo2.reduce(add)  
    print ("La suma del cuadrado de los elementos de la sucesión es: ", fibo2Sum)
```

```
La suma del cuadrado de los elementos de la sucesión es: 12816
```

# Ejemplo – Calcular el número PI

Calculemos el número PI utilizando el método de Monte-Carlo



$$A_S = (2r)^2 = 4r^2$$

$$A_C = \pi r^2$$

$$\pi = 4 \times \frac{A_C}{A_S}$$

- Generamos puntos aleatorios en el cuadrado unitario (0,0) a (1,1)
- Contamos cuantos de esos puntos generados están dentro del círculo unitario
- El total de puntos en el círculo unitario dividido el total de puntos generados es aproximadamente  $\pi/4$



# Ejemplo – Calcular el número PI

Ejemplo: Calculemos el número PI utilizando el método de Monte-Carlo

```
[38] def func(p):  
    #Creamos aleatoriamente una cooredenada x,y  
    x, y = np.random.random(), np.random.random()  
    #Devolvemos 1 si la coordenada hace parte del círculo y 0 de lo contrario  
    return 1 if x*x + y*y < 1 else 0
```

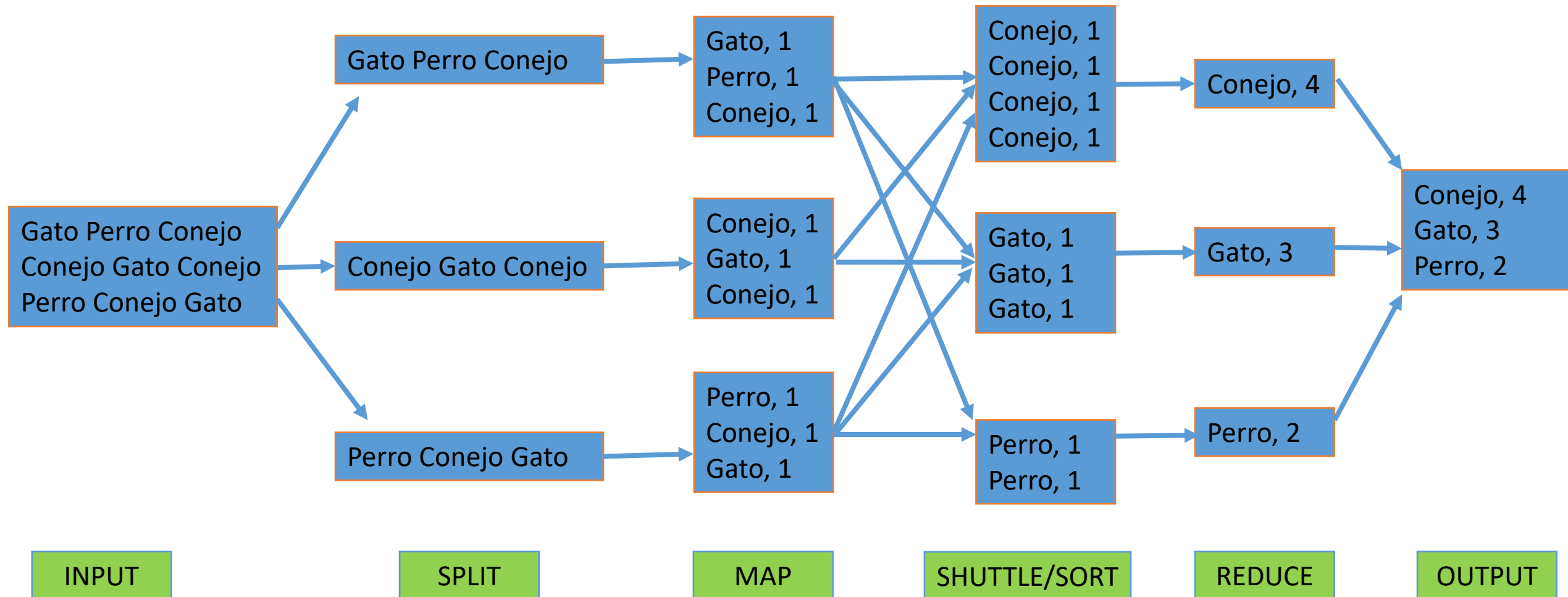
```
[39] n = 1000000  
    count = sc.parallelize(range(0, n)).map(func) \  
        .reduce(lambda a, b: a + b)
```

```
[40] #pi= 4*r  
    r = float(count) / float(n)  
    print ("El valor de PI es aproximadamente %f" % (4.0 * r))
```

📄 El valor de PI es aproximadamente 3.142432

# Ejemplo – Contador de Palabras

## Contador de Palabras



# Ejemplo – Contador de Palabras

```
[10] texto = sc.parallelize(['gato', 'perro', 'conejo', 'conejo', 'gato', 'conejo', 'perro', 'conejo', 'gato'])
      texto.collect()
```

Con la ayuda de la función map generamos la tupla (palabra,1) para cada una de las palabras ingresadas

```
[11] cont = texto.map(lambda x: (x,1))
      cont.collect()
```

Con la ayuda de la función reduceByKey tomamos todas las tuplas que contengan la misma clave, en este caso la misma palabra, y sumamos sus valores, es decir las veces que aparece cada palabra

```
[12] res = cont.reduceByKey(lambda x, y: x + y)
      print("Las siguientes tuplas nos muestran cuantas veces aparece cada palabra en el texto ingresado")
      res.collect()
```

Las siguientes tuplas nos muestran cuantas veces aparece cada palabra en el texto ingresado

```
[('perro', 2), ('conejo', 4), ('gato', 3)]
```

# Ejemplo – Contador de Palabras

## Contemos palabras en un archivo de Texto

### ▼ Leyendo un archivo de texto

- Leemos el archivo de texto
- Aplicamos la función flatMap para obtener una variable con todas las palabras
- Aplicamos la función map para genera la tupla (palabra,1)
- Aplicamos la función reduceByKey para sumar los valores de las tuplas que contienen la misma palabra



```
text_file = sc.textFile("local/data/animales.txt")
counts = text_file.flatMap(lambda line: line.split(" ")) \
    .map(lambda word: (word, 1)) \
    .reduceByKey(lambda a, b: a + b)
counts.collect()
```

```
[('Gato', 3), ('Perro', 2), ('Conejo', 4)]
```

# Ejemplo – Temperatura Máxima por ciudad

Ejemplo: Temperatura máxima de cada ciudad

```
temperatura.txt: Bloc de notas
Archivo Edición Formato Ver Ayuda
META,34.3
AMAZONAS,23.3
META,24.1
CALDAS,4.1
META,33.4
CAUCA,20.4
CUNDINAMARCA,5.5
CALDAS,18.1
ANTIOQUIA,29.9
CUNDINAMARCA,17
BOYACÁ,19.6
CAUCA,7.4
TOLIMA,11
CALDAS,13.1
PUTUMAYO,27.8
CUNDINAMARCA,13
ANTIOQUIA,25.1
SANTANDER,8.8
CUNDINAMARCA,10.6
.....
```



```
('AMAZONAS', 48.6),
('BOYACÁ', 28.7),
('TOLIMA', 33.5),
('PUTUMAYO', 28.9),
('SUCRE', 35.0),
('CESAR', 35.0),
('MAGDALENA', 36.6),
('CÓRDOBA', 37.4),
('NARIÑO', 25.7),
('ATLÁNTICO', 36.5),
('RISARALDA', 21.3),
('CHOCÓ', 29.8),
('CORDOBA', 33.3),
.....
```

# Ejemplo – Temperatura Máxima por ciudad

En el archivo temperatura.txt tenemos el departamento y la temperatura separados por coma. Aplicamos un split para separar ambos valores mediante una función map

```
[19] data1=text_file.map(lambda a: (a.split(",")))  
     data1.take(10)
```

Usamos nuevamente una función map para generar la tupla (departamento, temperatura)

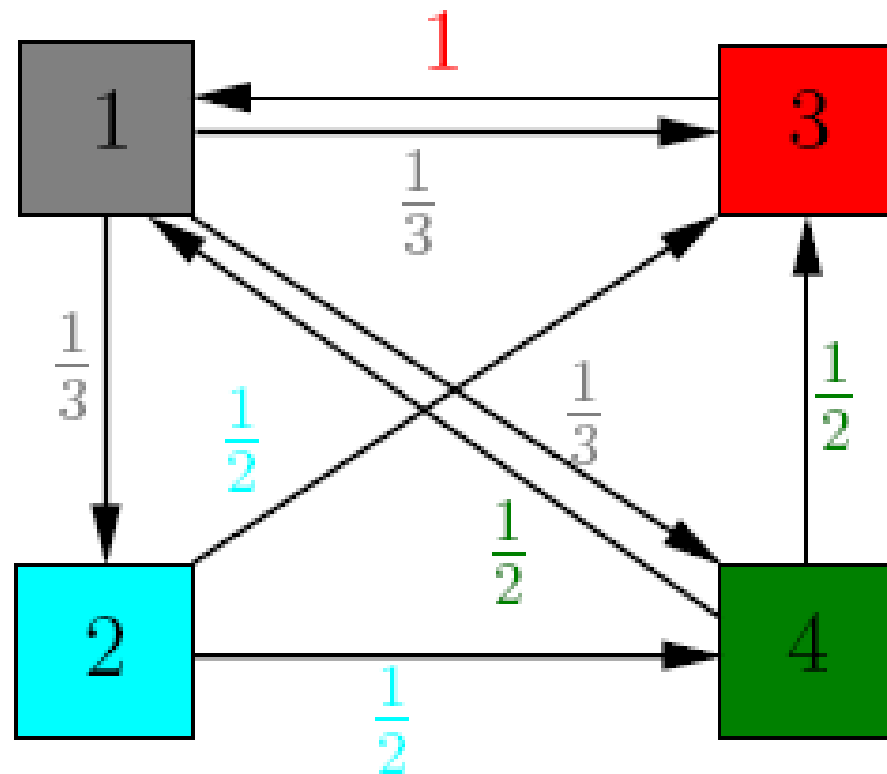
```
[20] data2=data1.map(lambda word: (word[0], float (word[1]) ) )  
     data2.take(10)
```

Utilizamos la función reduceByKey para encontrar el valor máximo de temperatura en cada grupo de tuplas que contengan el mismo departamento

```
[21] data3=data2.reduceByKey(lambda a,b: max(a,b))  
     print("A continuación se presenta la temperatura máxima de cada departamento")  
     data3.collect()
```

# Ejemplo – Page Rank

PageRank es un método que propone Google para evaluar la autoridad que tiene un sitio web en internet, esa autoridad se calcula evaluando la cantidad y calidad de enlaces provenientes de otros sitios web.



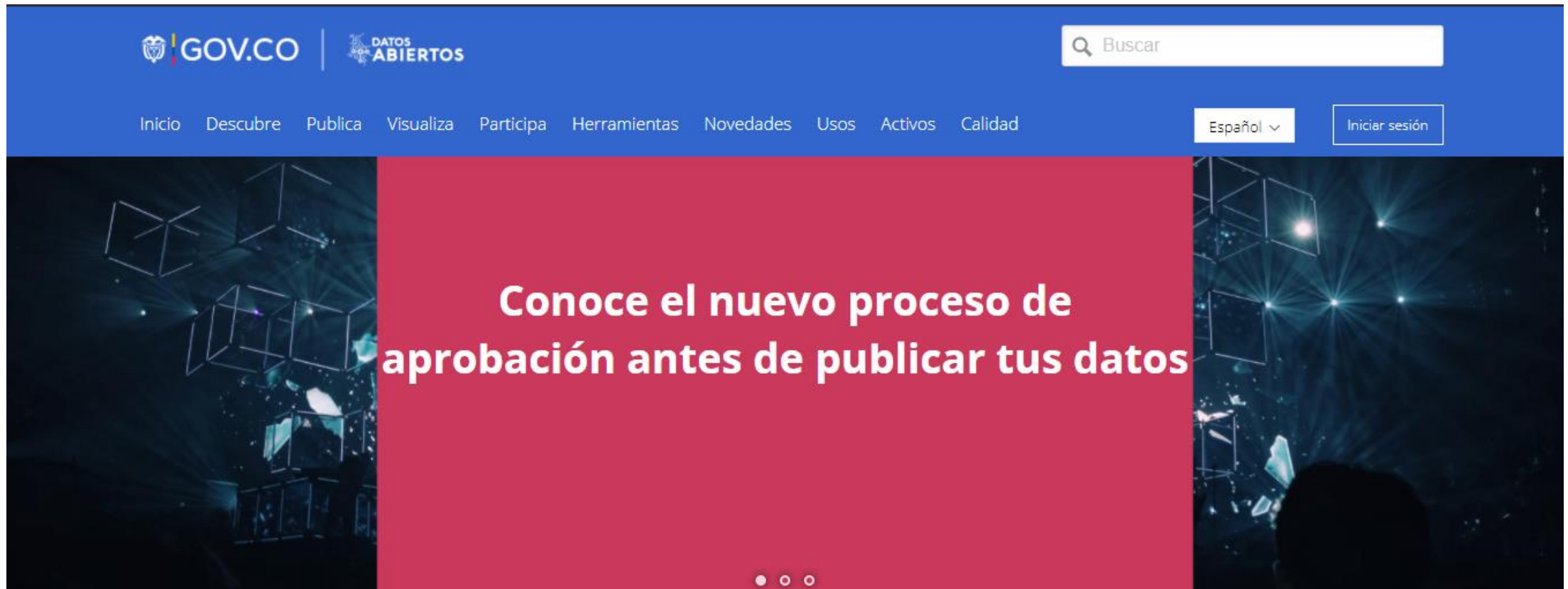
# AGENDA

1. Map Reduce en Spark
2. Ejemplos
- 3. Ejercicio**



# Ejercicio

Analicemos las Cifras del Covid19 en Colombia obtenidas de <https://www.datos.gov.co/>



# Covid19 en Colombia

Estos datos relacionan los casos positivos en Colombia a través de las siguientes variables

- Ciudad
- Departamento
- Atención (casa, fallecido, hospital, hospital/uci, recuperado)
- Edad
- Tipo (En estudio, importado, relacionado)
- Estado (Asintomático, Fallecido, Grave, Leve, Moderado)

# Datos disponibles

Id	Ciudad	Departamento	Atención	Edad	Sexo	Tipo	Estado
1	Bogota	Bogota	Recuperado	19	F	Importado	Leve
2	Guadalajara de Buga	Valle del Cauca	Recuperado	34	M	Importado	Leve
3	Medellin	Antioquia	Recuperado	50	F	Importado	Leve
4	Medellin	Antioquia	Recuperado	55	M	Relacionado	Leve
5	Medellin	Antioquia	Recuperado	25	M	Relacionado	Leve
6	Itagui	Antioquia	Recuperado	27	F	Relacionado	Leve
7	Cartagena de Indias	Cartagena D.T. y C.	Recuperado	85	F	Importado	Leve
8	Bogota	Bogota	Recuperado	22	F	Importado	Leve
9	Bogota	Bogota	Recuperado	28	F	Importado	Leve
10	Bogota	Bogota	Recuperado	36	F	Importado	Leve