UNIVERSIDAD COMPLUTENSE MADRID

Características avanzadas de Apache Spark

Enrique Martín (emartinm@ucm.es)
Sistemas de Gestión de Datos y de la Información
Master Ing. Informática
Fac. Informática

Contenidos

Acumuladores

Acumuladores

- Variable creada en el `SparkContext` y que se comparte entre todos los *workers*
- Los *workers* solo pueden incrementarlo (`accum += X`)
- El *driver* sí puede consultar su contenido
- Usados principalmente para depuración, p. ej. contar el número de líneas corruptas detectadas en todos los *workers* sin requerir varias transformaciones o RDDs adicionales

Acumuladores en pySpark

```
nerr = sc.accumulator(0)
# Acumulador como variable global

def parse(line, acc):
  if len(line) < 5:
    acc += 1  # Error detectado -> incrementa acumulador
    return []
  else:
    return line.split()


lines = sc.textFile('fichero.txt')
result = lines.map(lambda x: parse(x,nerr))
result.count()  # Lanza el cómputo en el RDD
print('#Errors: ', nerr)
```

Variables *broadcast*

Variables *broadcast*

- Los datos usados en las transformaciones se **codifican** y **transmiten** a cada una de las tareas ejecutadas en los *workers*

```
# users es una lista de varios cientos de MB
rdd.filter(lambda x: x in users)
```

- Un *worker* puede ejecutar varias tareas, así que recibirá varias veces esos datos. Esto genera un desperdicio de ancho de banda y de tiempo de transmisión

Variables *broadcast*

- Las variables *broadcast* sirven para definir datos de **solo lectura** que se compartirán de manera eficiente:
  - se enviarán **una sola vez** a cada *worker*, quedarán almacenadas y se podrán reutilizar
  - se enviarán usando un **protocolo optimizado** para transmitir **datos grandes**

Variables *broadcast* en pySpark

```
stopwords = ['a', 'the', 'an', 'to', ...]
stopwords_bc = sc.broadcast(stopwords)
# La variable de broadcast es global

def valid_word(word):
  return word not in stopwords_bc.value
  # Se accede al contenido de la variable de broadcast
  # a través del atributo 'value'


lines = sc.parallelize(['a', 'hello', 'a'])
lines.filter(valid_word).collect()
```