HELLENIC REPUBLIC
UNIVERSITY OF IOANNINA

Παραδείγματα επίλυσης προβλημάτων με το υπολογιστικό μοντέλο Map Reduce

Χρήστος Γκόγκος

29/10/2019

https://github.com/chgogos/big_data

Προσέγγιση προβλημάτων με το MapReduce

- Το υπολογιστικό μοντέλο MapReduce (MR) αποτελεί έναν αποδοτικό τρόπο για την παράλληλη επεξεργασία μεγάλων δεδομένων.
- Στις ακόλουθες διαφάνειες θα παρουσιαστούν διάφορα προβλήματα στα οποία επιδεικνύεται η λειτουργία του MR.
- Θα χρησιμοποιηθεί η python βιβλιοθήκη MapReduce.py που υλοποιεί το υπολογιστικό μοντέλο MR, αλλά η εκτέλεση θα γίνεται εξ' ολοκλήρου σε έναν απλό Η/Υ.
- Η βιβλιοθήκη MapReduce.py έχει αναπτυχθεί από τον Bill Howe του Univ. of Washington στα πλαίσια του specialization "Data Science at Scale" που προσφέρεται ως MOOC μέσω του Coursera.

Είσοδος

- Σε όλα τα παραδείγματα τα δεδομένα βρίσκονται σε json αρχεία που έχουν παρόμοια μορφή με την ακόλουθη:

```
["text1","aaa bbb ccc"]
["text2","aaa bbb eee"]
["text3","aaa bbb fff"]
["text4","aaa eee fff"]
```

text1.json

Το βασικό template κώδικα

```
import MapReduce
import sys

"""
Word Count Example
"""
mr = MapReduce.MapReduce()
```

```
def mapper(record):
    # key: document identifier
    # value: document contents
    key = record[0]
    value = record[1]
    words = value.split()
    for w in words:
      mr.emit_intermediate(w, 1)
```

```
def reducer(key, list_of_values):
    # key: word
    # list_of_values: list of occurrence counts
    total = 0
    for v in list_of_values:
      total += v
    mr.emit((key, total))
```

```
if __name__ == '__main__':
  inputdata = open(sys.argv[1])
  mr.execute(inputdata, mapper, reducer)
```

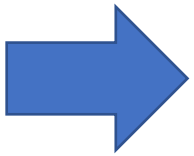
mr0.py

Παράδειγμα 0: Καταμέτρηση λέξεων

$ python mr0.py text1.json

```
["text1","aaa bbb ccc"]
["text2","aaa bbb eee"]
["text3","aaa bbb fff"]
["text4","aaa eee fff"]
```

text1.json



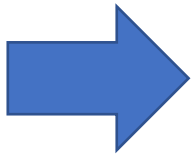
```
["eee", 2]
["fff", 2]
["aaa", 4]
["bbb", 3]
["ccc", 1]
```

Παράδειγμα 1: Ανεστραμμένο ευρετήριο

- Δεδομένου ενός συνόλου εγγράφων, ένα ανεστραμμένο ευρετήριο (inverted index) είναι ένα λεξικό στο οποίο κάθε λέξη συσχετίζεται με μια λίστα αναγνωριστικών εγγράφων στα οποία η λέξη υπάρχει.

$ python mr1.py text1.json

```
["text1","aaa bbb ccc"]
["text2","aaa bbb eee"]
["text3","aaa bbb fff"]
["text4","aaa eee fff"]
```



text1.json

```
["eee", ["text2", "text4"]]
["fff", ["text3", "text4"]]
["aaa", ["text1", "text2", "text3", "text4"]]
["bbb", ["text1", "text2", "text3"]]
["ccc", ["text1"]]
```

Λύση παραδείγματος 1

```
def mapper(record):
    key = record[0]
    value = record[1]
    words = value.split()
    words = list(set(words))
    for w in words:
      mr.emit_intermediate(w, key)

def reducer(key, list_of_values):
    mr.emit((key, list_of_values))
```

mr1.py

Παράδειγμα 2: Join πινάκων

- Ο κώδικας MR θα πρέπει να παράγει το ίδιο αποτέλεσμα με την ακόλουθη SQL εντολή:

```
SELECT *
FROM Orders, LineItems
WHERE Orders.order_id = LineItems.order_id;
```

- Τα δεδομένα εισόδου αποτελούνται από ένα αρχείο που περιέχει αρχικά τις εγγραφές order και στη συνέχεια τις εγγραφές line_item

```
["order", "1", "36901", "O", "173665.47", "1996-01-02", ...]
["order", "2", "78002", "O", "46929.18", "1996-12-01", ...]
...
["line_item", "1", "155190", "7706", "1", "17", ...]
["line_item", "1", "67310", "7311", "2", "36", ...]
...
["line_item", "2", "106170", "1191", "1", "38", ...]
["line_item", "3", "4297", "1798", "1", "45", ...]
...
```

records.json

Λύση παραδείγματος 2

```
def mapper(record):
    key = record[1]
    mr.emit_intermediate(key, record)

def reducer(key, list_of_values):
    row = []
    for record in list_of_values:
        if record[0]=='order':
            row=record
        else:
            mr.emit(row+record)
```

mr2.py

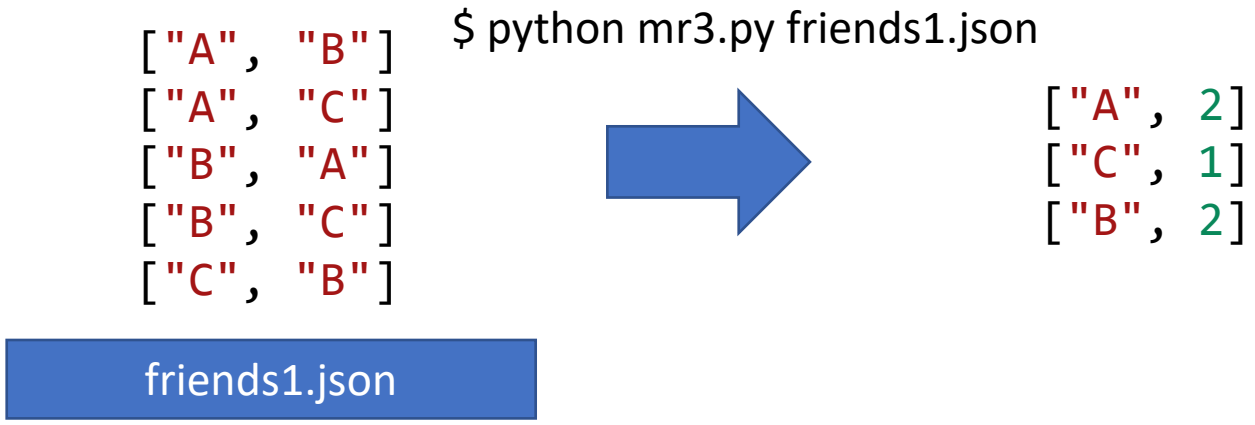
```
["order", "1", "36901", "O", "173665.47", "1996-01-02", ...]
["order", "2", "78002", "O", "46929.18", "1996-12-01", ...]
...
["line_item", "1", "155190", "7706", "1", "17", ...]
["line_item", "1", "67310", "7311", "2", "36", ...]
...
["line_item", "2", "106170", "1191", "1", "38", ...]
["line_item", "3", "4297", "1798", "1", "45", ...]
...
```

$ python mr2.py records.json

```
["order", "1", "36901", "O", "173665.47", "1996-01-02", ...,
"line_item", "1", "155190", "7706", "1", "17",...],
...
```

Παράδειγμα 3: Πλήθος φίλων

- Θεωρείστε ένα απλό σύνολο δεδομένων που περιγράφει σχέσεις φιλίας σε ένα κοινωνικό δίκτυο. Για παράδειγμα η εγγραφή [“A”,“C”] σημαίνει ότι ο A θεωρεί ότι ο C είναι φίλος του. Ο κώδικας MR θα πρέπει να παράγει το πλήθος των φίλων κάθε ατόμου.



Λύση παραδείγματος 3

```
def mapper(record):
    key = record[0]
    mr.emit_intermediate(key, 1)

def reducer(key, list_of_values):
    total = 0
    for v in list_of_values:
      total += v
    mr.emit((key, total))
```

mr3.py

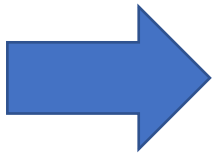
Παράδειγμα 4: Ασύμμετρες σχέσεις φιλίας

- Για τα δεδομένα του παραδείγματος 3 να βρεθούν όλες οι ασύμμετρες σχέσεις δηλαδή οι σχέσεις στις οποίες ενώ ο Α θεωρεί ότι ο C είναι φίλος του, δεν ισχύει το αντίστροφο. Δηλαδή, ενώ υπάρχει καταγεγραμμένη η πληροφορία [“A”,“C”] δεν υπάρχει καταγεγραμμένη η πληροφορία [“C”,“A”].

```
["A", "B"]
["A", "C"]
["B", "A"]
["B", "C"]
["C", "B"]
```

friends1.json

$ python mr4.py friends1.json



"For A relationship with C is asymmetric"
"For C relationship with A is asymmetric"

Λύση παραδείγματος 4

```
def mapper(record):
    person = record[0]
    friend = record[1]
    mr.emit_intermediate(person, record)
    mr.emit_intermediate(friend, record)

def reducer(key, list_of_values):
    for v in list_of_values:
        if [v[1],v[0]] not in list_of_values:
          if (v[0] == key):
            mr.emit("For %s relationship with %s is asymmetric" % (v[0],v[1]))
          else:
            mr.emit("For %s relationship with %s is asymmetric" % (v[1],v[0]))
```

mr4.py

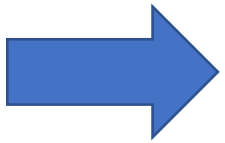
Παράδειγμα 5: Ακολουθίες DNA

- Θεωρείστε ένα σύνολο από ζεύγη κλειδί-τιμή κάθε κλειδί είναι ένα αναγνωριστικό ακολουθίας και κάθε τιμή είναι μια ακολουθία νουκλεοτιδίων (π.χ. GCTTCCGAAA…). Γράψτε MR κώδικα που αφαιρεί τους τελευταίους 10 χαρακτήρες από κάθε ακολουθία νουκλοτιδίων και στη συνέχεια αφαιρεί τυχόν διπλότυπα που θα προκύψουν.

```
["001", "GGGGTGGCTACC"]
["002", "AAGAACAACCTT"]
["003", "AAGATCAACCTT"]
["004", "GGGATCAACCTT"]
["005", "GAGATCAACCCC"]
```

dna1.json

$ python mr5.py dna1.json



"GG"
"AA"
"GA"

Λύση παραδείγματος 5

```
def mapper(record):
  sequence = record[1][:-10]
  mr.emit_intermediate(sequence, 1)

def reducer(key, _):
  mr.emit(key)
```

mr5.py

Παράδειγμα 6: Πολλαπλασιασμός δισδιάστατων πινάκων

- Θεωρείστε ότι διαθέτετε δεδομένα δύο πινάκων A(L x M) και B (M x N) σε μορφή αραιού πίνακα (κάθε μη μηδενική τιμή καταγράφεται ως τριάδα: <αριθμός σειράς>,<αριθμός στήλης>,<τιμή>).
- Γράψτε MR κώδικα που να πολλαπλασιάζει τους δύο πίνακες.

- Τα δεδομένα δίνονται ως ένα ενιαίο αρχείο κειμένου με εγγραφές όπου το πρώτο στοιχείο αναγνωρίζει σε ποιον πίνακα (A ή B) ανήκει η εγγραφή.

```
["a", 0, 0, 2]
["a", 0, 2, 1]
["a", 1, 0, 4]
["a", 1, 2, 5]
["b", 0, 0, 1]
["b", 0, 1, 7]
["b", 2, 0, 3]
```

matrix1.json

a

2	0	1
4	0	5

2 x 3

b

1	7
0	0
3	0

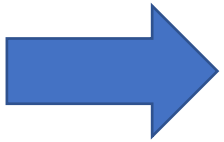
3 x 2

Ένας reducer ανά κελί του πίνακα a x b

```
["a", 0, 0, 2]
["a", 0, 2, 1]
["a", 1, 0, 4]
["a", 1, 2, 5]
["b", 0, 0, 1]
["b", 0, 1, 7]
["b", 2, 0, 3]
```

matrix1.json

$ python mr6.py matrix1.json



```
[[0, 1], 14]
[[1, 0], 19]
[[0, 0], 5]
[[1, 1], 28]
```

a

2	0	1
4	0	5

x

b

1	7
0	0
3	0

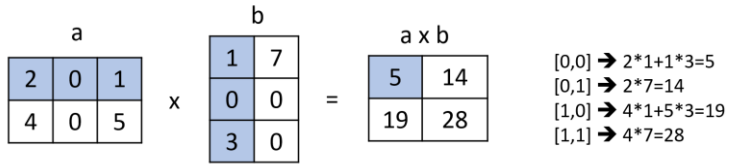
=

a x b

5	14
19	28

[0,0] ➔ 2*1+1*3=5
[0,1] ➔ 2*7=14
[1,0] ➔ 4*1+5*3=19
[1,1] ➔ 4*7=28

Λύση παραδείγματος 6




```
L = 2
M = 3
N = 2

def mapper(record):
    i = record[1]
    j = record[2]
    v = record[3]
    if record[0] == "a":
        for k in range(N):
            mr.emit_intermediate((i, k), ('a', i, j, v))

    if record[0] == "b":
        for k in range(L):
            mr.emit_intermediate((k, j), ('b', i, j, v))
```

```
def reducer(key, list_of_values):
    sum = 0
    for v1 in list_of_values:
        if v1[0] == "a":
            for v2 in list_of_values:
                if v2[0] == "b" and v1[2] == v2[1]:
                    sum += v1[3]*v2[3]
    mr.emit((key, sum))
```

mr6.py

```
[[0, 1], [["a", 0, 0, 2], ["a", 0, 2, 1],
["b", 0, 1, 7]]]
[[1, 0], [["a", 1, 0, 4], ["a", 1, 2, 5],
["b", 0, 0, 1], ["b", 2, 0, 3]]]
[[0, 0], [["a", 0, 0, 2], ["a", 0, 2, 1],
["b", 0, 0, 1], ["b", 2, 0, 3]]]
[[1, 1], [["a", 1, 0, 4], ["a", 1, 2, 5],
["b", 0, 1, 7]]]
```