Week 6
(Local) MapReduce with Ngram Count

Jason S. Chang 張俊盛

2019 0326

MapReduce according to Wikipedia

- MapReduce is a parallel, distributed model implemented on a cluster for processing and generating big data sets.
  - "MapReduce System"
    - marshalling the distributed servers
    - running the various tasks (mapper and reducer) in parallel
    - managing all communications and data transfers between tasks
    - providing for redundancy and fault tolerance
  - mapper program, which performs filtering and sorting (such as sorting students by first name into queues, one queue for each name)
  - reduce method, which performs a summary operation (such as counting the number of students in each queue, yielding name frequencies).
- key contributions: scalability and fault-tolerance by multi-threaded implementations on multi-processors with optimized shuffle operation

More about MapReduce

- Open source tools
  - Hadoop (flat text files)
  - Pig (SQL files and operations)
  - Apache Hive
  - Local MapReduce (invented here for this course)
- Use cases
  - word count
  - sorting
  - constructing inverted file for Web search engines
  - document clustering
  - machine learning

Local MapReduce and Examples

- See `github.com/dspp779/local-mapreduce`
- Usage

```
./lmr <chunk size> <#reducer> <mapper> <reducer> <directory>
```

  - <chunk size>: Split data into chunks with <chunk size>
  - <#reducer>: Each output line from mappers would then be hashed into <num of reducer> different reducer
  - <mapper>, <reducer>: Shell command/Python program
  - <directory>: The output directory

Local MapReduce–Word Count

- Mapper and Reducer

```
tr -sc "a-zA-Z" "\n"   (s = Squeeze; c = Complement)
uniq -c                (c = add Count)
```

- Testing mapper

```
$ echo 'Colorless green ideas \n sleep furiously. Colorless green ideas' | tr -sc "a
Colorless
green
ideas
sleep
furiously
Colorless
green
ideas
```

- Testing reducer

```
$ echo $'Colorless green ideas \n sleep furiously' | tr -sc "a-zA-Z" "\n"
| sort | uniq -c
   2 Colorless
   2 furiously
   2 green
   2 ideas
   1 sleep
   1 furiously
```

Ngram Count

- Mapper

```
import re, sys

def tokens(str1): return re.findall('[a-z]+', str1.lower())
def ngrams(sent, n):
    return [ ' '.join(x) for x in zip(*[sent[i:] for i in range(n)
        if i <= len(sent) ] ) ]

for line in sys.stdin:
    sent = tokens(line)
    for n in range(2, 6):
        for ngram in ngrams(sent, n):
            print ('%s\t%s' % (ngram, 1))
```

- Testing mapper

```
echo $'Colorless green ideas \n sleep furiously' | python nc-mapper.py

colorless green 1
green ideas 1
colorless green ideas 1
sleep furiously 1
```

- Reducer

```
import sys
from collections import Counter, defaultdict

ngm_count = defaultdict(Counter)
for line in sys.stdin:
    ngm, count = line.split('\t'); n = ngm.count(' ')+1
    ngm_count[n][ngm] += int(count)

for n in range(2, 6):
    for ngm in ngm_count[n]:
        if ngm_count[n][ngm] >= 3:
            print( '%s\t%s' % (ngm, ngm_count[n][ngm]) )
```

- Testing reducer

```
echo $'Colorless green ideas \n sleep furiously' | python nc-mapper.py
```

```
| sort | python nc-reducer.py

colorless green 1
green ideas 1
sleep furiously 1
colorless green ideas 1
```

- Running local MapReduce

```
echo $'Colorless green ideas \n sleep furiously'
 | ./lmr 5m 16  'python nc-mapper.py' 'python nc-reducer.py' out

hashing script hashing.py.BWar
 >>> Temporary output directory for mapper created: mapper_tmp.YZ4i
 >>> Mappers running...
 >>> Reducer running. Temporary input directory: mapper_tmp.YZ4i
 >>> Cleaning...
 >>> Temporary directory deleted: mapper_tmp.YZ4i
 * Output directory: out
 * Elasped time: 0:00:02

$ cat out/*
sleep furiously 1
colorless green ideas 1
colorless green 1
green ideas 1
```

- Life-size Test on British National Corpus

```
$ time cat bnc.sent.txt | python nc-mapper.py | sort | python nc-reducer.py 3 > bnc

$ grep '^ability ' bnc.ngm.3.plus.txt | sort -k2nr -t $'\t'
ability to pay 108
ability to make 97
ability to cope 64
...
ability range 17
...
ability and willingness 9
...
ability and enthusiasm 6
ability and motivation 6
ability could 6
ability of local 6
ability of the system 6
ability tests 6
...
ability to conceive and develop 3
ability to conduct 3
ability to construct and convey 3
...
ability to make sense 3
ability to meet the challenges 3
ability to recognise words 3
...
ability to solve problems 3
ability to summon 3
ability to talk and write 3
ability to think logically 3
...
$
```

Extracting Collocations with Local MapReduce

- Mapper

```
from collections import defaultdict, Counter
import sys
from nltk.corpus import stopwords

eng_stopwords = set(stopwords.words('english'))
max_distance = 5
skipbigram = defaultdict(Counter)

for line in sys.stdin:
    ngm, count = line.strip().split('\t')
    ngm = ngm.split(); distance = len(ngm)-1
    skipbigram[ngm[0]+' '+ngm[-1]][distance] += int(count)
    skipbigram[ngm[-1]+' '+ngm[0]][-distance] += int(count)

for bigram in sorted(skipbigram.keys()):
    print('%s\t%s\t%s'%(bigram, sum(skipbigram[bigram].values()),
          sorted(list(skipbigram[bigram].items()), key=lambda x: x[1] )) )
```

- Reducer

```
from math import sqrt
from itertools import groupby
import sys
```

```
k0, U0, k1 = 1, 10, 5
def getHighCounts(list1, COUNT, k):
    if not list1:
        return []
    size = len(list1)
    totals = [ COUNT(x) for x in list1 ]
    grandtotal = sum(totals)
    avg = (0.0+grandtotal)/size
    sdv = sqrt( sum( (x-avg)**2 for x in totals )/size )
    return [ x for x in list1 if COUNT(x) >= avg+k*sdv ]

lines = [ line.strip().split('\t') for line in sys.stdin ]
lines = [ x[0].split()+x[1:] for x in lines]
for head, headgroup in groupby(lines, key=lambda x: x[0]):
    cands = [ (x[0], x[1], int(x[2]), eval(x[3])) for x in headgroup]
    cands.sort(key= lambda x: x[2] )
    goodColls = getHighCounts(cands, lambda x: x[2], k0)
    goodColls = [ (head, coll, total,
                            getHighCounts(dCounts, lambda x: x[1], k1) )
                            for head, coll, total, dCounts in goodColls ]
    for head, coll, total, dCounts in goodColls:
        if dCounts: print('%s\t%s\t%s\t%s' % (head, coll, total, dCounts))
```

Lab Work

- To be announced
- Purpose:
- Input:
- Output:
- Mapper
- Reducer