Introduction to MapReduce

EECS 4415

Big Data Systems

Tilemachos Pechlivanoglou

tipech@eecs.yorku.ca

MapReduce

- Our first peek into MapReduce implementation
- Using Python
- Example program: WordCount

Conventional approach

```
#!/usr/bin/python

import sys
import re

sums = {}

for line in sys.stdin:
    line = re.sub( r'^\W+|\W+$', '', line )
    words = re.split(r'\W+', line)

    for word in words:
        word = word.lower()
        sums[word] = sums.get( word, 0 ) + 1

print sums
```

Conventional (step 0)

Preparation:

```
import sys
import re

sums = {}
```

Loading file line by line:

```
for line in sys.stdin:
```

Conventional (step 1)

Removing non-word characters:

```
line = re.sub( r'^\W+|\W+$', '', line )
```

Splitting into words:

```
words = re.split( r'\W+', line )
```

Conventional (step 2)

Iterating over words:

```
for word in words:
```

Making everything lowercase:

```
word = word.lower()
```

Incrementing the count of every word in the dictionary

```
sums[word] = sums.get( word, 0 ) + 1
```

(if word doesn't exist, get 0)

Conventional (Moby Dick)

Section 5. General Information About Project Gutenberg-tm electronic works.

Professor Michael S. Hart is the originator of the Project Gutenberg-tm concept of a library of electronic works that could be freely shared with anyone. For thirty years, he produced and distributed Project Gutenberg-tm eBooks with only a loose network of volunteer support.

Project Gutenberg-tm eBooks are often created from several printed editions, all of which are confirmed as Public Domain in the U.S. unless a copyright notice is included. Thus, we do not necessarily keep eBooks in compliance with any particular paper edition.

Most people start at our Web site which has the main PG search facility:

http://www.gutenberg.org

This Web site includes information about Project Gutenberg-tm, including how to make donations to the Project Gutenberg Literary Archive Foundation, how to help produce our new eBooks, and how to subscribe to our email newsletter to hear about new eBooks.

Conventional (output)

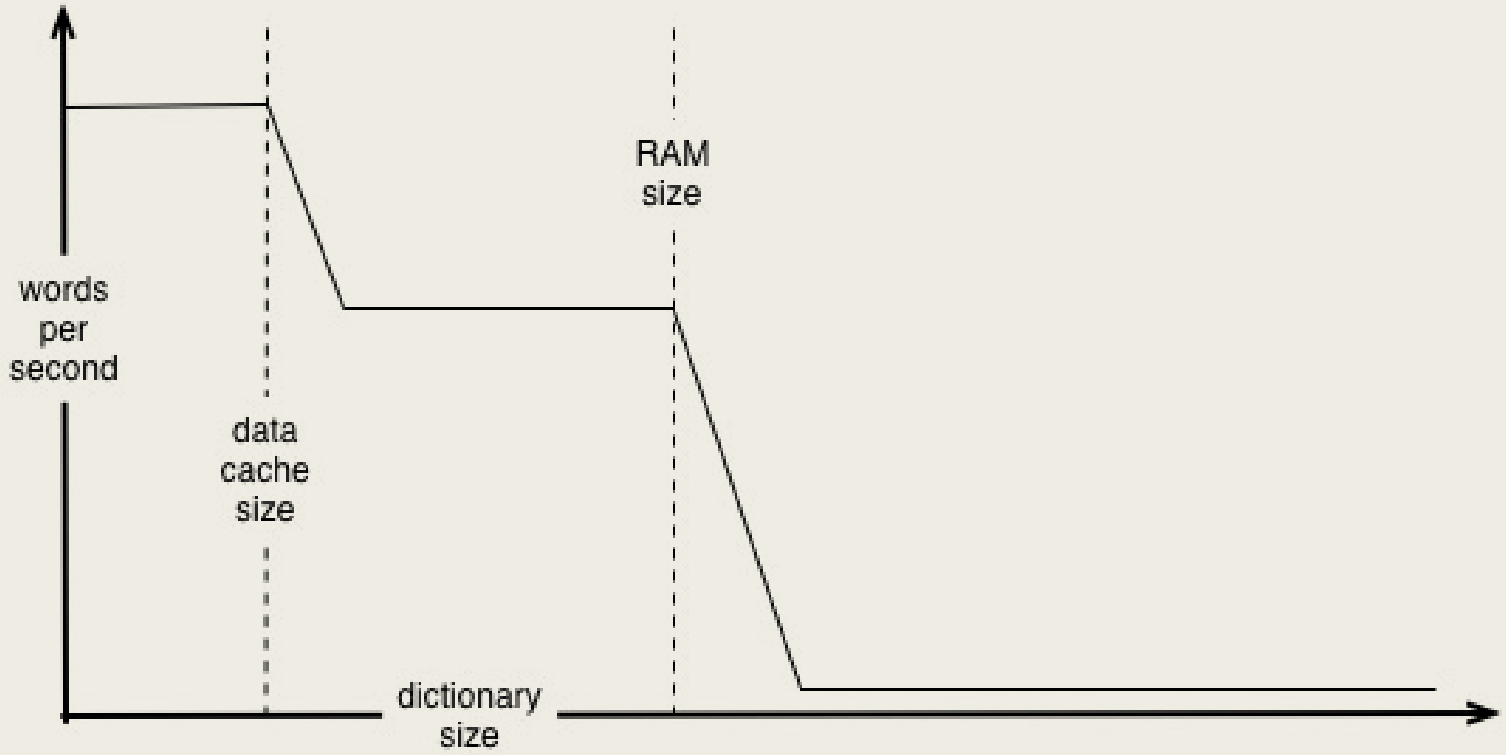
```
ty': 2, 'outset': 3, 'own': 205, 'polished': 7, 'boggy': 1, 'strangeness': 3, 's
ugary': 1, 'owe': 1, 'degenerated': 3, 'canaan': 2, 'trunks': 2, 'promise': 7, '
brush': 1, 'decree': 1, 'freeze': 1, 'zoology': 2, 'intricacies': 4, 'barques':
1, 'fired': 3, 'linnaeus': 5, 'van': 5, 'pillaged': 1, 'crave': 1, 'rivals': 2,
'transfer': 2, 'spiral': 1, 'captains': 24, 'continental': 2, 'intention': 8, 'a
ppals': 2, 'monopolising': 2, 'powdered': 1, 'breeding': 2, 'throttled': 1, 'vat
': 1, 'callings': 1, 'shank': 3, 'tashtego': 57, 'hollanders': 1, 'cankerous': 1
, 'billion': 1, 'mutter': 3, 'volume': 13, 'wight': 5, 'contradictory': 3, 'cana
llers': 8, 'assail': 1, 'swayings': 1, 'tinges': 1, 'unreverenced': 1, 'jaffa':
2, 'unwearied': 2, 'made': 178, 'israelites': 1, 'whether': 91, 'protesting': 1,
 'swashing': 2, 'recede': 1, 'nightgown': 1, 'venetian': 4, 'record': 7, 'below'
: 52, 'persian': 6, 'ruling': 1, 'cake': 2, 'demonstrate': 1, 'rickety': 1, 'hov
eringly': 1, 'stirring': 4, 'liturgies': 1, 'unfolding': 1, 'heralding': 1, 'eag
erness': 8, 'apothecary': 3, 'ploughing': 2, 'scepticism': 2, 'betrayed': 6, 'gn
awing': 1, 'sheered': 1, 'goodness': 5, 'globules': 2, 'theatre': 1, 'raced': 1,
 'trotting': 1, 'domesticated': 1, 'kant': 1, 'swaine': 1, 'mutual': 5, 'improvi
ng': 2, 'monsoons': 1, 'besmoked': 1, 'incredible': 9, 'honing': 1, 'afoam': 1,
'boot': 2, 'illinois': 3, 'unreluctantly': 1, 'book': 60, 'boom': 8, 'sick': 10,
 'unlettered': 1, 'repute': 1, 'incredibly': 1, 'flexion': 1, 'conclusion': 6, '
lance': 45, 'junk': 5, 'kinds': 3, 'scabbards': 1, 'nourishment': 2, 'june': 3,
'circumspection': 1, 'frustrate': 1, 'risked': 1, 'pumps': 15, 'earthly': 16, 'r
ichardson': 1, 'gap': 1, 'upwards': 19, 'auxiliary': 2, 'ranks': 2, 'pods': 4, '
yawed': 1, 'jewel': 2, 'gam': 7, 'sash': 1, 'uncatastrophied': 1, 'races': 1, 'v
olumes': 2, 'sleeves': 5, 'expands': 2, 'sunda': 6}
```

Limitations of approach

- Requires use of dictionary
  - *entire object stored in memory*
  - *if too big for memory crashes*

- Slower as dictionary grows
  - *the bigger it is, the more time needed to get key (word)*

Limitations of approach (graph)



MapReduce approach

Does not require a central data structure (dictionary)

Steps:

- **map**: intermediate results, associates them with output key
- **shuffle**: intermediate results, same output key
- **reduce**: final result, takes keys as input

MapReduce Mapper

```
#!/usr/bin/python

import sys
import re

for line in sys.stdin:
    line = re.sub( r'^\W+|\W+$', '', line )
    words = re.split(r"\W+", line)

    for word in words:
        print( word.lower() + "\t1" )
```

MapReduce Mapper (step 0)

Same first steps:

```
import sys
import re

sums = {}

for line in sys.stdin:
        line = re.sub( r'^\W+|\W+$', '', line )
        words = re.split( r'\W+', line )

        for word in words:
```

MapReduce Mapper (step 1)

Output word and count:

- convert to lowercase
- “\t” (tab) is Hadoop for “:” separates key from value

```
print( word.lower() + "\t1" )
```

Conventional execution:

```
./mapper.py < input.txt
```

MapReduce Mapper (output)

```
gutenberg       1
literary        1
archive 1
foundation      1
how     1
to      1
help    1
produce 1
our     1
new     1
ebooks  1
and     1
how     1
to      1
subscribe       1
to      1
our     1
email   1
newsletter      1
to      1
hear    1
about   1
new     1
ebooks  1
```

MapReduce Shuffle

Simple sort of calculated words

- Running on a cluster, more distribution happens here

Conventional execution (Linux command):

```
./mapper.py < input.txt | sort
```

MapReduce Shuffle (output)

```
zealanders      1
zephyr  1
zeuglodon       1
zig     1
zip     1
zodiac  1
zodiac  1
zodiac  1
zodiac  1
zodiac  1
zogranda        1
zone    1
zone    1
zone    1
zone    1
zone    1
zoned   1
zoned   1
zones   1
zones   1
zones   1
zoology 1
zoology 1
zoroaster       1
```

MapReduce Reducer

```
#!/usr/bin/python

import sys

previous = None
sum = 0

for line in sys.stdin:
    key, value = line.split( '\t' )

    if key != previous:
        if previous is not None:
            print str( sum ) + '\t' + previous
        previous = key
        sum = 0

    sum = sum + int( value )

print str( sum ) + '\t' + previous
```

MapReduce Reducer (step 0)

Preparation:

```
import sys

previous = None
sum = 0
```

Loading previous results line by line:

```
for line in sys.stdin:
```

MapReduce Reducer (step 1)

Split pairs again:

```
key, value = line.split( '\t' )
```

If we are still counting occurences of the same word:

```
if key != previous:
```

Unless it's the first entry:

```
if previous is not None:
```

MapReduce Reducer (step 1)

Sum up 2 words:

```
print str( sum ) + '\t' + previous
```

Otherwise, re-initialize for next word

```
previous = key
         sum = 0
```

Either way, add new value to sum

```
sum = sum + int( value )
```

MapReduce Reducer (step 4)

Return those two words:

```
print str( sum ) + '\t' + previous
```

Conventional execution:

```
./mapper.py < input.txt | sort | ./reducer.py
```

MapReduce Reduce (output)

258	your
9	yours
1	yourselbs
26	yourself
7	yourselves
9	youth
2	youthful
1	zag
1	zay
2	zeal
7	zealand
1	zealanders
1	zephyr
1	zeuglodon
1	zig
1	zip
5	zodiac
1	zogranda
5	zone
2	zoned
3	zones
2	zoology
1	zoroaster

MapReduce Execution

```
hadoop jar /usr/hadoop-3.0.0/share/hadoop/tools/lib/hadoop-streaming-
3.0.0.jar \
-file ./mapper.py \
-mapper ./mapper.py \
-file ./reducer.py \
-reducer ./reducer.py \
-input /input.txt \
-output /output
```

Thank you!

Based on

https://zettadatanet.wordpress.com/2015/04/04/a-hands-on-introduction-to-mapreduce-in-python/