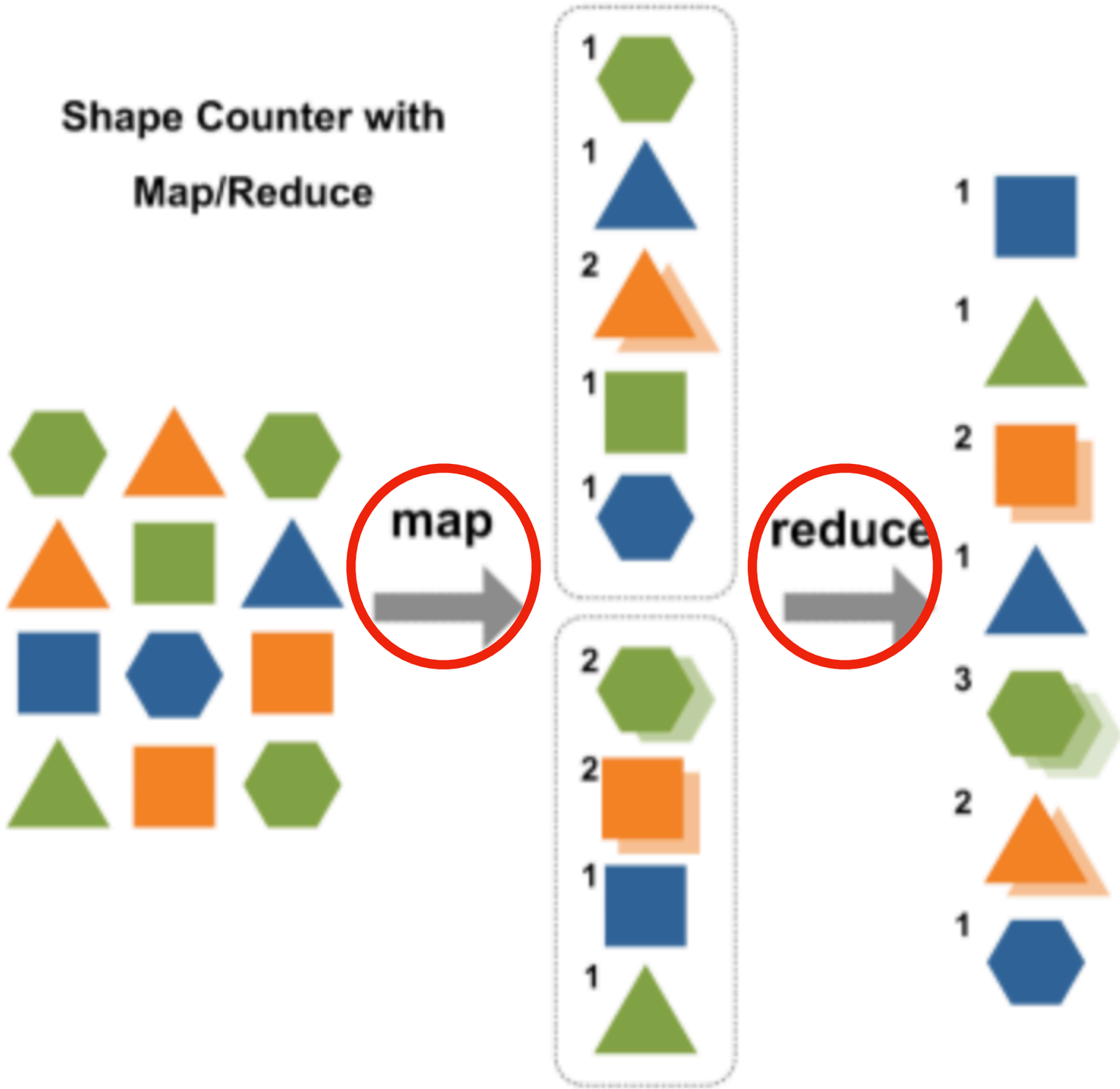
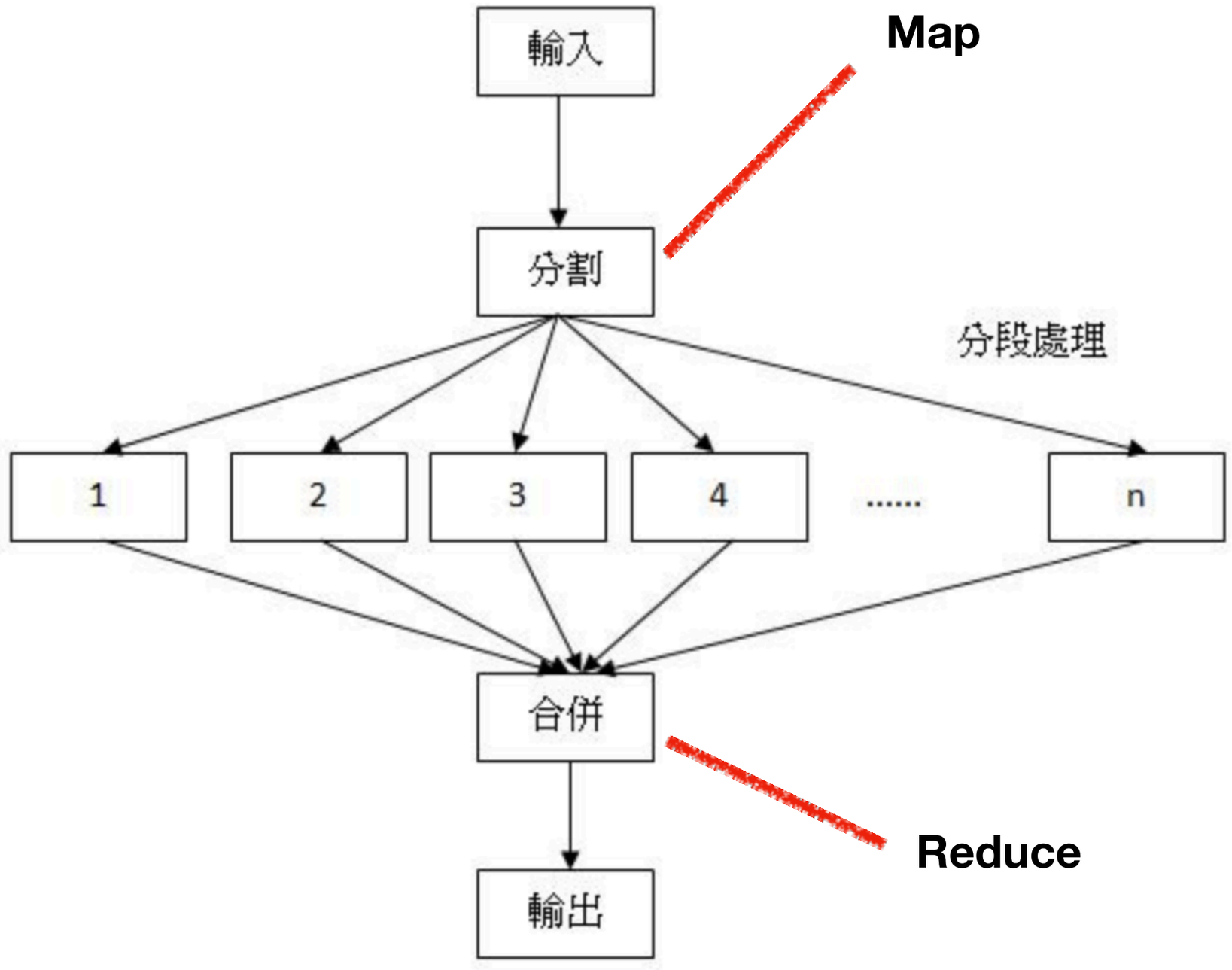
Local Map Reduce

2018/1/15

Map Reduce 是什麼？




輸入
Map
分割
分段處理
1
2
3
4
.....
n
合併
Reduce
輸出

Map Reduce Excample 1: Word Count

先來個 Wordcount !!

```
charles@227-134 ~/lmr  master echo 'aaa bbb ccc\naaa bbb aaa' | tr -cs "a-zA-Z" "\n"
aaa
bbb
ccc
aaa
bbb
aaa
```

解釋：

```
tr -cs "a-zA-Z" "\n"
```

將連續的字母內容轉換成迴圈，就是將單詞分行輸出。

```
echo 'aaa bbb ccc\naaa bbb aaa' | tr -sc "a-zA-Z" "\n"
```

Wordcount(計次) !!
(類似於mapreduce中的reduce)

```
charles@227-134 ~/lmr master echo 'aaa bbb ccc\naaa bbb aaa' | tr -sc "a-zA-Z" "\n" | sort -k 1,1 -t $'\t' | uniq -c
  3 aaa
  2 bbb
  1 ccc
```

```
echo 'aaa bbb ccc\naaa bbb aaa' | tr -sc "a-zA-Z" "\n" | sort -k 1,1 -t $'\t' | uniq -c
```

Map Reduce Excample 2: N-gram count in python

首先～要準備兩個檔案分別為

1.map.py

2.reducer.py

map.py

```python
#!/usr/bin/env python
import re
def tokens(str1): return re.findall('[a-z]+', str1.lower())

def to_ngrams( words, length):
    return zip(*[words[i:] for i in range(length)])

import fileinput
from collections import Counter
for line in fileinput.input():
    words = tokens(line)
    for n in range(1, 6):
        ngrams_counts = Counter(to_ngrams(words, n))
        for ngram, count in ngrams_counts.iteritems():
            print  '{}\t{}'.format(' '.join(ngram), count)
```

reducer.py

```python
#!/usr/bin/env python
import fileinput
from collections import Counter, defaultdict
ngram_counts = defaultdict(Counter)
for line in fileinput.input():
    ngram, count = line.split('\t', 1)
    ngram, count = tuple(ngram.split(' ')), int(count)
    length = len(ngram)
    ngram_counts[length][ngram] += count

for length in ngram_counts:
    for ngram, count in ngram_counts[length].iteritems():
        print '{}\t{}'.format(' '.join(ngram), count)
```

測試map.py會跑出什麼！

```
charles@227-134 ~/lmr master echo 'aaa bbb ccc\n aaa bbb aaa' | python map.py
aaa     1
bbb     1
ccc     1
aaa bbb 1
bbb ccc 1
aaa bbb ccc     1
aaa     2
bbb     1
aaa bbb 1
bbb aaa 1
aaa bbb aaa     1
```

```
echo 'aaa bbb ccc\naaa bbb aaa' | python map.py
```

測試加上Reducer會跑出什麼！

```
charles@227-134 ~/lmr master echo 'aaa bbb ccc\n aaa bbb aaa' | python map.py | sort -k 1,1 -t $'\t' | python reducer.py
aaa     3
bbb     2
ccc     1
aaa bbb 2
bbb aaa 1
bbb ccc 1
aaa bbb aaa     1
aaa bbb ccc     1
```

```
echo 'aaa bbb ccc\naaa bbb aaa' | python map.py | sort -k 1,1 -t $'\t' | python reducer.py
```

看看Reducer的作用～

```
charles@227-134 ~/lmr master echo 'aaa bbb ccc\n aaa bbb aaa' | python map.py
aaa     1
bbb     1
ccc     1
aaa bbb 1
bbb ccc 1
aaa bbb ccc     1
aaa     2
bbb     1
aaa bbb 1
bbb aaa 1
aaa bbb aaa     1
```

```
charles@227-134 ~/lmr master echo 'aaa bbb ccc\n aaa bbb aaa' | python map.py | sort -k 1,1 -t $'\t' | python reducer.py
aaa     3
bbb     2
ccc     1
aaa bbb 2
bbb aaa 1
bbb ccc 1
aaa bbb aaa     1
aaa bbb ccc     1
```

single process

```
$ time pv citeseerx_descriptions_sents.txt.300000 | python map.py | sort -k1 | python reduce.py
37.6MiB 0:01:12 [ 528KiB/s] [=============================================================
pv citeseerx_descriptions_sents.txt.300000  0.06s user 0.18s system 0% cpu 1:12.81 total
python map.py  156.55s user 2.34s system 76% cpu 3:27.56 total
sort -k1  67.68s user 1.94s system 16% cpu 7:09.31 total
python reduce.py > result_single  217.10s user 0.26s system 50% cpu 7:09.38 total
```

時間相差五倍！！

Local-map reduce

```
$ rm -r result ; time pv citeseerx_descriptions_sents.txt.300000 | ./lmr 2m 16 'python map.py' 'python
37.6MiB 0:00:23 [1.58MiB/s] [=============================================================
pv citeseerx_descriptions_sents.txt.300000  0.01s user 0.03s system 0% cpu 23.789 total
./lmr 2m 16 'python map.py' 'python reduce.py' result  1024.95s user 11.64s system 1318% cpu 1:18.63 t
```

End