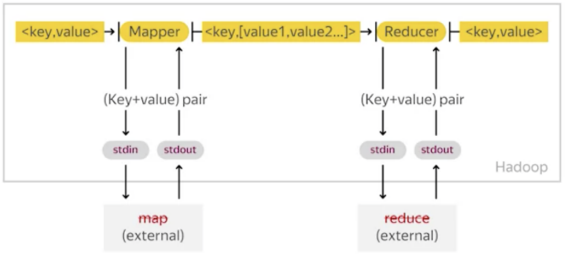
Week 2: Hadoop MapReduce Streaming Applications in Python

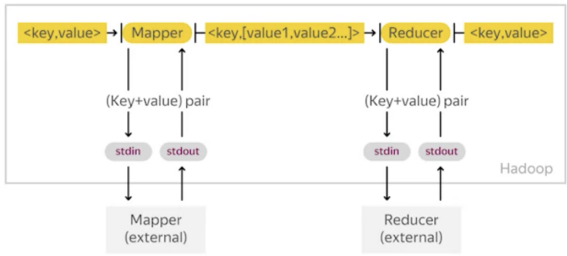
Streaming

Shell streaming

Scipy와 Numpy같은 C로 만든 라이브러리와 호환이 불가하다.

하둡에서 다양한 언어를 지원





- 표준 입출력을 활용하여 프로그램 코드 작성

Mapper

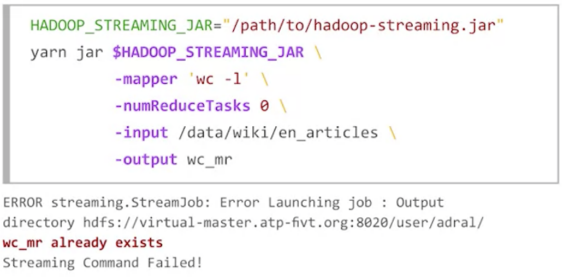
- define input format
- process data
- define output format

Reducer

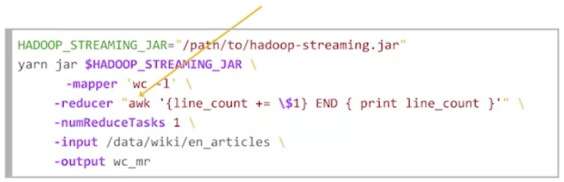
- define input format
- aggregate sorted data by key
- process data
- define output format

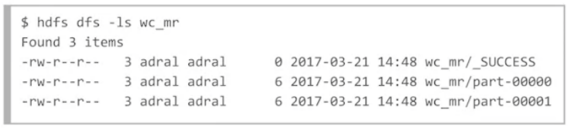
```
$ man locate //streaming jar 경로 찾기
```

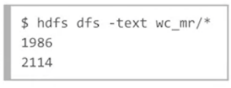
- yarn jar: open yarn service
- -numReduceTasks option: reduce task X



- output path는 새로워야함
- 작은 데이터에 대해 검증해보는 습관







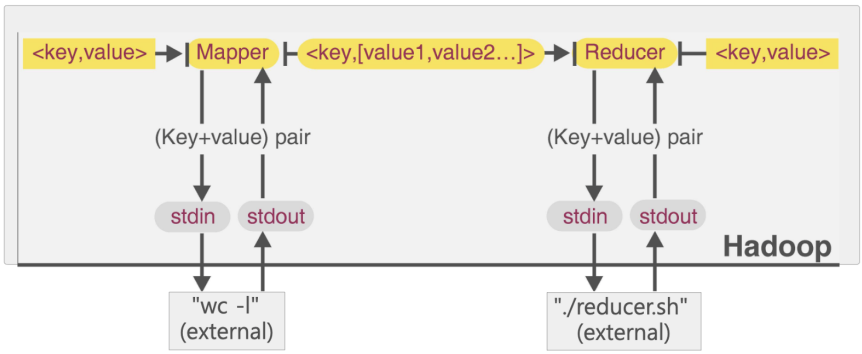
1968 + 2114 = 4100

- **reducer.sh**

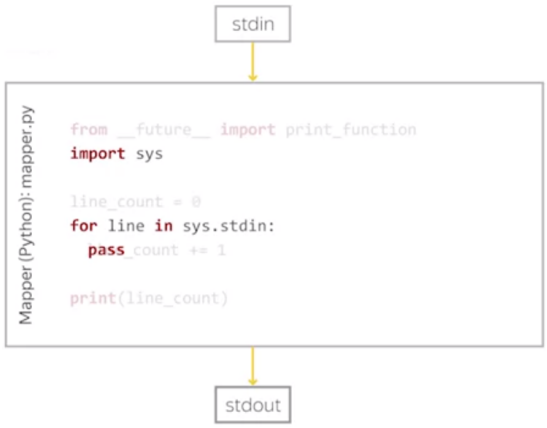
sh 파일을 직접 만들어 node에 업로드 후 사용

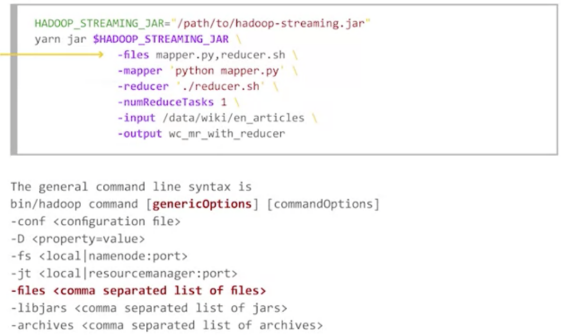
```
#! /usr/bin/env bash
awk '{line_count += $1} END { print line_count }'

HADOOP_STREAMING_JAR = '/path/to/hadoop-streaming.jar'
yarn jar $HADOOP_STREAMING_JAR \
              -mapper 'wc -l' \
              -reducer './reducer.sh' \
```

```
    -file reducer.sh \
    -numReduceTasks 1 \
    -input /data/wiki/en_articles \
    -output wc_mr_with_reducer
```



Streaming in Python





```
HADOOP_STREAMING_JAR="/path/to/hadoop-streaming.jar"
yarn jar $HADOOP_STREAMING_JAR \
        -files mapper.py,reducer.sh \
        -mapper 'python mapper.py' \
        -reducer './reducer.sh' \
        -numReduceTasks 1 \
        -input /data/wiki/en_articles \
        -output wc_mr_with_reducer
```

```
The general command line syntax is
bin/hadoop command [genericOptions] [commandOptions]
-conf <configuration file>
-D <property=value>
-fs <local|namenode:port>
-jt <local|resourcemanager:port>
-files <comma separated list of files>
-libjars <comma separated list of jars>
-archives <comma separated list of archives>
```

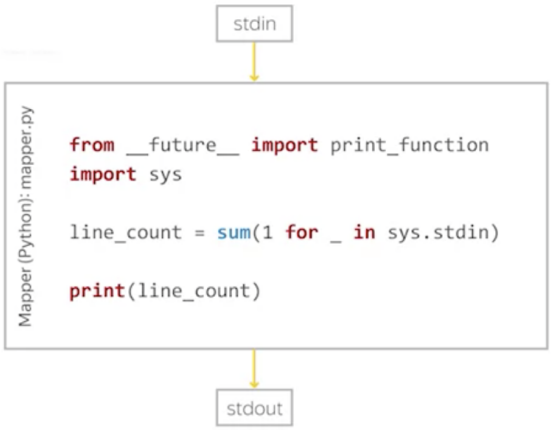
```
HADOOP_STREAMING_JAR="/path/to/hadoop-streaming.jar"
yarn jar $HADOOP_STREAMING_JAR \
        -files mapper.py,reducer.sh \
        -mapper 'python mapper.py' \
        -reducer './reducer.sh' \
        -numReduceTasks 1 \
        -input /data/wiki/en_articles \
        -output wc_mr_with_reducer
```

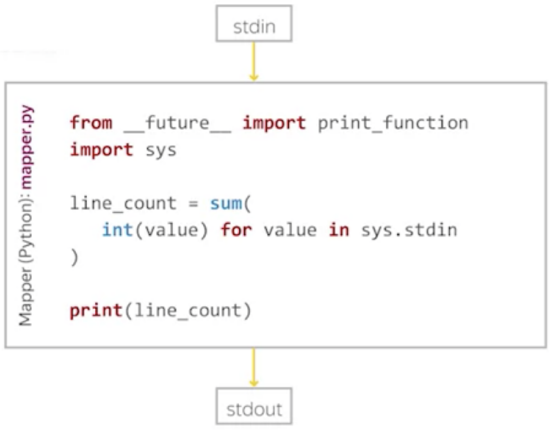
```
$ hdfs dfs -ls wc_mr_with_reducer
Found 2 items
-rw-r--r-- 3 adral adral 0 <date> wc_mr_with_reducer/_SUCCESS
-rw-r--r-- 3 adral adral 0 <date> wc_mr_with_reducer/part-00000
```

permissions **number_of_replicas** userid groupid **filesize** modification_date modification_time filename

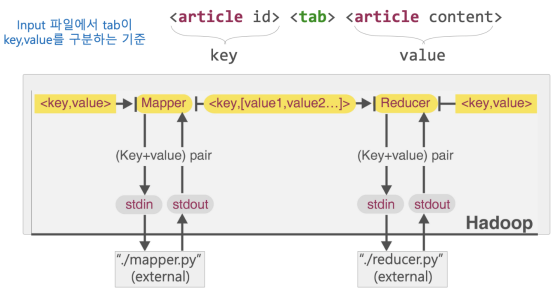
```
$ hdfs dfs -text wc_mr_with_reducer/*
--
```

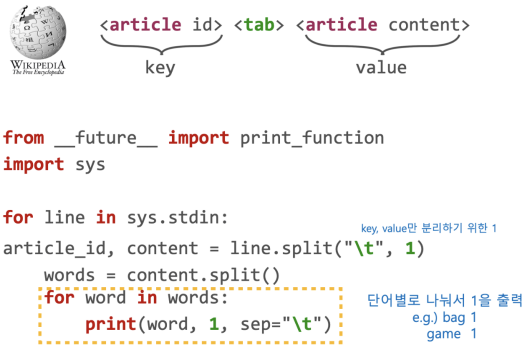
- generic options (job meta configuration parameters..) 다음 command options (mapper or reducer to use)





WordCount in Python






```
yarn jar $HADOOP_STREAMING_JAR \
                -files mapper.py \
                -mapper 'python mapper.py' \
                -numReduceTasks 0 \
                -input /data/wiki/en_articles \
                -output word_count
```

```
$ hdfs dfs -text /data/wiki/en_articles/* | head -c 80
12 <tab> Anarchism          Anarchism is often defined as a political
philosophy which …
```

```
$ hdfs dfs -ls -h word_count
Found 3 items
-rw-r--r-- 3 adral adral 0 2017-03-22 11:40 word_count/_SUCCESS
-rw-r--r-- 3 adral adral 47.8 M 2017-03-22 11:40 word_count/part-00000
-rw-r--r-- 3 adral adral 47.9 M 2017-03-22 11:40 word_count/part-00001
```

```
yarn jar $HADOOP_STREAMING_JAR \
                -files mapper.py \
                -mapper 'python mapper.py' \
                -numReduceTasks 0 \
                -input /data/wiki/en_articles \
                -output word_count
```

```
$ hdfs dfs -text /data/wiki/en_articles/*
| head -c 80
12 <tab> Anarchism          Anarchism is
often defined as a political philosophy
which …
```

part-00000	part-00001
Basel 1	Anarchism 1
Basel 1	Anarchism 1
(1	is 1
) 1	often 1
or 1	defined 1
…	…

```
$ hdfs dfs -text
    word_count/part-… | head -5
```

```
yarn jar $HADOOP_STREAMING_JAR \
                    -files mapper.py \
                    -mapper 'python mapper.py' \
                    -numReduceTasks 1 \
                    -input /data/wiki/en_articles \
                    -output word_count
```

```
$ hdfs dfs -text word_count/part-00000 | head
!   1
!   1
!   1
!   1
!   1
…
```

mapper만 실행해도 numReduceTasks를 1이상으로 선언하면 Shuffle&sort가 수행된다.

key별로 정렬이 된 것을 확인가능

<article id> <tab> <article content>

key value

mapper.py 수정버전

```
from __future__ import print_function
import re
import sys

for line in sys.stdin:
   article_id, content = line.split("\t", 1)
    words = re.split("\W+", content)
    for word in words:
       if word:
          print(word, 1, sep="\t")
```

특수문자를 지우기 위한 정규식 처리

₩W : word가 아닌 것들(!, 공백 등)

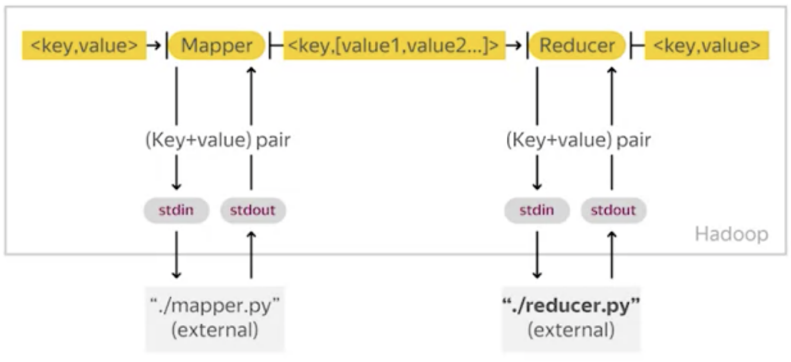
```
yarn jar $HADOOP_STREAMING_JAR \
                    -files mapper.py \
                    -mapper 'python mapper.py' \
                    -numReduceTasks 1 \
                    -input /data/wiki/en_articles \
  특수문자가 제거됨  -output word_count
```

```
$ hdfs dfs -text word_count/part-00000 | head -4
0 1
0 1
0 1
0 1
```

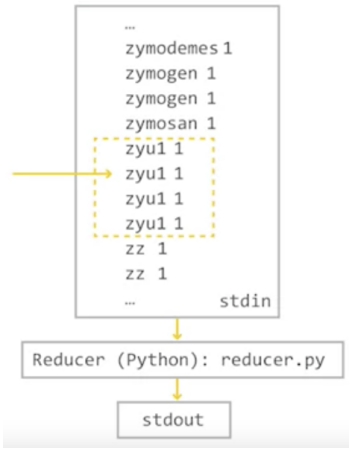
```
$ hdfs dfs -tail word_count/part-00000 | tail -4
zyu1 1
zyu1 1
zz   1
zz   1
```

\W: 특수문자 제거

\w: 문자 제거..



- define input format
- **aggregate sorted data by key**
- process data
- define output format



- numReduceTasks 지정하지 않으면 자동으로 여러개 생성

```
$ hdfs dfs -tail word_count/part-... | tail -5
```

→ 각각의 reducer에서는 결과가 정렬되지만 globally sort는 되지 않음

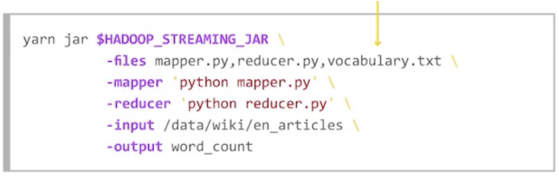
⇒ TotalOrderPartitioner 옵션 존재

- You know what MapReduce Streaming is and how it works
- You know how to write MapReduce Bash and Python Streaming applications
- You should be able to solve WordCount or similar problems in MapReduce in Python by yourself

---

Distributed Cache

worker들이 사용해야 할 파일을 미리 Node에 업로드해서 공유

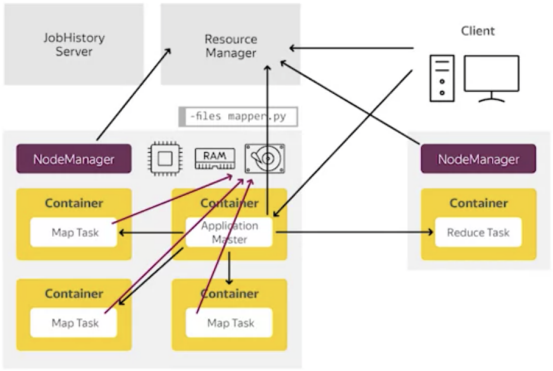


단어 파일을 worker Node에 업로드

```python
from __future__ import print_function
import re
import sys

def read_vocabulary(file_path):
    return set(word.strip() for word in open(file_path))

vocabulary = read_vocabulary("vocabulary.txt")

for line in sys.stdin:
    article_id, content = line.split("\t", 1)
    words = re.split("\W+", content)
    for word in words:
        if word in vocabulary:
            print(word, 1, sep="\t")
```

Mapper (Python): mapper.py

```python
def read_vocabulary(file_path):
    return set(word.strip() for word in open(file_path))

vocabulary = read_vocabulary("vocabulary.txt")
```

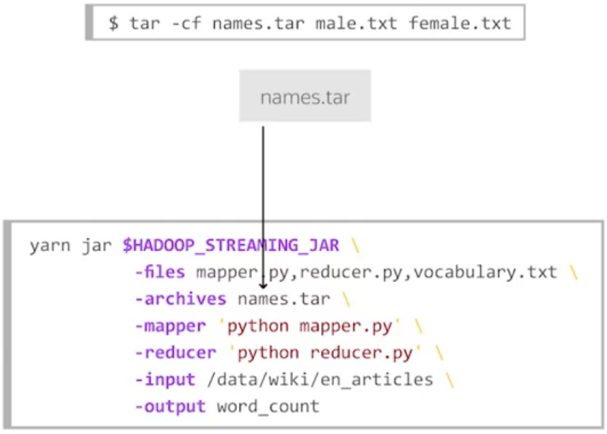


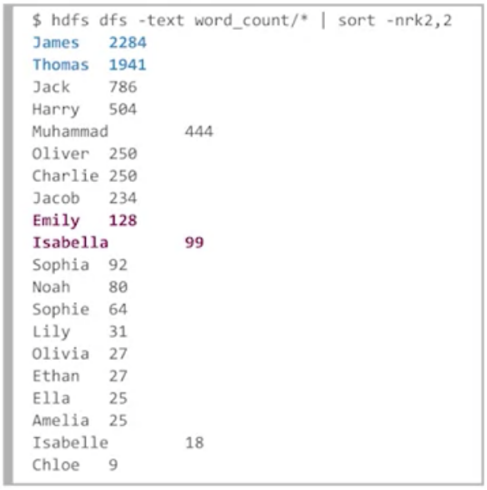
- -files에 지정된 파일들을 모든 worker node에 업로드함
- task container들은 node의 디스크에 접근하여 데이트를 읽음

Do **NOT MODIFY** a file content in a distributed cache!

every other container on the same host machine will access the modified version of this file (they are accessed via symlinks). Therefore, if you your script execution relies on it then you can get a different output. It breaks functional paradigm and, therefore, you will have non reproducible results.

-files -archives -libjars

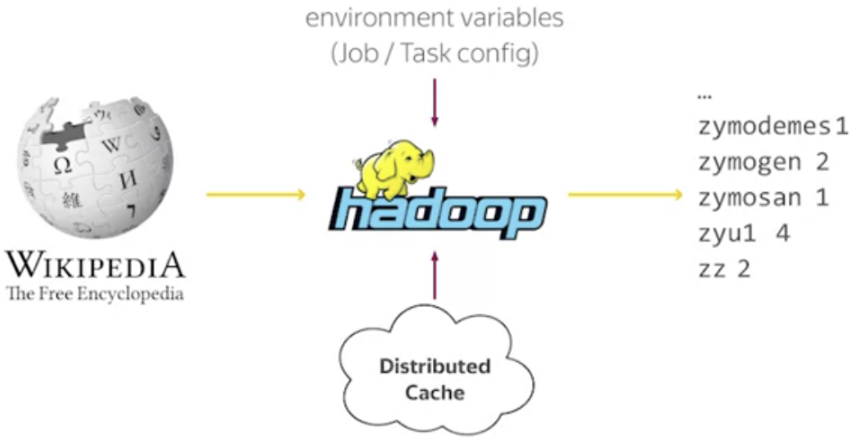





```
$ hdfs dfs -text word_count/* | sort -nrk2,2
```

Environment, Counters

environment variable also possible



Mapper (Python): mapper.py

```
from __future__ import print_function
import re
import sys

if os.environ["mapred_task_is_map"] == "true":
print("input_file:{}, start:{}, size:{}".format(
os.environ["mapreduce_map_input_file"],
os.environ["mapreduce_map_input_start"],
os.environ["mapreduce_map_input_length"],
))

for line in sys.stdin:
    pass
```

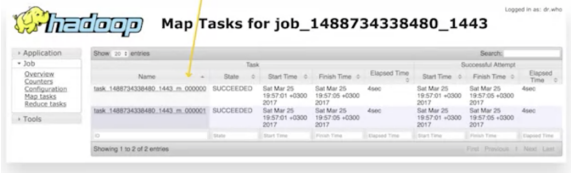
default 환경변수 예시

environment variables
(Job / Task config)

Mapper (Python): mapper.py

```
os.environ["mapreduce_task_id"]
os.environ["mapreduce_task_partition"]
```

task_1488734338480_1443_m_000001
task_1488734338480_1443_r_000008




```
from __future__ import print_function
import re
import sys

CHARS_IN_LINE = 9

if os.environ["mapred_task_is_map"] == "true":
   split_input_start = int(
        os.environ["mapreduce_map_input_start"]
   ) // CHARS_IN_LINE

for split_line_index, line in enumerate(sys.stdin):
   line_number = split_line_index + split_input_start
   if (line_number < 10):
      print(line_number, line, sep='\t')
```

<article id> <tab> <article content>

key value

Mapper (Python): wc_mapper.py

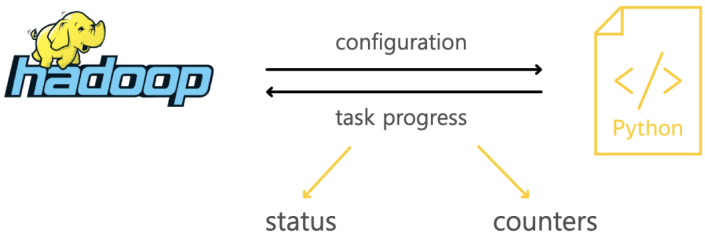
```
from __future__ import print_function
import os
import re
import sys

pattern = re.compile(os.environ["word_pattern"])

for line in sys.stdin:
    article_id, content = line.split("\t", 1)
    words = re.findall(pattern, content)
    for word in words:
        print (word, 1, sep="\t")
```

유저가 생성한 환경변수

유저가 생성한 환경변수 -D를 활용

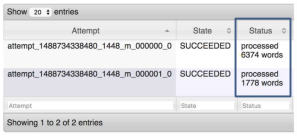
```
yarn jar $HADOOP_STREAMING_JAR -D word_pattern="\w+\d+" \
                -files mapper.py \
                -mapper 'python mapper.py' \
                -reducer 'python reducer.py' \
                -input /data/wiki/en_articles \
                -output word_count
```

```
$ hdfs dfs -text word_count/*
…
test2   4
times11 1
times48 3
tinctoria1      1
titan2
…
```



반대로 Script 코드에서 Hadoop에게 status와 counters 값을 전달할 수 있다.

Status example



Mapper (Python): reporter_mapper.py

```
from __future__ import print_function
import re
import sys

for line in sys.stdin:
    article_id, content = line.split("\t", 1)
    words = re.findall("\w+", content)
    for index, word in enumerate(words):
            print(word, 1, sep="\t")
            print("reporter:status:processed {} words"
                .format(index + 1), file=sys.stderr)
```

현재 worker의 상태를 표준출력으로 전달한다.

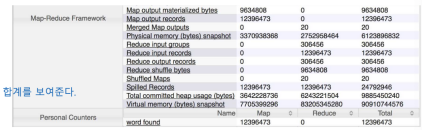
Counter example

Mapper (Python): reporter_mapper.py

```
from __future__ import print_function
import re
import sys

for line in sys.stdin:
    article_id, content = line.split("\t", 1)
    words = re.findall("\w+", content)
    for index, word in enumerate(words):
            print(word, 1, sep="\t")
            print("reporter:status:processed {} words"
                .format(index + 1), file=sys.stderr)

print("reporter:counter:Personal Counters,word found,1",file=sys.stderr)
```

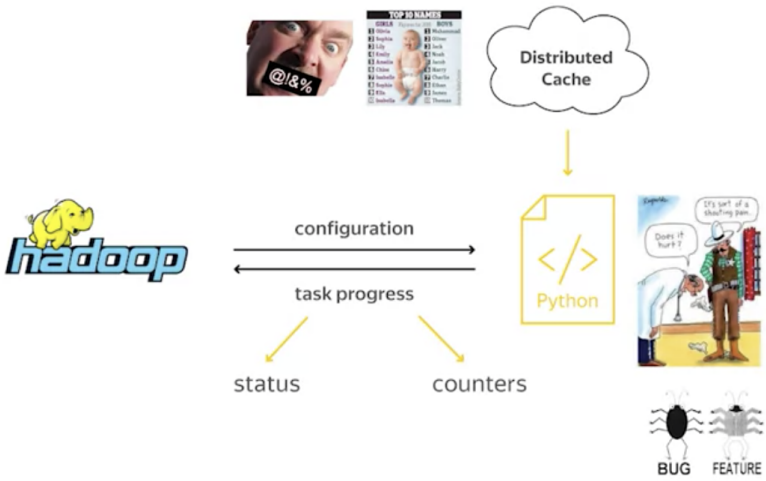
reporter:counter:<group>,<counter>,<amount>

각 worker에서 보내는 amount의 합계를 보여준다.

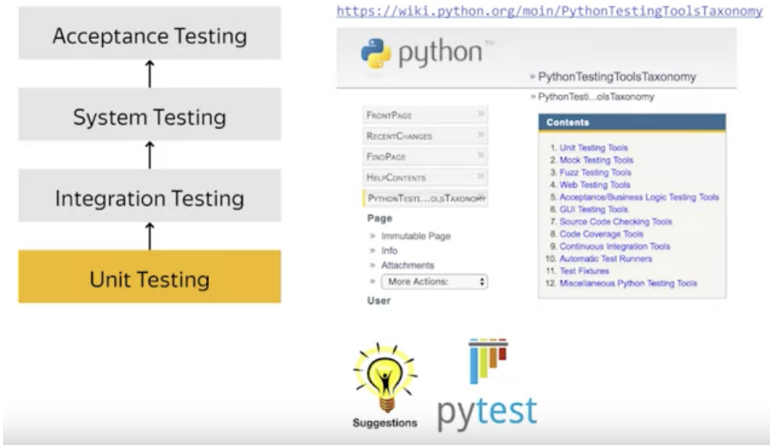


> You know how to:
  > provide environment variables (global configuration) to your streaming scripts
  > access job configuration options (e.g. map input file)
  > report progress back to Hadoop MapReduce Framework

Testing



- **Unit Testing - debug by python libraries**



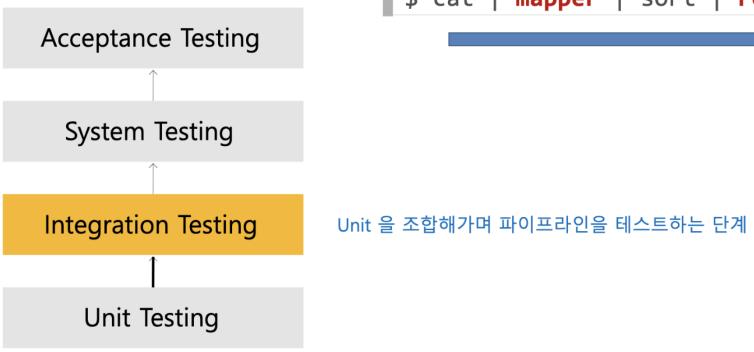
원치 않는 단어, 빈 값, input 없는 경우

```python
def get_words(input_line):
    return input_line.split()

def test_get_words_parse_simple_string():
    assert get_words("a b cd efg ") == ["a", "b", "cd", "efg"]

def test_get_words_parse_empty_string():
    assert get_words("") == []

def test_get_words_raise_exception_if_no_input():
    with pytest.raises(AttributeError):
        get_words(None)
```

- **Integration Testing - Unit 조합해가며 파이프라인을 테스트 (emulate mapreduce locally with cat|map|sort|reduce)**

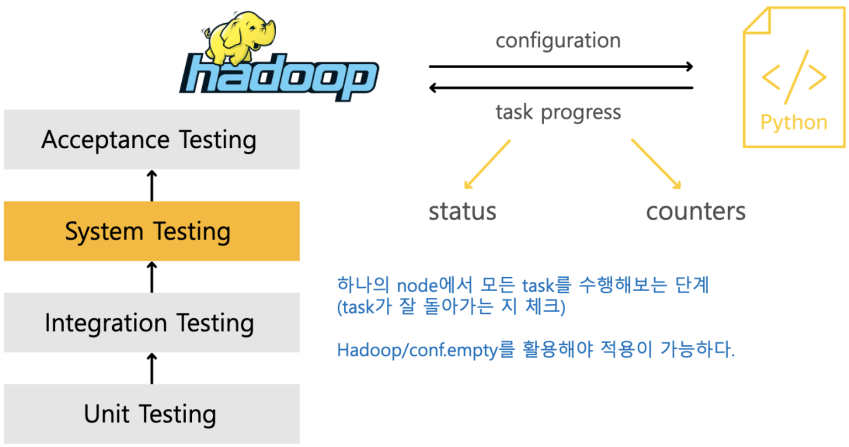
```
$ cat -n wikipedia.dump | tr ' ' '\n' | sort | uniq -c
```

```
$ cat | mapper | sort | reducer
```



Unit 을 조합해가며 파이프라인을 테스트하는 단계

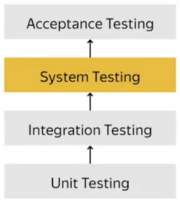
- **System Testing - 하나의 node에서 모든 task 수행해봄 (run mapreduce in a standalone mode)**

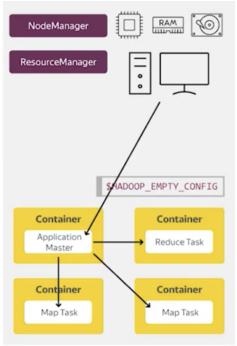


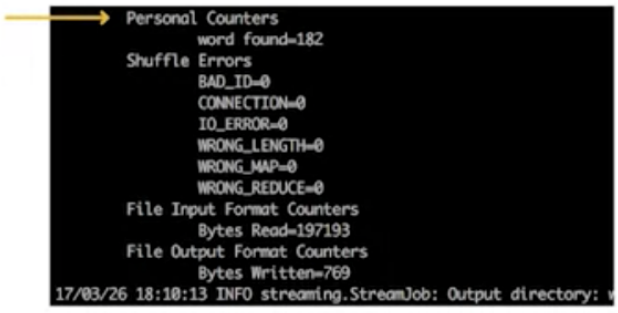

```
hdfs --config $HADOOP_EMPTY_CONFIG dfs -rm -r word_count
yarn --config $HADOOP_EMPTY_CONFIG jar $HADOOP_STREAMING_JAR
```

```
$ locate "hadoop/conf.empty"
```

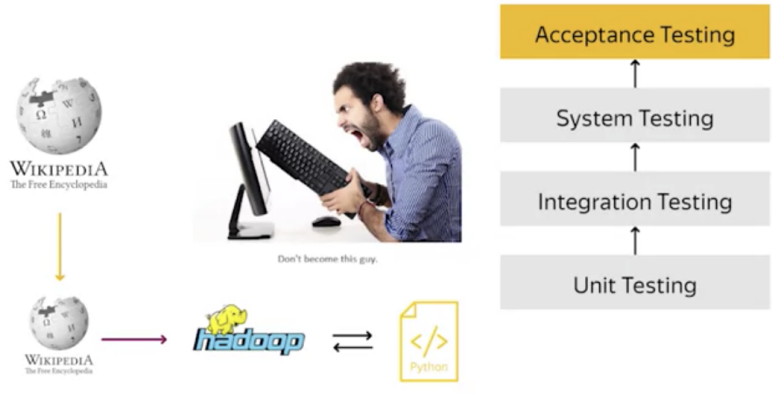
```
$ find "hadoop/conf.empty"
$ locate "hadoop/conf.empty"
```







- **Accentance Testing - why you need to execute mapreduce against sample datasets**



MapReduce Streaming (Quiz)