과목 명: 시스템프로그래밍

담당 교수 명: 소 정 민

<<Assignment 6>>

**서강대학교 컴퓨터공학과**

**[20151529]**

**[김예찬]**

목 차

1. 프로그램 개요 3

2. 프로그램 설명 3

2.1 프로그램 흐름도 3

2.2 프로그램 특징 3

3. 모듈 정의 4

3.1 모듈 이름 : Mapper.py 4

3.1.1 기능 4

3.1.2 사용 변수 4

3.2 모듈 이름: Reducer.py 4

3.2.1 기능 4

3.2.2 사용 변수 4

4. 코드 4

4.1 Mapper.py 4

4.2 Reducer.py 6

4.3 코드 실행 결과 사진 첨부 7

# 프로그램 개요

Microsoft Azure의 클라우드 시스템을 통해 할당받은 HDFS를 이용해 input file - 각 Group에 따른 value –를 직접 짠 Mapper.py, Reducer.py 로 MapReduce하여 각 Group 별 Maximum Value 값을 획득하는 프로그램이다. 각 노드들에 task를 할당 병행 처리하는 HDFS를 이용해보는 프로그램이다.

# 프로그램 설명

## 프로그램 흐름도

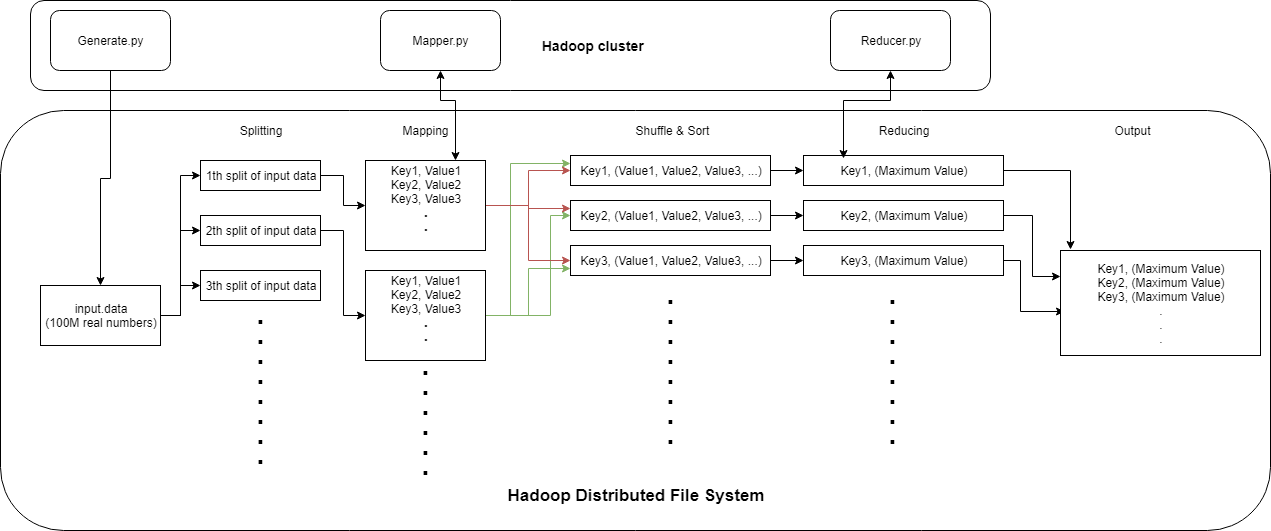


그림 1> 프로그램 흐름도

Generate.py로 생성한 100 million개의 Real number로 구성된 input file을 HDFS로 transfer한 뒤, input data를 split하여 Map node들에게 task 를 할당해 준다. 이 후 작성한 Mapper.py, Reducer.py 프로그램(function, 알고리즘)을 이용해 각 노드들에서data를 process한다. 이 때, Distributed file system의 병행 처리 및 abstraction 부분을 이용하여 구동시켜 원하는 output (Group 별 Maximum Value) 값을 병렬적으로 동시에 획득한다.

## 프로그램 특징

Hadoop의 MapReduce framework는 input data를 어떻게 나누어야 하는지에 대한 부분과, 프로그램 오류시 data의 reliability와 fault tolerance가 잘 되었는지, 마지막 결과들을 어떤 식으로 모아올지 등의 부분을 사용자가 고려하지 않아도 알아서 처리해주고, 오직 코드 로직에만 집중할 수 있게끔 해준다.

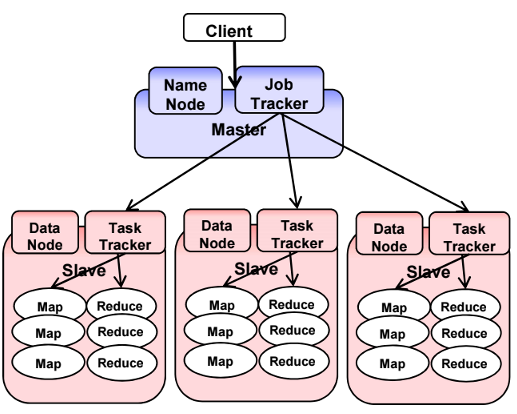
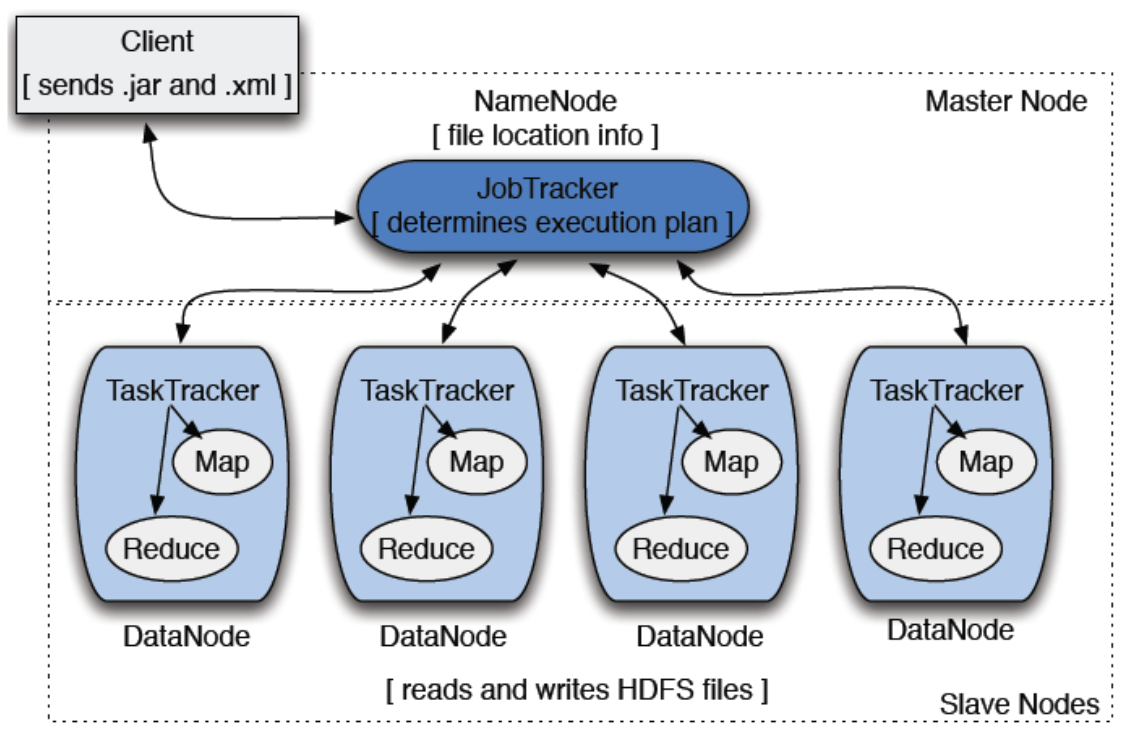
 

그림 2> HDFS 구조

이러한 MapReduce를 위한 HDFS구조는, 크게 클라이언트(유저), NameNode (Master)의 Job Tracker, DataNode(slave)의 Task Tracker로 나뉜다. Job Tracker는 여러 DataNode들의 역할을 결정하고 task를 나누어 스케줄링한 뒤, 유저가 작성한 프로그램을 구동하도록 request한다.

그런데 본래 MapReduce Program은 failure발생시 recovery를 위해 distributed system에 중간 결과를 저장함에 따라, 빠른 퍼포먼스를 보장한 모델이 아닌 거대한 규모의 data processing을 위한 모델이다. 이 모델을 이용하는 주된 이유는 framework의 최적화된 shuffle operation과 reliability 기능을 이용하고, 유저는 오직 Map과 Reduce function만 작성하면 되기 때문이다. 유저는 MapReduce 알고리즘을 작성하는데 있어서 각 노드에서의 Computation (예를 들어 Sorting)과 노드들 간의 Communication cost의 tradeoff를 고려하고 Data partition 역시 고려할 수 있다.

MapReduce process의 특징으로는, 각 DataNode들은 주기적으로 work, status update를 NameNode에 보고(“Heartbeat”)하는데, 만약 특정 시간간격에 보고가 오지 않는다면 죽은 것으로 판단, 다른 노드들에 task를 할당해준다는 것이다.

# 모듈 정의

## 모듈 이름 : mapper.py

### 기능

Input.data 파일을 한 줄씩 읽어 refine 되지 않은 각 line을 Group\_key와 Real number로 split, Reduce하기 위해 적절한 forma(Group key – Value)으로 intermediate file을 생성한다.

### 사용 변수

1. Data : system의 standard input으로부터 입력 받은 각 줄을 리스트로 반환받아 저장받는 변수
2. Value : “data” 변수로부터 받은 [Group key, Real number] 형태의 리스트에서 string으로 된 Real number을 slicing해서 저장하는 변수로, 이후 문자열의 마지막에 ‘\n’ 개행문자가 붙어있는 경우 이를 제외한 앞에 인덱스까지의 문자열을 다시 slicing, 재할당 및 저장하는 변수
3. Words : “data” 변수의 element들을 for in 반복문을 통해 순차적으로 받는 리스트 변수로, 첫 번째 원소로 group key, 두 번째 원소로 value 값을 모두 string으로 받고 있다.

## 모듈 이름: Reducer.py

### 기능

Mapper.py로부터 생성된 intermediate file이 shuffle & sort 되어 input으로 들어온 것을 process하여 각 Group 별 maximum value를 비교 연산을 이용해 추출해 내 출력 및 저장하는 모듈이다. HDFS의 분산, 병행 처리를 이용해, 각 reduce node들에서 생성된 output은 aggregate되어 하나의 최종 output file로 생성된다.

### 사용 변수

1. Data : Mapper.py 로부터 생성된 intermediate file로부터 한 줄 씩 읽어 저장하는 변수로, 임의의 한Group key에 대해 속해있는 여러 value들을 리스트로 갖고있는 data structure을 저장한다.
2. Firstflag : 최대값 compute 연산을 위해 최대값 변수를 임의의 한 group에 속한 여러 value들 중 첫번째 value로 초기화시키는 코드를, 각 group 별로 한 번씩만 프로그램에서 구동시키기 위해 존재하는 flag이다.
3. Maxval : 임의의 한 group에 대해 최대값을 저장하는 변수로, 초기값은 value 리스트 중 첫번째 value로, 이후 비교 연산을 통해 갱신될 수 있다.
4. Current\_word : Group key를 나타내는 변수로, for in 문을 통해 group의 첫번째 element를 나타내는 변수.
5. Count : 임의의 group 별로 갖고 있는 value 리스트의 하나의 value를 지칭하는 for in 문 변수.

# 코드

**4-1) Mapper.py**

#!/usr/bin/env python

# Use the sys module

import sys

# 'file' in this case is STDIN

def read\_input(file):

# Split each line into words

for line in file:

yield line.split(',')

def main(separator='\t'):

# Read the data using read\_input

data = read\_input(sys.stdin)

# Process each word returned from read\_input

for words in data:

# Process each word

value = words[1][:]

#refine the value if '\n' is concatenated

if value[-1]=='\n':

value = value[:len(value)-1]

#Write key, and value to STDOUT

print('%s%s%f' % (words[0],separator,float(value)))

if \_\_name\_\_ == "\_\_main\_\_":

main()

**4-2) Reducer.py**

from itertools import groupby

from operator import itemgetter

import sys

# 'file' in this case is STDIN

def read\_mapper\_output(file, separator='\t'):

# Go through each line

for line in file:

# Strip out the separator character

yield line.rstrip().split(separator, 1)

def main(separator='\t'):

# Read the data using read\_mapper\_output

data = read\_mapper\_output(sys.stdin, separator=separator)

# Group words and counts into 'group'

# Since MapReduce is a distributed process, each word

# may have multiple counts. 'group' will have all counts

# which can be retrieved using the word as the key.

for current\_word, group in groupby(data, itemgetter(0)):

try:

# For each word, pull the count(s) for the word

# from 'group' and create a total count

firstflag = 0

#firstflag is for initializing maxval into first value from group

for current\_word,count in group :

if firstflag == 0:

firstflag+=1

maxval = float(count)

if(maxval < float(count)):

maxval = float(count)

# Write key and max value to stdout

print("%s%s%f" % (current\_word, separator, maxval))

except ValueError:

# Count was not a number, so do nothing

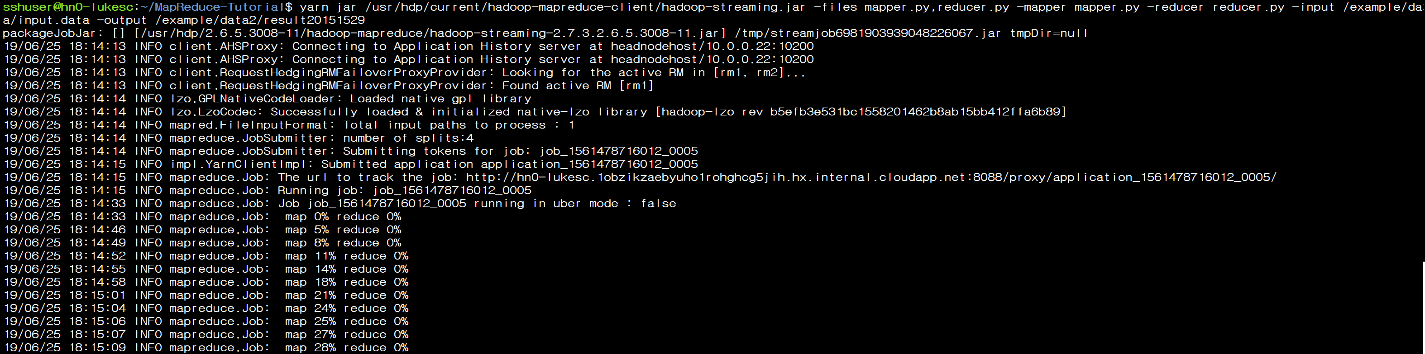
pass

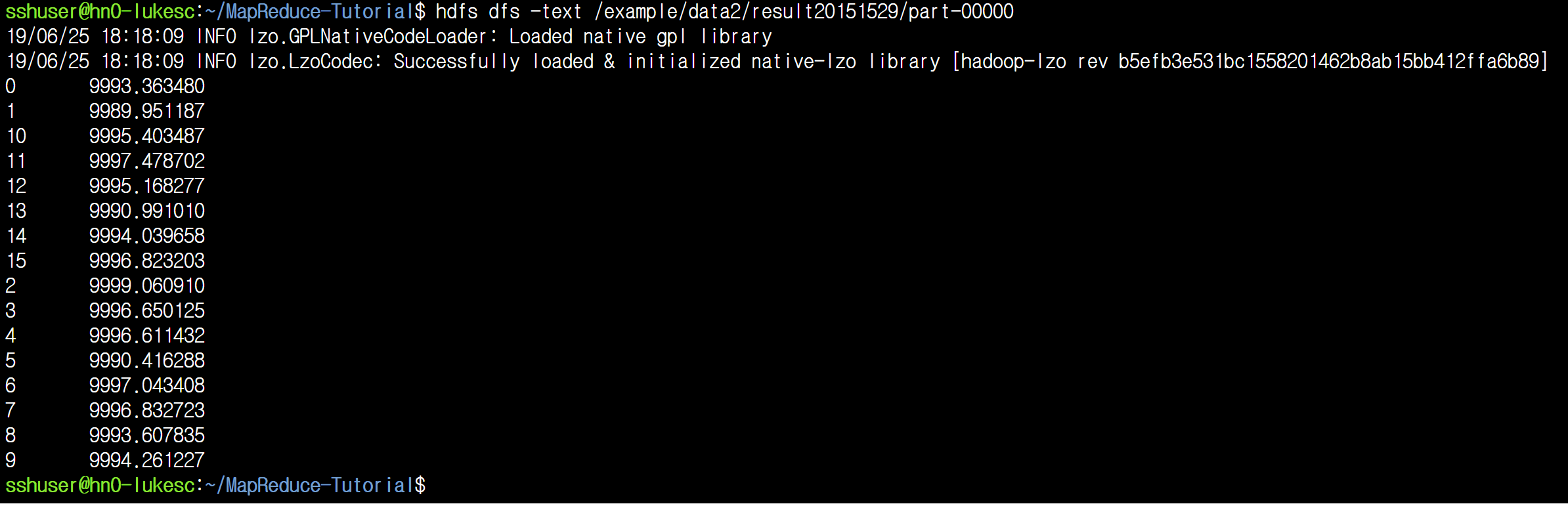
if \_\_name\_\_ == "\_\_main\_\_":

main()

**4-3) 코드 실행 결과 사진 첨부**

-생성한 두 input.data 중 첫 번째 것으로 돌렸을 때





-생성한 두 input.data 중 두 번째 것으로 돌렸을 때

