**2019학년도 1학기 시스템 프로그래밍 Project 6 보고서**

**담당교수: 서강대학교 컴퓨터공학과 소정민**

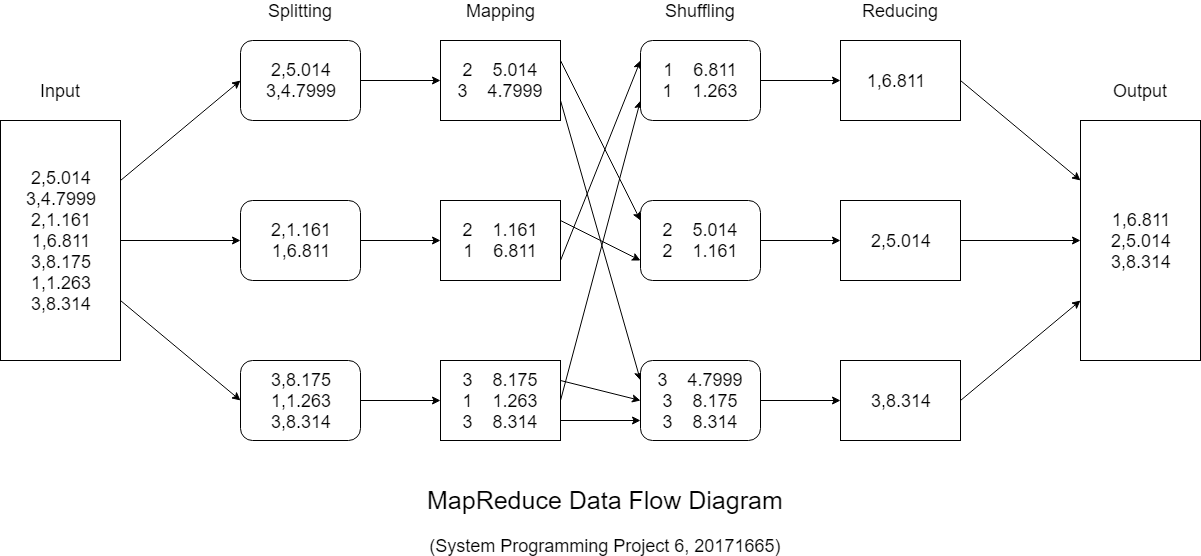
**1. 프로그램 개요**

이번 프로젝트에서는 Hadoop Framework에 관한 이해도를 높이고, Hadoop에서 분산 시스템을 사용하여 수행하는 방법인 MapReduce가 실제로 어떻게 시행되는지를 확인하는 것을 목표로 했다.

MapReduce 방법을 구현하기 위해 필요한 mapper.py와 reducer.py는 Python 3으로 작성되었으며, Azure HDInsight Service를 통해 분산 시스템 환경을 구현했다.

본 프로그램은 generate.py를 통해 생성된 16개 그룹의 임의의 10000000개의 수들 중에서 각 그룹마다 가장 큰 값을 구해서 이를 출력한다.

**2. 프로그램 설명**



Hadoop Framework에서의 MapReduce는 방대한 데이터를 분산 시스템을 사용해서 효율적으로 연산을 처리하기 위한 하나의 기법이다. 일반적으로, 하나의 컴퓨터에서 이번 프로젝트에서 주어진 문제처럼 매우 큰 데이터를 그룹별로 나눠서 각 그룹마다 가장 큰 값을 구하기에는 분산 시스템을 사용할 때보다 상대적으로 부하가 더 생긴다.

분산 시스템에서 MapReduce 기법을 사용할 경우, input을 받으면 독립적인 컴퓨터들이 주어진 일을 수행하기에 overload되거나 under-utilized되지 않도록 input이 적절한 데이터 크기로 나눠져 분배되고, 각 데이터는 mapper.py를 거쳐서 intermediate output이 생성되어 local disk에 저장된다. 본 프로그램에서는 input으로서 “(그룹의 번호),(값)” 형식으로 받으므로 mapper에서 그룹의 번호와 값을 탭 기호로 분리하여 MapReducer가 자동적으로 정렬할 수 있도록 한다.

Mapper를 통해 나온 immediate output을 가지고 Reducer를 통해 독립적인 컴퓨터들이 각 그룹마다 가장 큰 값을 구할 수 있도록 MapReducer가 자동적으로 output 데이터가 그룹별로 묶이도록 정렬한다. 정렬된 데이터들은 병렬적으로 독립적 연산을 수행하는 컴퓨터인 reducer에 각각 대응되며, 같은 그룹 번호를 가진 데이터는 그룹 번호를 마치 hash처럼 사용하여 대응되는 reducer의 input으로 보내진다.

Reducer에서는 각 그룹별로 묶인 데이터들의 값 중에서 가장 큰 값을 구한다. 이를 구현하기 위한 reducer.py 프로그램에서는 탭으로 구분된 그룹의 번호와 값을 받아서 각 그룹의 번호별로 이제까지 읽은 데이터 중에서 가장 큰 값을 저장해놓고 읽는 값을 비교하여 maximum 값을 수정해 가고, 그룹 번호가 바뀌면 이전 그룹 번호와 그에 대응되는 가장 큰 값을 출력하고 나서 다음 그룹 번호에 관해 이를 수행한다. 실제 MapReducer에서는 각 reducer별로 대응된 그룹에서 가장 큰 값을 병렬적으로 연산하여 구하고 reducer들의 결과를 output으로 모아서 출력한다.

이번 Hadoop File System에서는 yarn이라는 명령어를 사용하여 분산 시스템에 업로드 된 mapper와 reducer코드를 input 데이터와 함께 실행하는데, 이는 Hadoop 2.0부터 제공되는 리소스 관리 플랫폼이자 어플리케이션으로서, MapReducer 등 각 어플리케이션을 할당하고 모니터링하는 업무를 수행하는 명렁어다. 수업시간에 배운 JobTracker와 TaskTracker의 역할은 전자는 클러스터 전체의 리소스 관리와 job scheduling 및 모니터링이고 후자는 slave 단계에서 map reduce 작업을 수행하는 것이었다. 그러나 yarn 명령어를 사용하여 YARN 클러스터를 이용하면 이들 대신 Resource Manager와 Node Manager가 사용되어 전자는 client와 통신하고 job 실행에 필요한 node manager를 조정하고, 후자는 실제로 job을 처리하고 결과를 전달해주는 역할을 한다.

**3. 모듈 정의**

**3.1 mapper.py**

시스템 입력으로부터 “(그룹의 번호),(값)” 형식으로 된 여러 데이터를 줄로 나누어서 for loop을 사용하여 입력받는다. For loop 안에서는 입력받은 각 line마다 그룹 번호와 값이 ‘,’로 나뉘어 있으므로 split method로 ‘,’ 앞 뒤를 나눠서 각각 record 이름의 list에 저장한다. record list의 첫 번째 element는 integer data 형식으로 변환하여 key 변수에 저장하고, 두 번째 element는 float data 형식으로 변환하여 value 변수에 저장한다. 이를 print 문을 사용해서 key와 value를 탭으로 구분하여 각각 출력한다.

**3.2 reducer.py**

시스템 입력으로부터 mapper에서 output으로 생성된 데이터를 줄 단위로 나누어서 for loop을 사용하여 입력받는다. 각 줄마다 있을 수 있는 공백을 제거하기 위해 strip method를 호출하여 line 이름의 string 변수에 저장한다. mapper.py의 output은 탭으로 그룹 번호와 값이 구분되어 있으므로 이를 split method를 사용하여 분리해 각각 key와 number라는 변수에 저장한다. key와 number는 문자열이므로 가장 큰 값을 찾는 연산에서 numeric한 데이터로 쓰일 수 있도록 각각 integer와 float 형식으로 변환한다. 만약 numeric한 데이터 형식으로 변환이 불가능한 값이어서 ValueError가 발생하면 다음 데이터 줄을 읽어오도록 try와 except문을 사용했다. 전역변수로서 current\_key, max\_value를 가지고 있는데, current\_key는 현재 가장 큰 값을 구하려는 그룹의 번호를 저장해서 만일 다음 그룹의 번호가 새로 들어오면 현재 그룹의 가장 큰 값을 결정지어 출력하기 위한 목적으로, max\_value는 현재 그룹의 가장 큰 값을 저장하기 위한 용도로 쓰인다. 읽어들인 그룹 번호가 기존 그룹 번호와 다르지 않으면 읽어들인 값이 현재 그룹에서 가장 큰 값인지를 확인하기 위해 max\_value와 number를 대소 비교한다. 만약 number가 max\_value보다 크면 읽은 현재 데이터의 값이 그룹에서 가장 큰 값이므로 max\_value를 number 값으로 수정한다. 만약 다르면 기존의 그룹에서의 큰 값을 다 구한 것이므로 print문을 사용하여 current\_key와 max\_value를 출력한다. 그리고 현재 새로 입력받은 그룹의 번호를 current\_key에 assign하고, 새로운 그룹의 max\_value를 number로 assign한다. For loop이 종료되면 마지막에 남아 있는 그룹의 번호와 가장 큰 값도 출력해야 하므로 print 문을 통해 이를 출력해준다.

**4. 전역 변수 정의**

프로젝트 보고서 작성 안내 사항에 의하여 생략.

**5. 코드**

**5.1** mapper.py

#!/usr/bin/env python

# Use the sys module

import sys

import string

# 'file' in this case is STDIN

for line in sys.stdin:

record = line.split(',')

key = int(record[0])

value = float(record[1])

print('%d\t%s'%(key, value))

**5.2**  reducer.py

#!/usr/bin/env python

# import modules

from sys import stdin, stdout

import sys

current\_key = 0

max\_value = 0

key = None

for var in sys.stdin:

line = var.strip()

key, number = line.split('\t', 1)

try:

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

# from 'group' and create a total count

key = int(key)

number = float(number)

except ValueError:

# Count was not a number, so do nothing

pass

if current\_key == key:

if max\_value < number:

max\_value = number

else:

print("%s,%s" %(current\_key, max\_value))

current\_key = key

max\_value = number

# Write to stdout

print("%s,%s" %(current\_key, max\_value))