Введение

Выполнение практикума осуществляется на любой платформе Hadoop (IBM BigInsights, HortonWorks, Cloudera).

Если домашний компьютер позволяет запускать виртуальные машины дома, можно скачать виртуальную машину и работать с ней.

Если такой возможности нет, то можно использовать удаленный сервер

Использование виртуальной машины

Скачать виртуальную машину

Например, https://www.cloudera.com/downloads/hortonworks-sandbox/hdp.html

Установить Python

```
yum install python-pip
```

Установить MRJob

```
pip install mrjob
```

Соединение с сервером

Установить SSH клиент, например, Putty (https://www.putty.org/)

Установить WinSCP (https://winscp.net/eng/download.php)

Адрес: 83.149.227.84

Порт: 38222

Логин/пароль: student/msucmc19

Создание директории

pwd – посмотреть текущую директорию

/home/student

mkdir briukhov – создать директорию с уникальным именем

cd briukhov – перейти в созданную директорию

ls -l – посмотреть список файлов

mkdir wordcount –

cd wordcount

cp ~/templete/template.py .

Программа подсчета слов

vi template.py – редактировать файл

(команды: i – вставить символы, ESC – закончить вставку, d – удалить символ, :wq – сохранить файл и выйти)

Можно использовать локальный редактор и копировать файл в виртуальную машину с помощью программы WinSCP.

```
from mrjob.job import MRJob
import re

WORD_RE = re.compile(r"[\w']+")
```

```
class MRWordFreqCount(MRJob):

    def mapper(self, _, line):
        for word in WORD_RE.findall(line):
            yield word.lower(), 1

    def combiner(self, word, counts):
        yield word, sum(counts)

    def reducer(self, word, counts):
        yield word, sum(counts)

if __name__ == '__main__':
    MRWordFreqCount.run()
```

mv template.py wordcount.py – переименовать файл
cp ~/templete/Facts.txt . – скопировать текстовый файл
python wordcount.py Facts.txt – запустить подсчет слов в файле Facts.txt

Программа получения наиболее часто встречаемого слова

```
from mrjob.job import MRJob
from mrjob.step import MRStep
import re

WORD_RE = re.compile(r"[\w']+")


class MRMostUsedWord(MRJob):

    def mapper_get_words(self, _, line):
        # yield each word in the line
        for word in WORD_RE.findall(line):
            yield (word.lower(), 1)

    def combiner_count_words(self, word, counts):
        # sum the words we've seen so far
        yield (word, sum(counts))

    def reducer_count_words(self, word, counts):
        # send all (num_occurrences, word) pairs to the same reducer.
        # num_occurrences is so we can easily use Python's max() function.
        yield None, (sum(counts), word)

    # discard the key; it is just None
    def reducer_find_max_word(self, _, word_count_pairs):
        # each item of word_count_pairs is (count, word),
        # so yielding one results in key=counts, value=word
        yield max(word_count_pairs)

    def steps(self):
        return [
            MRStep(mapper=self.mapper_get_words,
                   combiner=self.combiner_count_words,
                   reducer=self.reducer_count_words),
            MRStep(reducer=self.reducer_find_max_word)
        ]


if __name__ == '__main__':
    MRMostUsedWord.run()
```