이름: 박건우 학생번호: 20175130

빅데이터 응용 (Fall 2019)

Midterm (practice) Exam

Oct. 8. 2019

시험시간: 90 분 11:00PM~12:50 PM

Instructions:

1. 이름 및 학생번호 기입 및 확인
2. Exam 문제는 총첨 60점

* 선택형 문제 6 문항: 배점 각 3점씩 총 18점
* 짧은 답형 문제 4 문항: 5 점씩 총 20점
* 그리고 advanced 문제 2문항: 10점+12점 총 22점

총점 60점 배점

1. 시험 시간은 총 110 분
2. 답은 문제지에 직접 기입하며, 시험 종료시 시험지로 제출
3. 답안은 간단 명료하게 작성하며, 문제에서 요구하는 것 이외는 작성할 필요 없음
4. 본시험에서는 아래 셀은 생성 및 실행되었다고 가정합니다.

|  |
| --- |
| from pyspark import SparkContext  sc = SparkContext(master='local[\*]') |

Good luck! ☺

|  |  |
| --- | --- |
| Pyspark operations   * cache() * collect() * reduce(func) * map( func) * count() * first() * take() * filter(func) * distinct() * flatmap(func) * intersection() * subtract() * cartesian() * join() | * sample() * reduceByKey(func) * mapValues(func) * groupByKey(func) * flatMapValues(func) * leftOuterJoin() * rightOuterJoin() * FullOuterJoin() * countByKey() * collectAsMap() * lookup() * groupByKey() * cartesian()   Python operations   * len() * append() * return * yield |

**List of useful operations**

**Choose one question 답안**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **1** | **2** | **3** | **4** | **5** | **6** |
|  |  |  |  |  |  |

1. **ANS:**
2. **ANS:**
3. **ANS:**
4. **ANS:**
5. **(a) ANS:**

**11. (b) ANS:**

1. **(a) ANS:**

**12. (b) ANS:**

**12. (c) ANS:**

**Choose one question 답안**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **1** | **2** | **3** | **4** | **5** | **6** |
|  |  |  |  |  |  |

**Part 1. Choose one questions (각 3점)**

1. 다음 중에서 output이 RDD인 것은?

A

* 1. map()
  2. reduce()
  3. collect()
  4. sample()

1. 다음 중에서 Spark가 지원하지 않는 언어는?

D

* 1. Java
  2. Python
  3. Scala
  4. C++

1. countByKey returns…

A

* 1. A dictionary
  2. A list of tuples
  3. A list of lists
  4. An RDD

1. 다음 중에서 spark lifecycle의 순서는?

C

|  |
| --- |
| 1. Lazy transform into new RDDs 2. Perform actions 3. Create RDD from python list or external file 4. cache() some RDDs |

* 1. 가), 나), 다), 라)
  2. 다), 가), 나), 라)
  3. 다), 가), 라), 나)
  4. 라), 나), 가), 다)

1. 다음 중에서 RDD에 대한 설명인 것은?

D

* 1. RDD는 database이다
  2. RDD는 프로그래밍 기법이다
  3. Spark에서 가장 기본적인 immutable collection of objects
  4. 모두 아니다

1. 다음 중에서 Spark의 특징은?

D

* 1. Fault-tolerance(내결함성)
  2. 다양한 file storage system과의 호환성
  3. In memory computation 지원
  4. 위 모두

**Part 2. Writing questions (각 5점)**

1. 다음 코드를 통해서 첫번째 line과 두번째 line의 결과를 작성하세요.

|  |
| --- |
| text=['you are my sunshine', 'my only sunshine']  text\_file = sc.parallelize(text)  # line 1  print('map:', text\_file.map(lambda line: line.split(" ")).collect())  # line 2  print('flatmap:',text\_file.flatMap(lambda line: line.split(" ")).collect()) |

**ANS:**

**Line1**

map: [['you', 'are', 'my', 'sunshine'], ['my', 'only', 'sunshine']]

**Line2**

flatmap: ['you', 'are', 'my', 'sunshine', 'my', 'only', 'sunshine']

1. 다음 RDD에서 양수의 개수를 출력하는 코드를 완성하세요?

|  |
| --- |
| B=sc.parallelize(range(-10,20))  print(B.[빈칸]) |

**ANS:**

print(B.filter(lambda x:x>0).count())

1. 다음 코드를 보고 빈칸을 완성하세요

|  |
| --- |
| RDD=sc.parallelize([‘abc,de', 'xyz,deeee,ze', 'abc,de,ze,pe', 'xyz,bababa'])  result=RDD.map( [빈칸] ).collect()  print(result)  #output  [('abc', 'de'), ('xyz', 'deeee'), ('abc', 'de'), ('xyz', 'bababa')] |

**ANS:** **result=RDD.map(lambda x:(x.split(","))).collect()**

1. 다음 코드를 보고, output 출력이 나오도록 빈칸을 완성하세요.

|  |
| --- |
| customerRDD ​= ​sc​.​parallelize​( [(​'C1'​,(​'m'​,​'CA'​)),(​'C2'​,(​'f'​,​'CA'​)),(​'C3'​,(​'f'​,​'AZ'​)),(​'C4'​,(​'m'​,​'NY'​)),(​'C5'​,(​'m'​,​'NJ'​))])  purchaseRDD​=​sc​.​parallelize​([(​'C1'​,​'2'​),(​'C3'​,​'4'​),(​'C4'​,​'7'​)])  print([빈칸])  #Output  [  ('C1', (('m', 'CA'), '2')),  ('C4', (('m', 'NY'), '7')),  ('C5', (('m', 'NJ'), None)),  ('C3', (('f', 'AZ'), '4')),  ('C2', (('f', 'CA'), None))  ] |

**ANS:** **print(customerRDD.leftOuterJoin(purchaseRDD).collect())**

**Part 3. Advanced questions**

다음 두 문제는 다음과 같은 dna\_seq.txt 파일을 활용합니다.

dna\_seq.txt

|  |
| --- |
| ATATCCCCGGGAT  ATCGATCGATAT |

|  |
| --- |
| recs = sc.texFile('dna\_seq.txt') |

1. (a) [8점]아래 mapper 라는 함수를 완성하세요. 각 dna sequence에 있는 ‘A’, ’T’, ‘C’, ‘G’의 수를 나타나는 순서대로 tuple로 출력하는 함수입니다. (python code)

|  |  |
| --- | --- |
| def mapper(seq):  freq = dict()     |  | | --- | | 코드작성 |   return kv |
| dnaCount = recs.flatMap(mapper)  dnaCount.collect()  # Output  [('A', 3),  ('T', 3),  ('C', 4),  ('G', 3),  ('A', 4),  ('T', 4),  ('C', 2),  ('G', 2)] |

**ANS:**

1. (b)[2점] 위에서 구한 dnaCount RDD로부터 아래 operation에 대한 출력 결과는? (출력순서는 채점하지 않습니다)

|  |  |
| --- | --- |
| baseCount = dnaCount.reduceByKey(lambda x,y : x+y)  print(baseCount.collect())  # Output   |  | | --- | | 결과값? | |

**ANS:**

아래 파일은 12번 문제에 활용되는 파일입니다.

data.txt

|  |
| --- |
| crazy crazy fox jumped over the fence  crazy fox jumped  the fence is high for fox  crazy fox is smart |

1. (a) [2점] 출력 결과를 쓰세요.

|  |
| --- |
| lines = sc.textFile('data.txt')  print(lines.collect()) |

**ANS:**

12. (b) [8점] 아래 bigram을 찾아서 value 1을 주는 코드를 완성하세요.

|  |  |
| --- | --- |
| |  | | --- | |  |   bigrams = lines.  # Output  [(('crazy', 'crazy'), 1),  (('crazy', 'fox'), 1),  (('fox', 'jumped'), 1),  (('jumped', 'over'), 1),  (('over', 'the'), 1),  (('the', 'fence'), 1),  (('crazy', 'fox'), 1),  (('fox', 'jumped'), 1),  (('the', 'fence'), 1),  (('fence', 'is'), 1),  (('is', 'high'), 1),  (('high', 'for'), 1),  (('for', 'fox'), 1),  (('crazy', 'fox'), 1),  (('fox', 'is'), 1),  (('is', 'smart'), 1)] |

**ANS:**

12. (c) [2점] bigram의 수를 정리하는 함수를 완성하세요

|  |  |
| --- | --- |
| |  | | --- | | 빈칸 |   results = bigrams.  print(results)  #output  [(('jumped', 'over'), 1),  (('over', 'the'), 1),  (('is', 'high'), 1),  (('for', 'fox'), 1),  (('the', 'fence'), 2),  (('fox', 'jumped'), 2),  (('crazy', 'fox'), 3),  (('high', 'for'), 1),  (('fox', 'is'), 1),  (('crazy', 'crazy'), 1),  (('fence', 'is'), 1),  (('is', 'smart'), 1)] |

**ANS:**

수고하셨습니다. sc.stop()!

Additional Homework

1. What is the benefit of RDDs being immutable and having lazy evaluations?

RDD lazy evalusations을 실시할 시 결과가 필요할 떄까지 계산을 미루고 있으며

중간 결과값을 저장할 필요도 없고 메모리 여유도 생기며 데이터를 두 번 검색하지 않고

한 번 검색으로 충분

1. What happens when we do
   * RDD = sc.parallelize(range(10))
   * RDD = RDD.map(lambda x: x\*\*2)

Does the above change the first RDD?

RDD는 생성 시 불변의 객체로 생성되기 때문에 변동은 따로 없다.

1. In the first jupyter cell run

|  |
| --- |
| RDD = sc.parallelize([1, 2, 3, 4, 5])  RDD.toDebugString().decode() |

1. In the next jupyter cell run

|  |
| --- |
| RDD = RDD.map(lambda x: x\*\*2)  RDD.toDebugString().decode() |

What result do you get and why?

첫 번째 결과: '(4) ParallelCollectionRDD[32] at parallelize at PythonRDD.scala:195 []'

두 번째 결과: '(4) PythonRDD[33] at RDD at PythonRDD.scala:53 []\n | ParallelCollectionRDD[32] at parallelize at PythonRDD.scala:195 []'

같은 scala에 collectionRDD[32]에 저장되었음.처음 생성시 불변이며 계산시 딱 필요할때만 공간을 저장하여 한번만 계산하므로 pyhtinRDD[33]에 들어가서 계산이 된 거 같다.