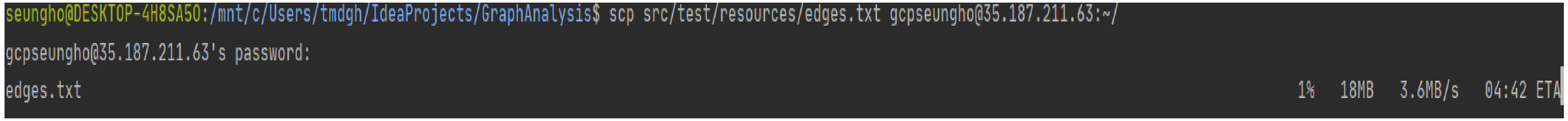
Graph Mining with MapReduce & Spark (LiveJounal Dataset)

20142772 최승호

인스턴스에 파일 업로드

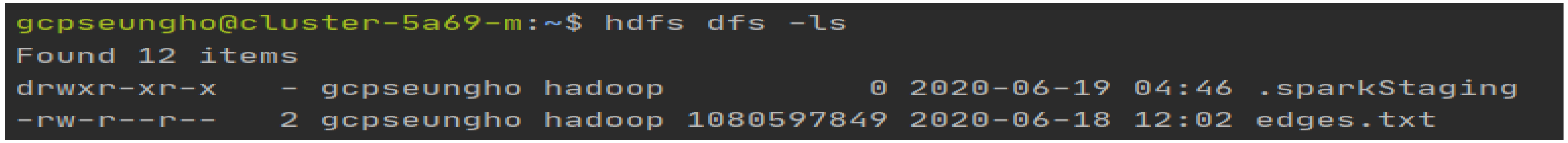
- edge만 남겨두기 위해 데이터 설명 부분 지우고 edges.txt로 파일명 변환

- scp를 이용하여 gcp클라우드에 저장



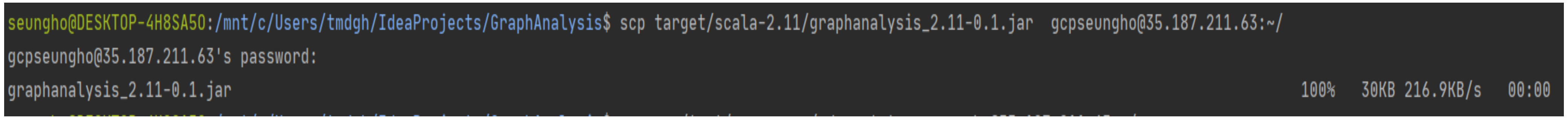
```
seungho@DESKTOP-4H8SA50:/mnt/c/Users/tmdgh/IdeaProjects/GraphAnalysis$ scp src/test/resources/edges.txt gcpseungho@35.187.211.63:~/
gcpseungho@35.187.211.63's password:
edges.txt                                                                                    1%   18MB   3.6MB/s   04:42 ETA
```

- Hdfs에 파일 넣기: hdfs –put edges.txt 확인



```
gcpseungho@cluster-5a69-m:~$ hdfs dfs -ls
Found 12 items
drwxr-xr-x   - gcpseungho hadoop          0 2020-06-19 04:46 .sparkStaging
-rw-r--r--   2 gcpseungho hadoop 1080597849 2020-06-18 12:02 edges.txt
```

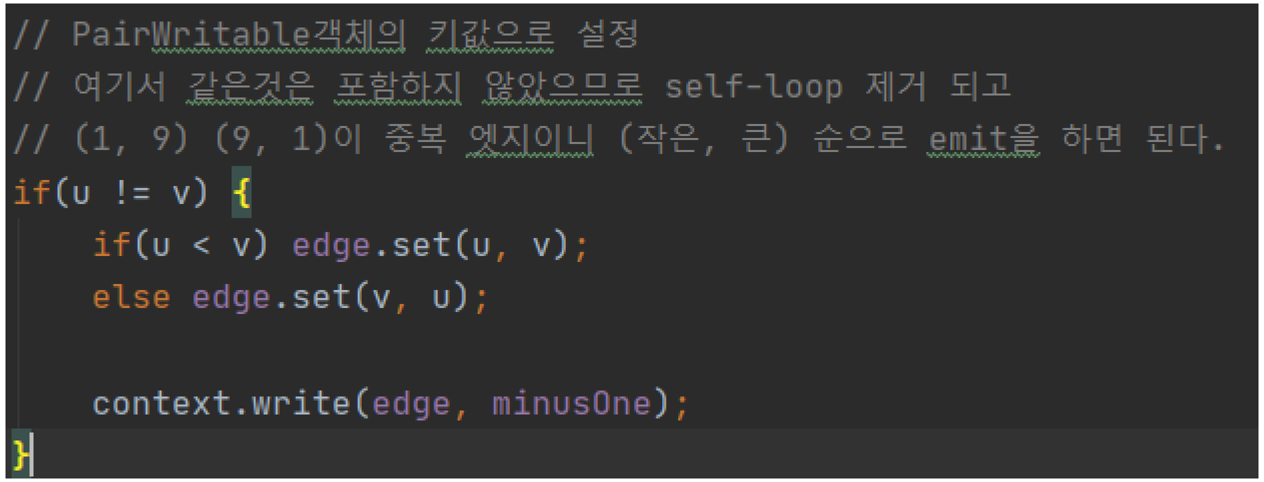
- Package후 Jar 파일올리기



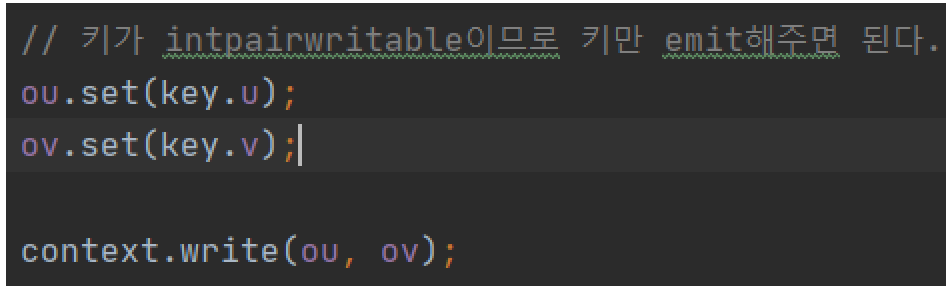
```
seungho@DESKTOP-4H8SA50:/mnt/c/Users/tmdgh/IdeaProjects/GraphAnalysis$ scp target/scala-2.11/graphanalysis_2.11-0.1.jar  gcpseungho@35.187.211.63:~/
gcpseungho@35.187.211.63's password:
graphanalysis_2.11-0.1.jar                                                              100%   30KB 216.9KB/s   00:00
```

Task1

- 중복엣지 제거 + self loop 제거
  - 핵심은 map단계에서 ((u, v), -1) 의 형태로 (u, v)가 키값이 되도록 emit
  - Self loop는 같은지 비교하고 중복은 대소 비교를 통해 하나만 emit
  - Reduce부분에서 key만 emit



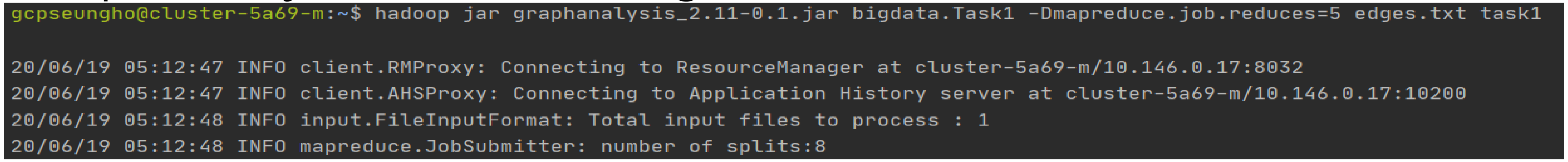
```
// PairWritable객체의 키값으로 설정
// 여기서 같은것은 포함하지 않았으므로 self-loop 제거 되고
// (1, 9) (9, 1)이 중복 엣지이니 (작은, 큰) 순으로 emit을 하면 된다.
if(u != v) {
    if(u < v) edge.set(u, v);
    else edge.set(v, u);

    context.write(edge, minusOne);
}
```



```
// 키가 intpairwritable이므로 키만 emit해주면 된다.
ou.set(key.u);
ov.set(key.v);

context.write(ou, ov);
```

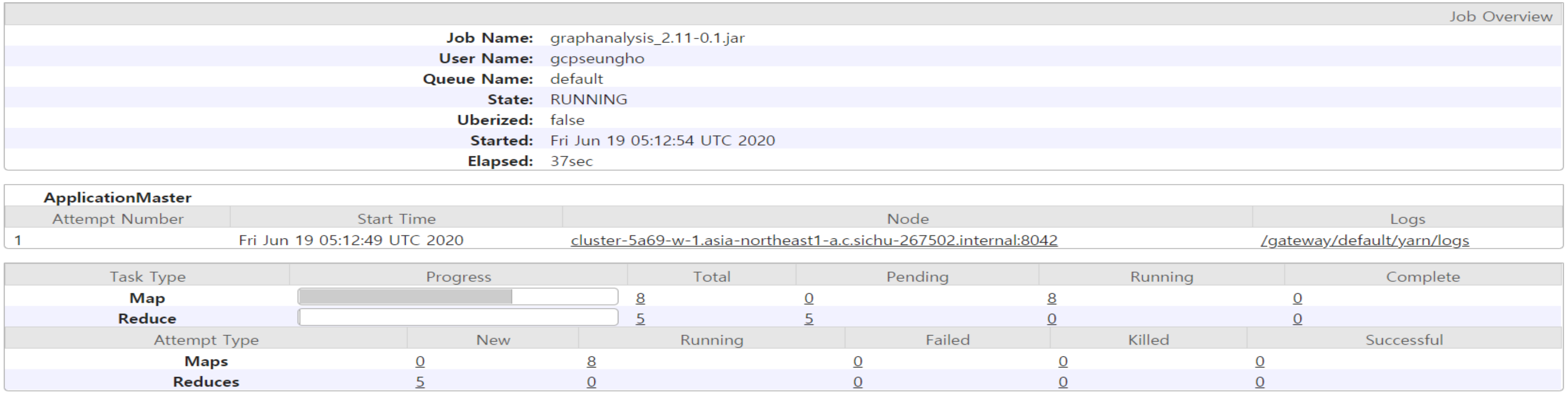
Task1

- 명령어: hadoop jar graphanalysis_2.11-0.1.jar bigdata.Task1 -Dmapreduce.job.reduces=5 edges.txt task1



```
gcpseungho@cluster-5a69-m:~$ hadoop jar graphanalysis_2.11-0.1.jar bigdata.Task1 -Dmapreduce.job.reduces=5 edges.txt task1

20/06/19 05:12:47 INFO client.RMProxy: Connecting to ResourceManager at cluster-5a69-m/10.146.0.17:8032
20/06/19 05:12:47 INFO client.AHSProxy: Connecting to Application History server at cluster-5a69-m/10.146.0.17:10200
20/06/19 05:12:48 INFO input.FileInputFormat: Total input files to process : 1
20/06/19 05:12:48 INFO mapreduce.JobSubmitter: number of splits:8
```

- 과정



Job Overview

Job Name:	graphanalysis_2.11-0.1.jar
User Name:	gcpseungho
Queue Name:	default
State:	RUNNING
Uberized:	false
Started:	Fri Jun 19 05:12:54 UTC 2020
Elapsed:	37sec

ApplicationMaster

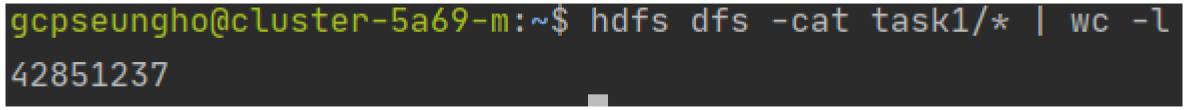
Attempt Number	Start Time	Node	Logs
1	Fri Jun 19 05:12:49 UTC 2020	cluster-5a69-w-1.asia-northeast1-a.c.sichu-267502.internal:8042	/gateway/default/yarn/logs

Task Type	Progress	Total	Pending	Running	Complete
Map		8	0	8	0
Reduce		5	5	0	0

Attempt Type	New	Running	Failed	Killed	Successful
Maps	0	8	0	0	0
Reduces	5	0	0	0	0

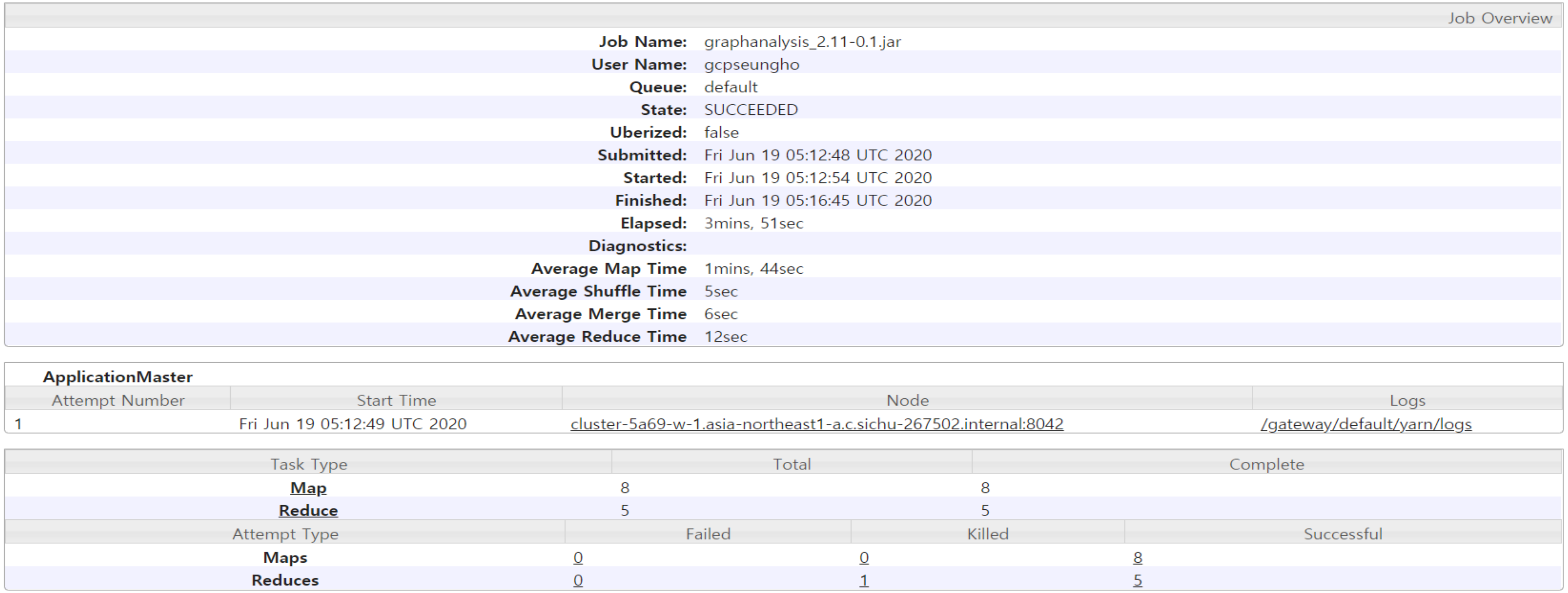
Task1

- 결과



```
gcpseungho@cluster-5a69-m:~$ hdfs dfs -cat task1/* | wc -l
42851237
```

- 엣지개수 68993773 -> 4285137개



Job Overview

Job Name:	graphanalysis_2.11-0.1.jar
User Name:	gcpseungho
Queue:	default
State:	SUCCEEDED
Uberized:	false
Submitted:	Fri Jun 19 05:12:48 UTC 2020
Started:	Fri Jun 19 05:12:54 UTC 2020
Finished:	Fri Jun 19 05:16:45 UTC 2020
Elapsed:	3mins, 51sec
Diagnostics:	
Average Map Time	1mins, 44sec
Average Shuffle Time	5sec
Average Merge Time	6sec
Average Reduce Time	12sec

ApplicationMaster

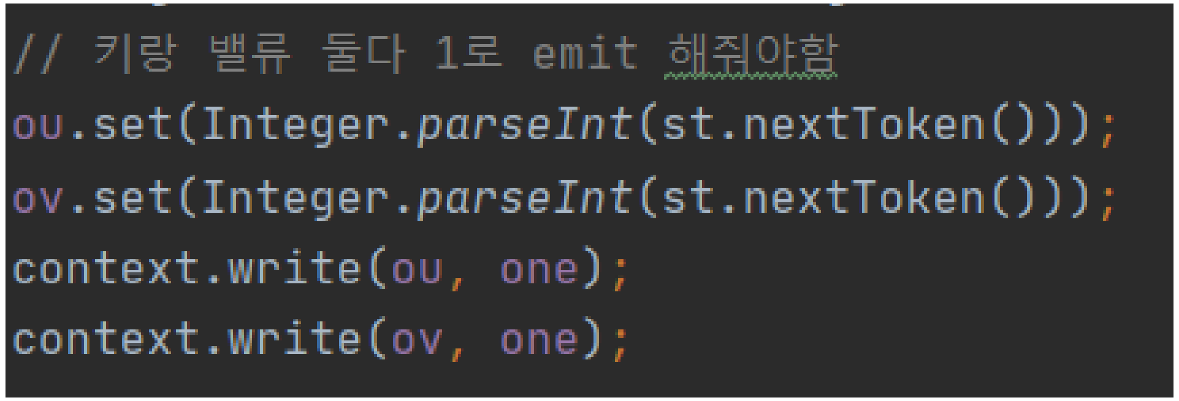
Attempt Number	Start Time	Node	Logs
1	Fri Jun 19 05:12:49 UTC 2020	cluster-5a69-w-1.asia-northeast1-a.c.sichu-267502.internal:8042	/gateway/default/yarn/logs

Task Type	Total	Complete
Map	8	8
Reduce	5	5

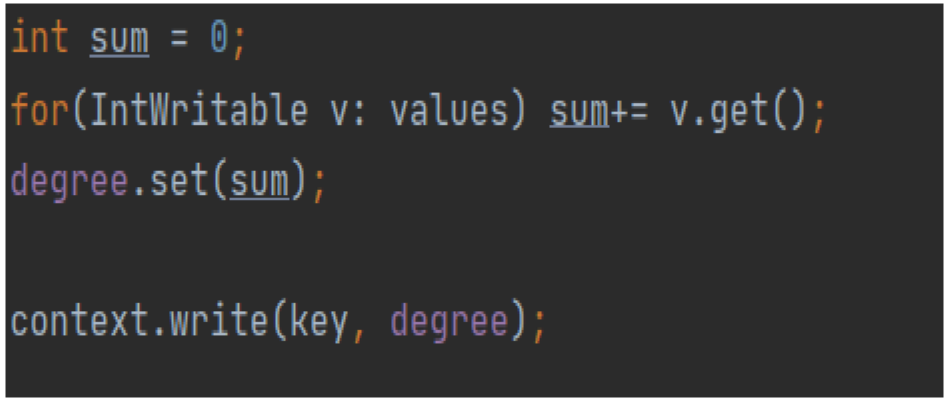
Attempt Type	Failed	Killed	Successful
Maps	0	0	8
Reduces	0	1	5

Task2

- 각 노드의 degree 구하기
  - 핵심은 map단계에서 (u, 1) (v, 1) 둘다 emit 하기
  - Reduce부분에서 key로 summation 해서 emit



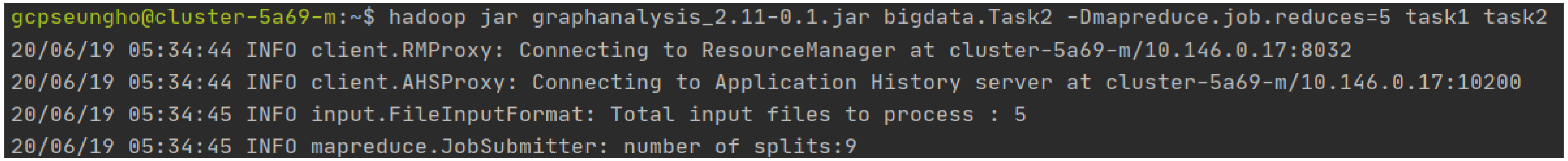
```
// 키랑 밸류 둘다 1로 emit 해줘야함
ou.set(Integer.parseInt(st.nextToken()));
ov.set(Integer.parseInt(st.nextToken()));
context.write(ou, one);
context.write(ov, one);
```



```
int sum = 0;
for(IntWritable v: values) sum+= v.get();
degree.set(sum);

context.write(key, degree);
```

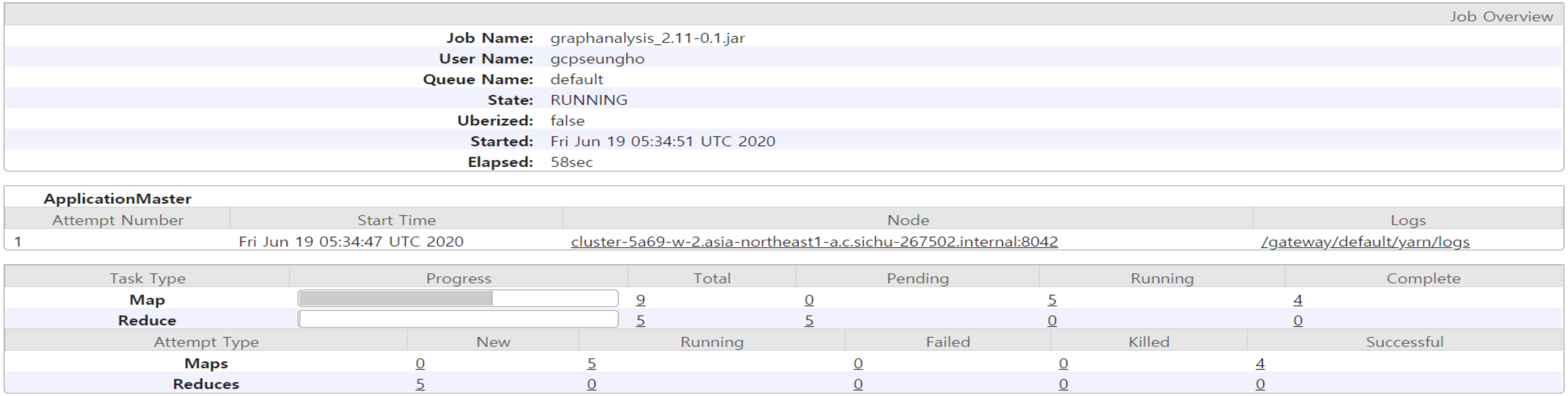
Task2

- 명령어: hadoop jar graphanalysis_2.11-0.1.jar bigdata.Task2 -Dmapreduce.job.reduces=5 task1 task2



```
gcpseungho@cluster-5a69-m:~$ hadoop jar graphanalysis_2.11-0.1.jar bigdata.Task2 -Dmapreduce.job.reduces=5 task1 task2
20/06/19 05:34:44 INFO client.RMProxy: Connecting to ResourceManager at cluster-5a69-m/10.146.0.17:8032
20/06/19 05:34:44 INFO client.AHSProxy: Connecting to Application History server at cluster-5a69-m/10.146.0.17:10200
20/06/19 05:34:45 INFO input.FileInputFormat: Total input files to process : 5
20/06/19 05:34:45 INFO mapreduce.JobSubmitter: number of splits:9
```

- 과정



Job Overview

Job Name:	graphanalysis_2.11-0.1.jar
User Name:	gcpseungho
Queue Name:	default
State:	RUNNING
Uberized:	false
Started:	Fri Jun 19 05:34:51 UTC 2020
Elapsed:	58sec

ApplicationMaster

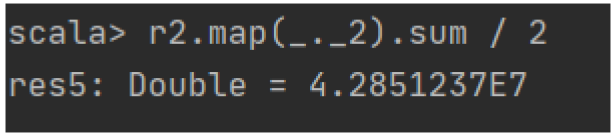
Attempt Number	Start Time	Node	Logs
1	Fri Jun 19 05:34:47 UTC 2020	cluster-5a69-w-2.asia-northeast1-a.c.sichu-267502.internal:8042	/gateway/default/yarn/logs

Task Type	Progress	Total	Pending	Running	Complete
Map		9	0	5	4
Reduce		5	5	0	0

Attempt Type	New	Running	Failed	Killed	Successful
Maps	0	5	0	0	4
Reduces	5	0	0	0	0

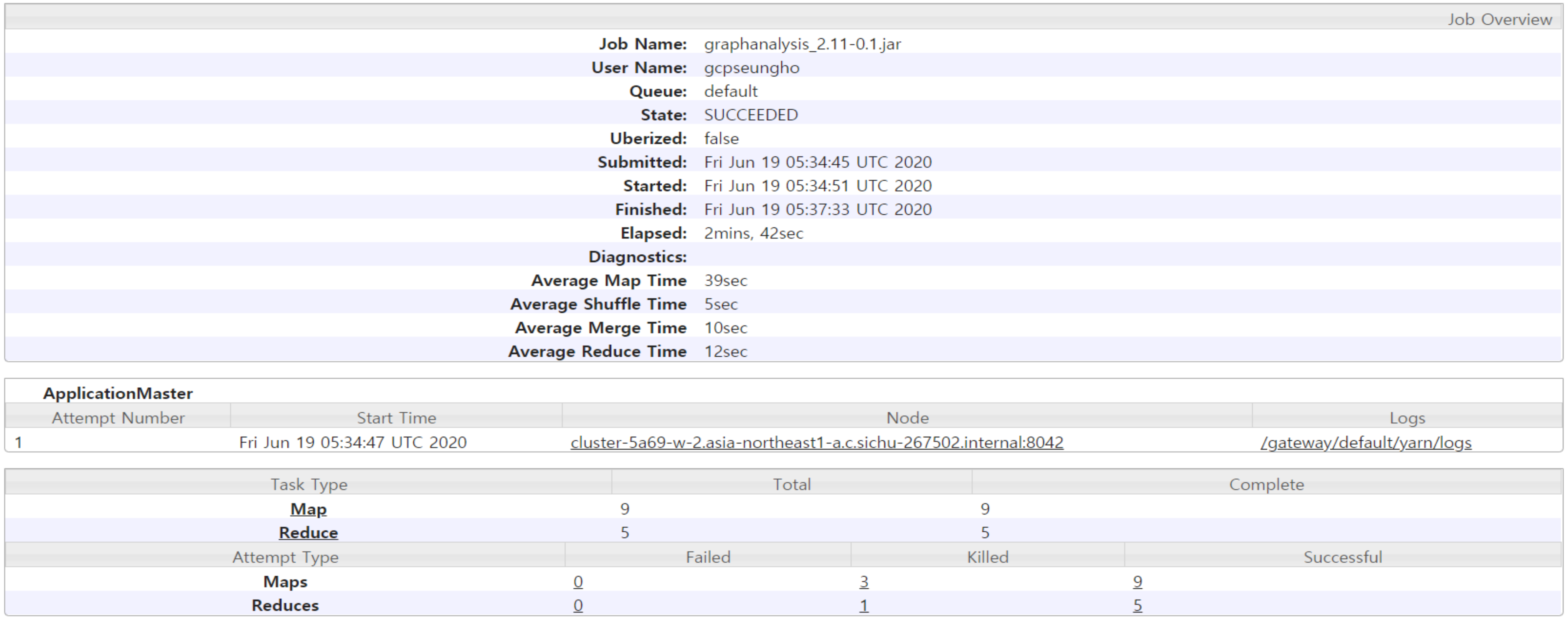
Task2

- 결과



```
scala> r2.map(_._2).sum / 2
res5: Double = 4.2851237E7
```

- Sum(degree) / 2 = edge의 개수 와 같음을 확인하였습니다.



Job Overview

Job Name:	graphanalysis_2.11-0.1.jar
User Name:	gcpseungho
Queue:	default
State:	SUCCEEDED
Uberized:	false
Submitted:	Fri Jun 19 05:34:45 UTC 2020
Started:	Fri Jun 19 05:34:51 UTC 2020
Finished:	Fri Jun 19 05:37:33 UTC 2020
Elapsed:	2mins, 42sec
Diagnostics:	
Average Map Time	39sec
Average Shuffle Time	5sec
Average Merge Time	10sec
Average Reduce Time	12sec

ApplicationMaster

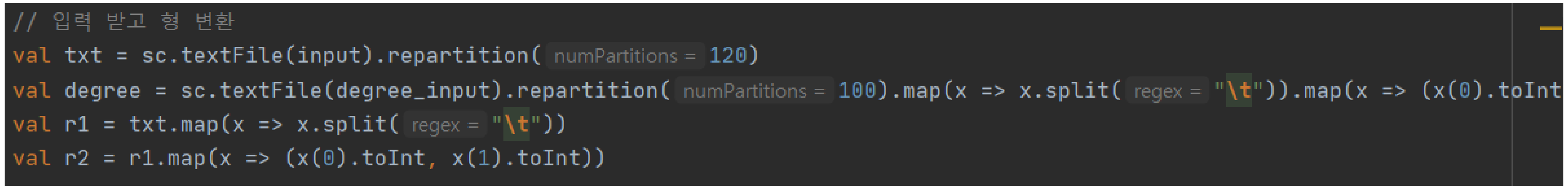
Attempt Number	Start Time	Node	Logs
1	Fri Jun 19 05:34:47 UTC 2020	cluster-5a69-w-2.asia-northeast1-a.c.sichu-267502.internal:8042	/gateway/default/yarn/logs

Task Type	Total	Complete
Map	9	9
Reduce	5	5

Attempt Type	Failed	Killed	Successful
Maps	0	3	9
Reduces	0	1	5

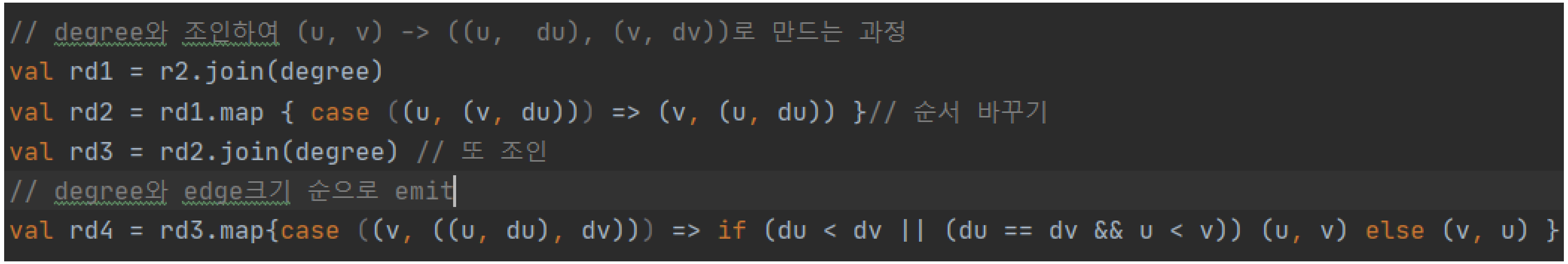
Task3

- 삼각형 개수세기
  - Edge와 degree 인풋받아 형변환



```
// 입력 받고 형 변환
val txt = sc.textFile(input).repartition( numPartitions = 120)
val degree = sc.textFile(degree_input).repartition( numPartitions = 100).map(x => x.split( regex = "\t")).map(x => (x(0).toInt
val r1 = txt.map(x => x.split( regex = "\t"))
val r2 = r1.map(x => (x(0).toInt, x(1).toInt))
```

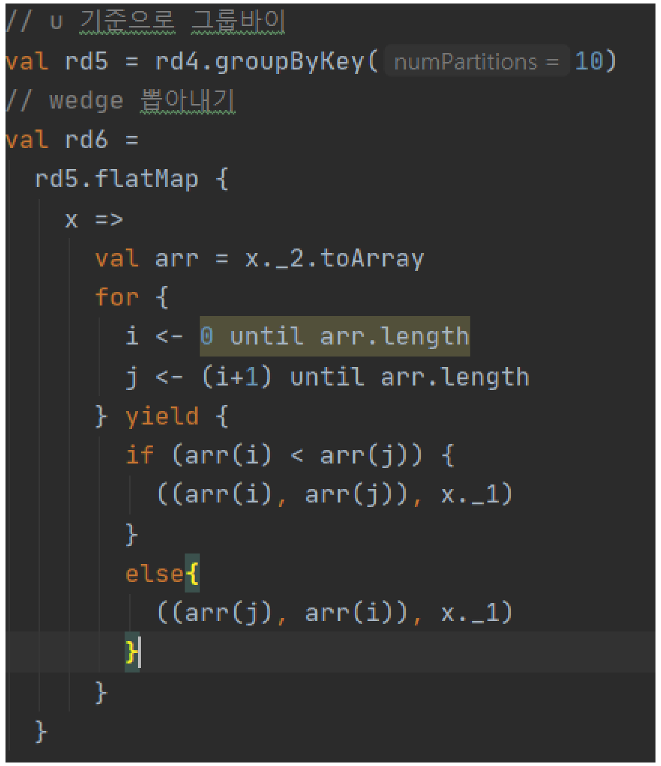
  - (u, v) 를 (u, v, du, dv)로 변환 후 total order에 맞게 u v 다시 생성



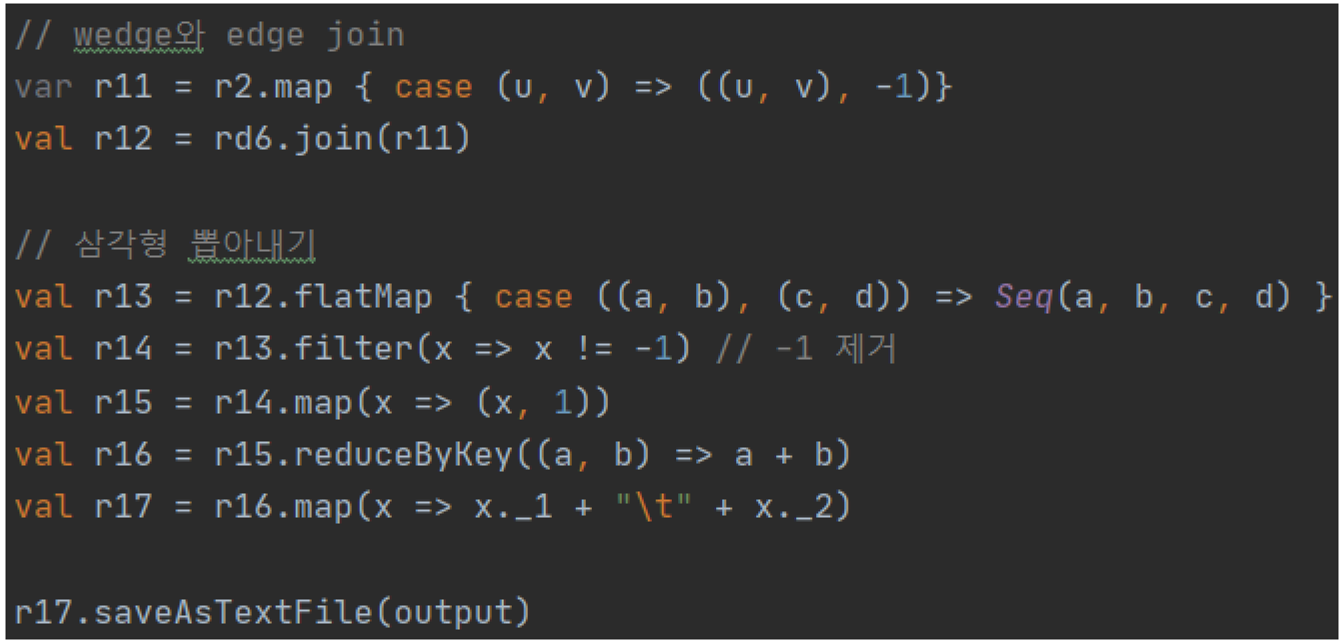
```
// degree와 조인하여 (u, v) -> ((u,  du), (v, dv))로 만드는 과정
val rd1 = r2.join(degree)
val rd2 = rd1.map { case ((u, (v, du))) => (v, (u, du)) }// 순서 바꾸기
val rd3 = rd2.join(degree) // 또 조인
// degree와 edge크기 순으로 emit
val rd4 = rd3.map{case ((v, ((u, du), dv))) => if (du < dv || (du == dv && u < v)) (u, v) else (v, u) }
```

Task3

- 삼각형 개수세기
  - U기준으로 그룹바이 후 wedge 뽑아내기
  - Edge와 조인 후 삼각형 출력



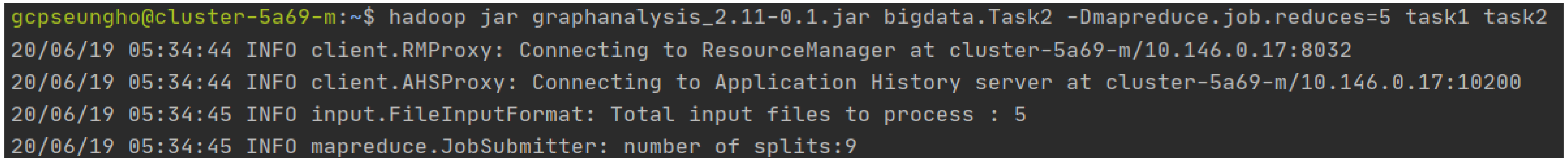
```
// u 기준으로 그룹바이
val rd5 = rd4.groupByKey( numPartitions = 10)
// wedge 뽑아내기
val rd6 =
  rd5.flatMap {
    x =>
      val arr = x._2.toArray
      for {
        i <- 0 until arr.length
        j <- (i+1) until arr.length
      } yield {
        if (arr(i) < arr(j)) {
          ((arr(i), arr(j)), x._1)
        }
        else{
          ((arr(j), arr(i)), x._1)
        }
      }
  }
```



```
// wedge와 edge join
var r11 = r2.map { case (u, v) => ((u, v), -1)}
val r12 = rd6.join(r11)

// 삼각형 뽑아내기
val r13 = r12.flatMap { case ((a, b), (c, d)) => Seq(a, b, c, d) }
val r14 = r13.filter(x => x != -1) // -1 제거
val r15 = r14.map(x => (x, 1))
val r16 = r15.reduceByKey((a, b) => a + b)
val r17 = r16.map(x => x._1 + "\t" + x._2)

r17.saveAsTextFile(output)
```

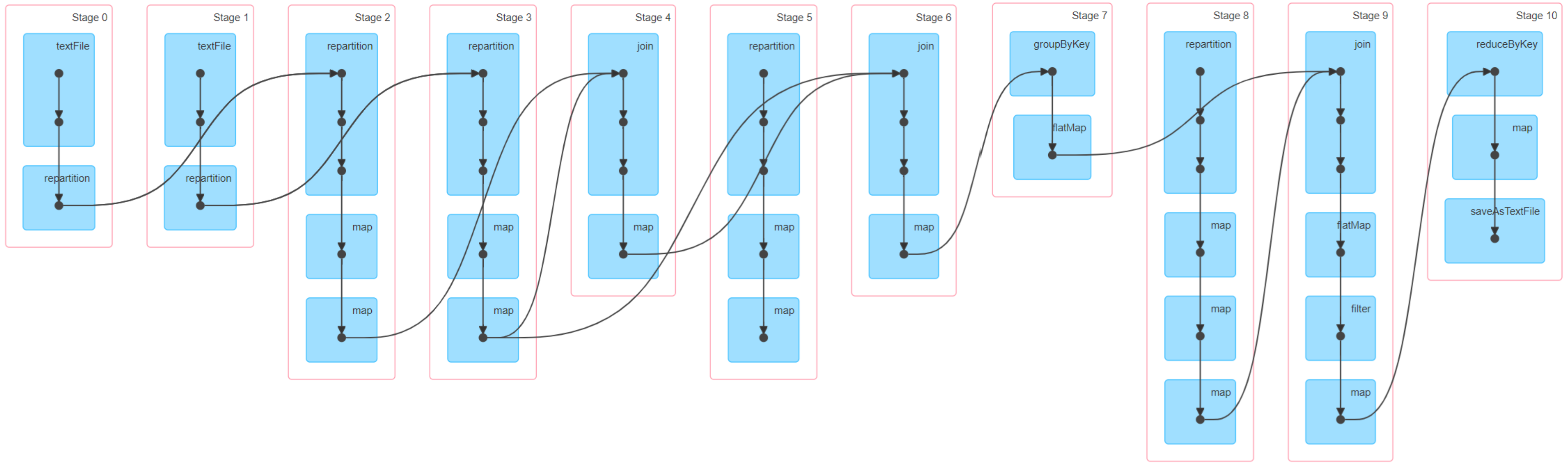
Task3

- 명령어: spark-submit --num-executors 10 --class bigdata.Task3 graphanalysis_2.11-0.1.jar task1 task2 task3



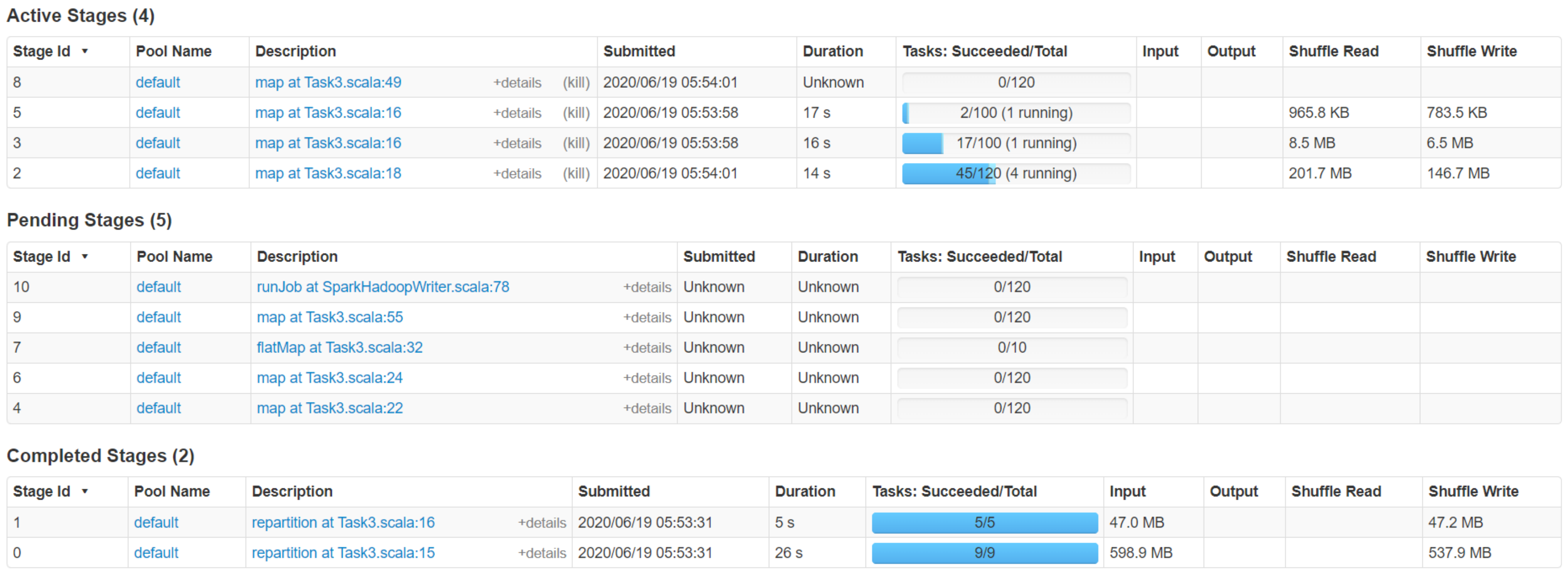
```
gcpseungho@cluster-5a69-m:~$ hadoop jar graphanalysis_2.11-0.1.jar bigdata.Task2 -Dmapreduce.job.reduces=5 task1 task2
20/06/19 05:34:44 INFO client.RMProxy: Connecting to ResourceManager at cluster-5a69-m/10.146.0.17:8032
20/06/19 05:34:44 INFO client.AHSProxy: Connecting to Application History server at cluster-5a69-m/10.146.0.17:10200
20/06/19 05:34:45 INFO input.FileInputFormat: Total input files to process : 5
20/06/19 05:34:45 INFO mapreduce.JobSubmitter: number of splits:9
```

- 과정



Task3

- 과정
- 분산되어 실행되고 있음



Active Stages (4)

Stage Id ▾	Pool Name	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
8	default	map at Task3.scala:49 +details (kill)	2020/06/19 05:54:01	Unknown	0/120				
5	default	map at Task3.scala:16 +details (kill)	2020/06/19 05:53:58	17 s	2/100 (1 running)			965.8 KB	783.5 KB
3	default	map at Task3.scala:16 +details (kill)	2020/06/19 05:53:58	16 s	17/100 (1 running)			8.5 MB	6.5 MB
2	default	map at Task3.scala:18 +details (kill)	2020/06/19 05:54:01	14 s	45/120 (4 running)			201.7 MB	146.7 MB

Pending Stages (5)

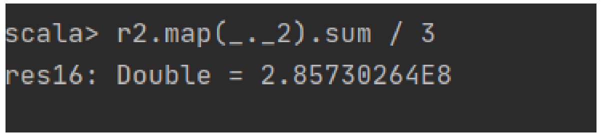
Stage Id ▾	Pool Name	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
10	default	runJob at SparkHadoopWriter.scala:78 +details	Unknown	Unknown	0/120				
9	default	map at Task3.scala:55 +details	Unknown	Unknown	0/120				
7	default	flatMap at Task3.scala:32 +details	Unknown	Unknown	0/10				
6	default	map at Task3.scala:24 +details	Unknown	Unknown	0/120				
4	default	map at Task3.scala:22 +details	Unknown	Unknown	0/120				

Completed Stages (2)

Stage Id ▾	Pool Name	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
1	default	repartition at Task3.scala:16 +details	2020/06/19 05:53:31	5 s	5/5	47.0 MB			47.2 MB
0	default	repartition at Task3.scala:15 +details	2020/06/19 05:53:31	26 s	9/9	598.9 MB			537.9 MB

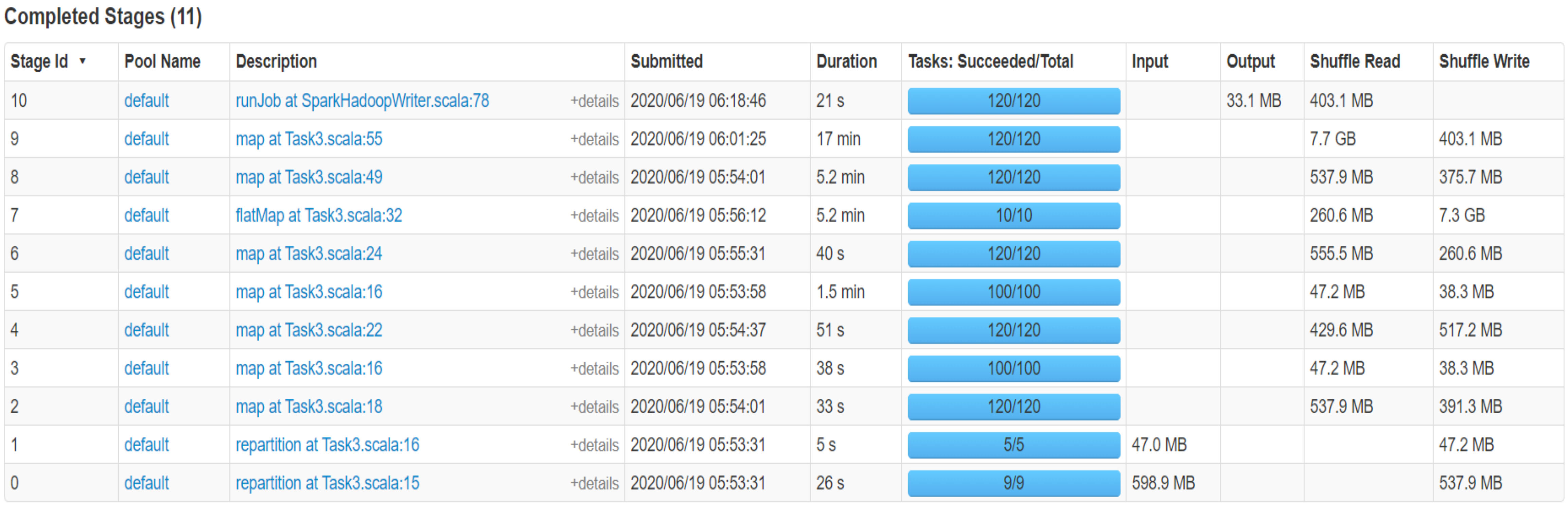
Task3

- 결과



```
scala> r2.map(_._2).sum / 3
res16: Double = 2.85730264E8
```

- 삼각형의 개수가 285730264개인 것을 확인하였습니다.

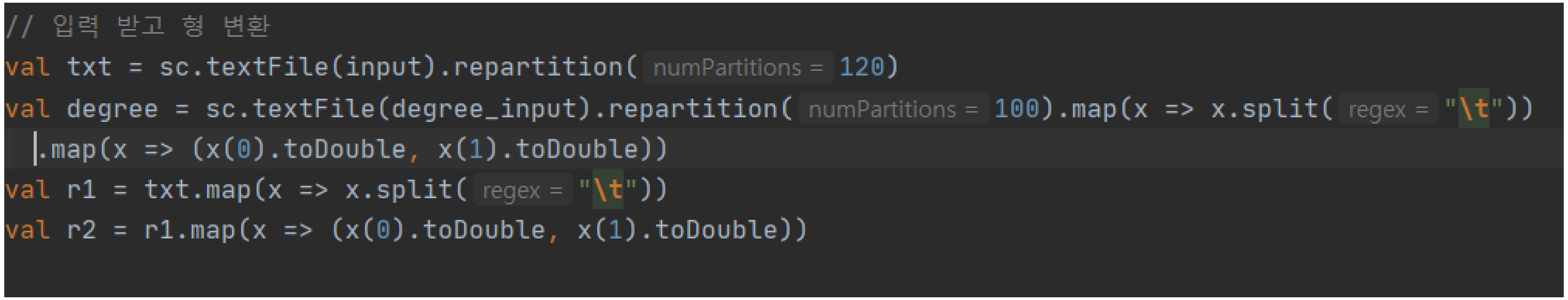


Completed Stages (11)

Stage Id ▾	Pool Name	Description		Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
10	default	runJob at SparkHadoopWriter.scala:78	+details	2020/06/19 06:18:46	21 s	120/120		33.1 MB	403.1 MB	
9	default	map at Task3.scala:55	+details	2020/06/19 06:01:25	17 min	120/120			7.7 GB	403.1 MB
8	default	map at Task3.scala:49	+details	2020/06/19 05:54:01	5.2 min	120/120			537.9 MB	375.7 MB
7	default	flatMap at Task3.scala:32	+details	2020/06/19 05:56:12	5.2 min	10/10			260.6 MB	7.3 GB
6	default	map at Task3.scala:24	+details	2020/06/19 05:55:31	40 s	120/120			555.5 MB	260.6 MB
5	default	map at Task3.scala:16	+details	2020/06/19 05:53:58	1.5 min	100/100			47.2 MB	38.3 MB
4	default	map at Task3.scala:22	+details	2020/06/19 05:54:37	51 s	120/120			429.6 MB	517.2 MB
3	default	map at Task3.scala:16	+details	2020/06/19 05:53:58	38 s	100/100			47.2 MB	38.3 MB
2	default	map at Task3.scala:18	+details	2020/06/19 05:54:01	33 s	120/120			537.9 MB	391.3 MB
1	default	repartition at Task3.scala:16	+details	2020/06/19 05:53:31	5 s	5/5	47.0 MB			47.2 MB
0	default	repartition at Task3.scala:15	+details	2020/06/19 05:53:31	26 s	9/9	598.9 MB			537.9 MB

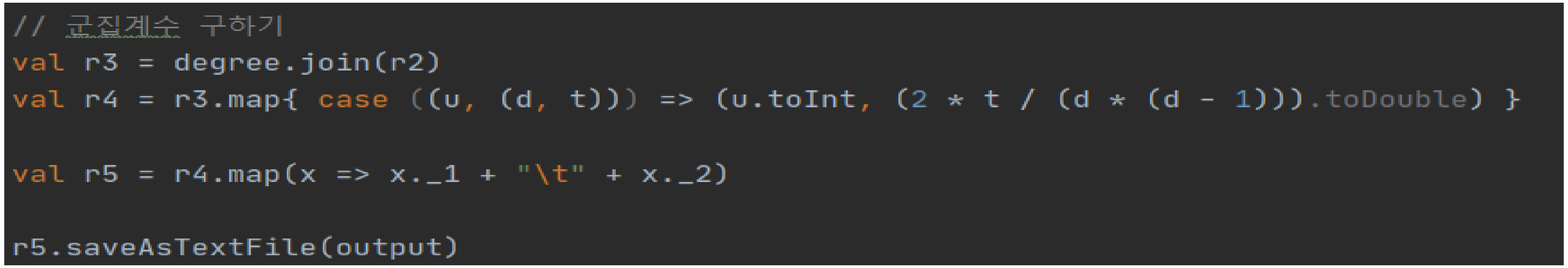
Task4

- 군집계수 구하기
  - 삼각형 개수와 degree 인풋받아 double로 형변환(나누기 연산에 필요)



```
// 입력 받고 형 변환
val txt = sc.textFile(input).repartition( numPartitions = 120)
val degree = sc.textFile(degree_input).repartition( numPartitions = 100).map(x => x.split( regex = "\t"))
  .map(x => (x(0).toDouble, x(1).toDouble))
val r1 = txt.map(x => x.split( regex = "\t"))
val r2 = r1.map(x => (x(0).toDouble, x(1).toDouble))
```

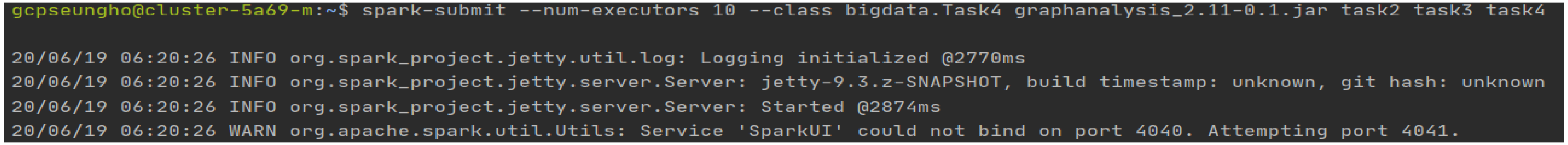
  - 2 * 삼각형의 개수 / degree (degree – 1) 식 적용



```
// 군집계수 구하기
val r3 = degree.join(r2)
val r4 = r3.map{ case ((u, (d, t))) => (u.toInt, (2 * t / (d * (d - 1))).toDouble) }

val r5 = r4.map(x => x._1 + "\t" + x._2)

r5.saveAsTextFile(output)
```

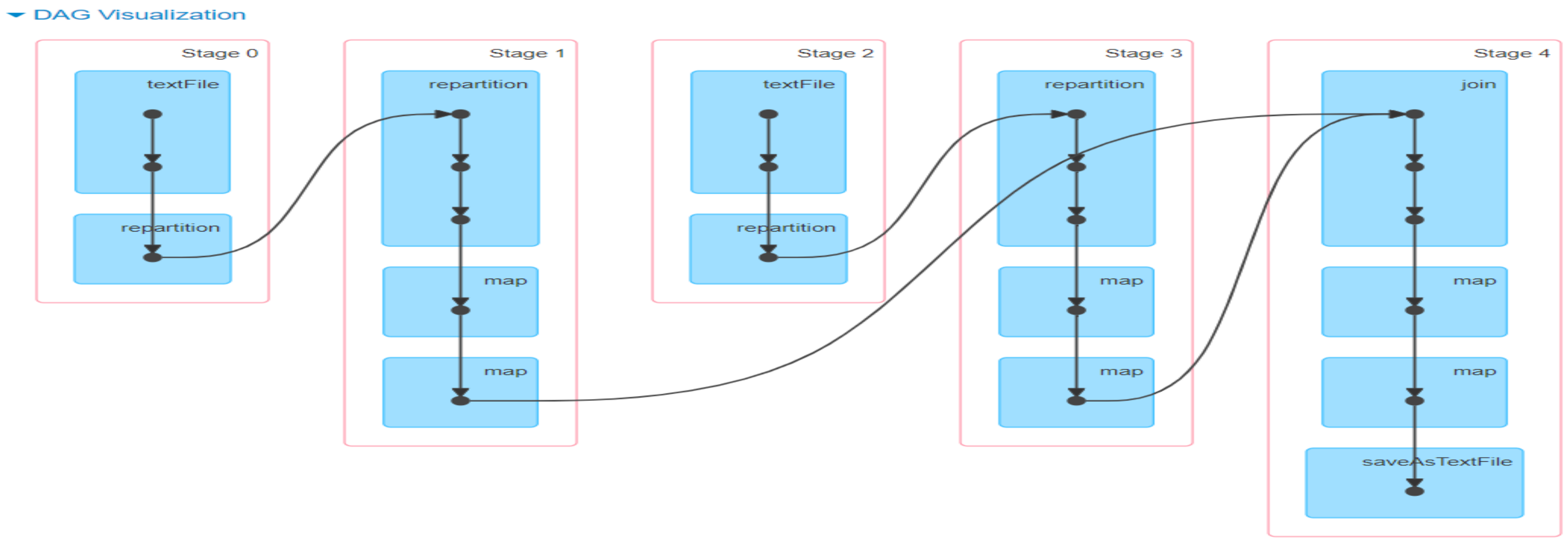
Task4

- 명령어: spark-submit --num-executors 10 --class bigdata.Task4 graphanalysis_2.11-0.1.jar task2 task3 task4



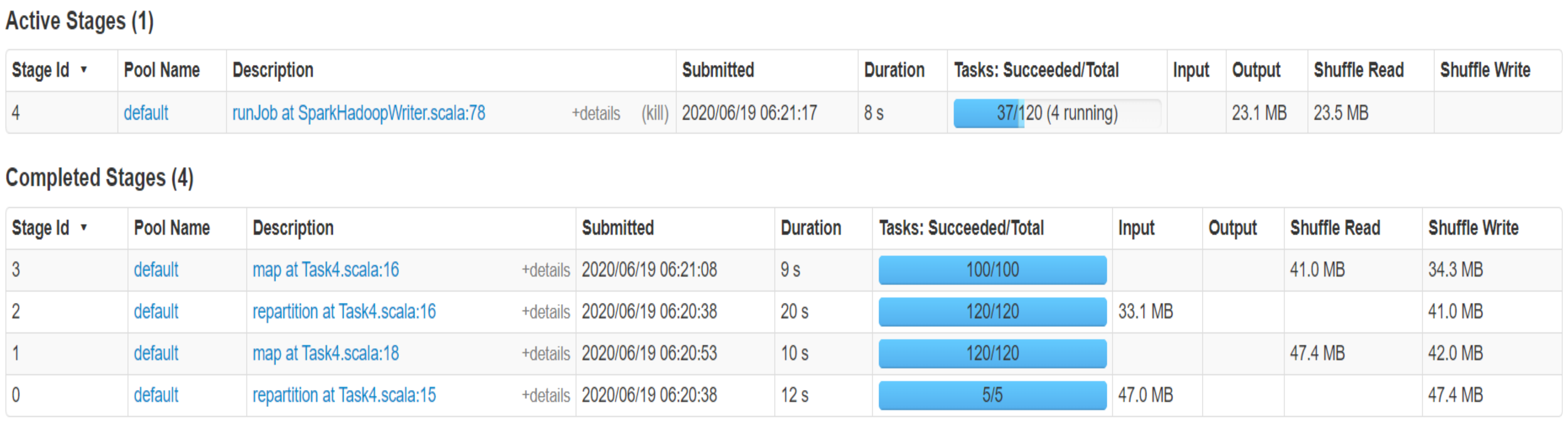
```
gcpseungho@cluster-5a69-m:~$ spark-submit --num-executors 10 --class bigdata.Task4 graphanalysis_2.11-0.1.jar task2 task3 task4

20/06/19 06:20:26 INFO org.spark_project.jetty.util.log: Logging initialized @2770ms
20/06/19 06:20:26 INFO org.spark_project.jetty.server.Server: jetty-9.3.z-SNAPSHOT, build timestamp: unknown, git hash: unknown
20/06/19 06:20:26 INFO org.spark_project.jetty.server.Server: Started @2874ms
20/06/19 06:20:26 WARN org.apache.spark.util.Utils: Service 'SparkUI' could not bind on port 4040. Attempting port 4041.
```

- 과정



Task4

- 과정
- 분산되어 실행되고 있음



Active Stages (1)

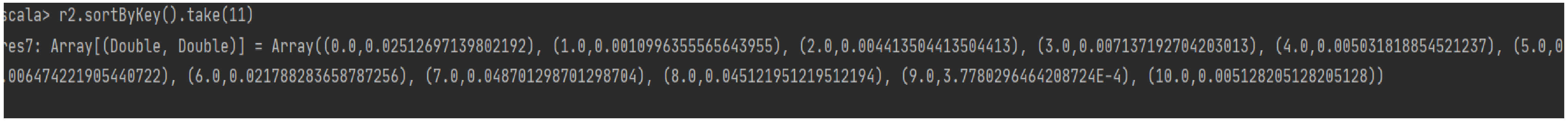
Stage Id ▾	Pool Name	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
4	default	runJob at SparkHadoopWriter.scala:78 +details (kill)	2020/06/19 06:21:17	8 s	37/120 (4 running)		23.1 MB	23.5 MB	

Completed Stages (4)

Stage Id ▾	Pool Name	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
3	default	map at Task4.scala:16 +details	2020/06/19 06:21:08	9 s	100/100			41.0 MB	34.3 MB
2	default	repartition at Task4.scala:16 +details	2020/06/19 06:20:38	20 s	120/120	33.1 MB			41.0 MB
1	default	map at Task4.scala:18 +details	2020/06/19 06:20:53	10 s	120/120			47.4 MB	42.0 MB
0	default	repartition at Task4.scala:15 +details	2020/06/19 06:20:38	12 s	5/5	47.0 MB			47.4 MB

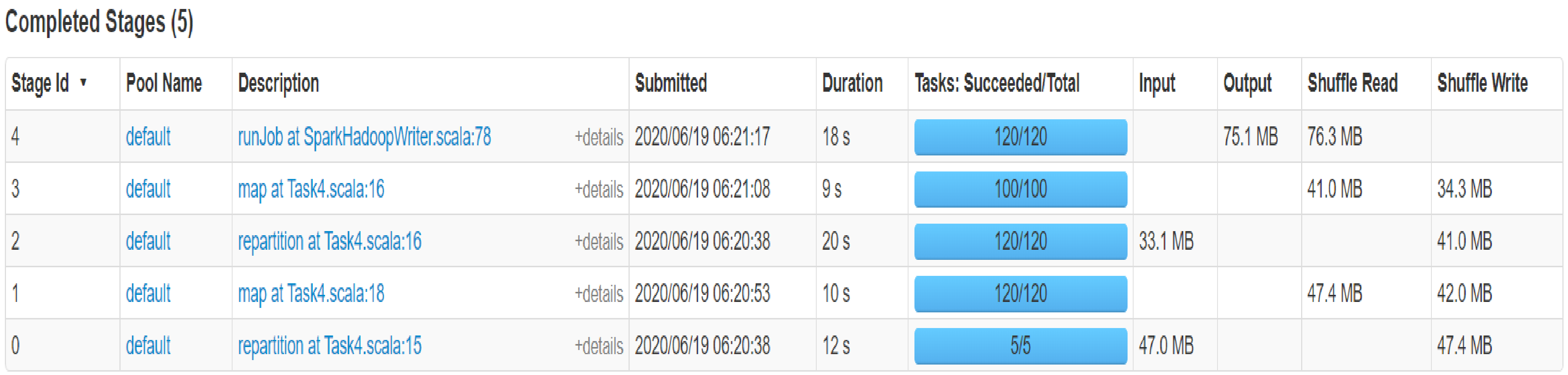
Task4

- 결과



```
scala> r2.sortByKey().take(11)
res7: Array[(Double, Double)] = Array((0.0,0.02512697139802192), (1.0,0.0010996355565643955), (2.0,0.004413504413504413), (3.0,0.007137192704203013), (4.0,0.005031818854521237), (5.0,0
.006474221905440722), (6.0,0.021788283658787256), (7.0,0.048701298701298704), (8.0,0.045121951219512194), (9.0,3.7780296464208724E-4), (10.0,0.005128205128205128))
```

- 예시로 0 ~ 10까지의 coefficient를 출력해보았습니다.

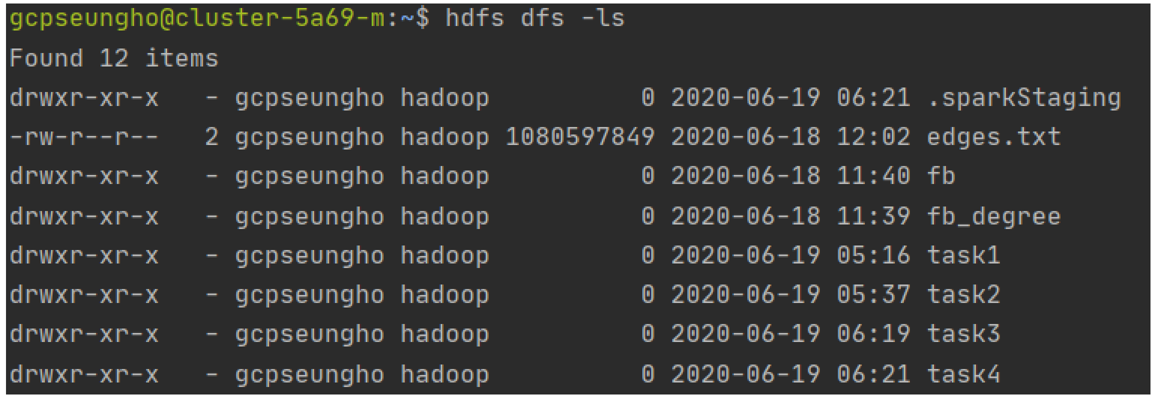


Completed Stages (5)

Stage Id ▾	Pool Name	Description		Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
4	default	runJob at SparkHadoopWriter.scala:78	+details	2020/06/19 06:21:17	18 s	120/120		75.1 MB	76.3 MB	
3	default	map at Task4.scala:16	+details	2020/06/19 06:21:08	9 s	100/100			41.0 MB	34.3 MB
2	default	repartition at Task4.scala:16	+details	2020/06/19 06:20:38	20 s	120/120	33.1 MB			41.0 MB
1	default	map at Task4.scala:18	+details	2020/06/19 06:20:53	10 s	120/120			47.4 MB	42.0 MB
0	default	repartition at Task4.scala:15	+details	2020/06/19 06:20:38	12 s	5/5	47.0 MB			47.4 MB

최종결과

- Hdfs에 task1, task2, task3, task4 잘 저장되어있음을 볼 수 있었다.



```
gcpseungho@cluster-5a69-m:~$ hdfs dfs -ls
Found 12 items
drwxr-xr-x   - gcpseungho hadoop          0 2020-06-19 06:21 .sparkStaging
-rw-r--r--   2 gcpseungho hadoop 1080597849 2020-06-18 12:02 edges.txt
drwxr-xr-x   - gcpseungho hadoop          0 2020-06-18 11:40 fb
drwxr-xr-x   - gcpseungho hadoop          0 2020-06-18 11:39 fb_degree
drwxr-xr-x   - gcpseungho hadoop          0 2020-06-19 05:16 task1
drwxr-xr-x   - gcpseungho hadoop          0 2020-06-19 05:37 task2
drwxr-xr-x   - gcpseungho hadoop          0 2020-06-19 06:19 task3
drwxr-xr-x   - gcpseungho hadoop          0 2020-06-19 06:21 task4
```

- Mapreduce방식으로 사고하는 것과 scala문법이 힘들었다....
- 분산저장을 위하여 고려해야할 점들이 많은 것을 깨달았다.
  - Commutative 하고 associative한 연산들로 구성
  - 중간 결과물의 최소화 작업(wedge)
  - 메모리 초과를 detecting하고 해결해나가는 과정