Πολυτεχνείο Κρήτης

Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών

Αναφορά Εργασίας

Συγγραφέας:
Καπελώνης Χαρίλαος

Υπεύθυνος Καθηγητής:
Ν. Γιατράκος

Βοηθός Εργαστηρίου:
Χ. Τσιναράκη

Η εργασία κατατέθηκε για το μάθημα:

Συναρτησιακός Προγραμματισμός, Αναλυτική & Εφαρμογές

Εαρινό Εξάμηνο 2024

Περιεχόμενα

Σενάριο εφαρμογής 1: Ανάλυση ειδήσεων Reuters με χρήση Spark RDDs (3 μονάδες)

Το dataset αφορά ειδήσεις (documents) που αφενός ανήκουν σε κάποια/-ες κατηγορίες (categories) και αφετέρου έχουν κάποιους όρους (terms) σημασίας.

Το ενδιαφέρον της ανάλυσης επικεντρώνεται στο να υπολογισθεί η μετρική του Jaccard Index που, χρησιμοποιώντας τα documents, ποσοτικοποιεί τη σχέση που έχει μία κατηγορία με έναν όρο.

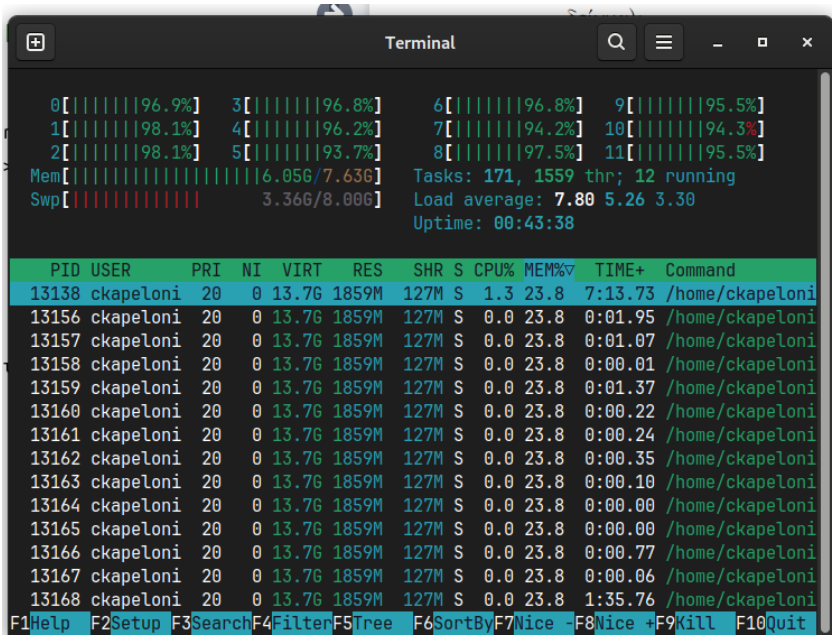
Τα δεδομένα είναι μεγάλα και, καθώς πρέπει να υπολογισθεί η παραπάνω μετρική για κάθε δυνατό ζευγάρι κατηγορίας - όρου, το πρόβλημα είναι απαιτητικό από άποψη υπολογιστικής ισχύος και μνήμης, άρα χρειάζεται πολύ προσεκτική και αποδοτική εφαρμογή HOF (transformations) του Apache Spark.

1.1 Configurations

Ακολουθούν οι ρυθμίσεις για να τρέχει `locally` η εφαρμογή.

```
// LOCAL CONFIGS
val spark = SparkSession.builder
 .master("local[*]") // local[*] means "use as many threads as the
    number of processors available to the Java virtual machine"
 .appName("Reuters")
 .getOrCreate()
val sc = spark.sparkContext
val categoriesPath =
    "hdfs://localhost:9000/Reuters/rcv1-v2.topics.qrels"
val termsPath =
    "hdfs://localhost:9000/Reuters/lyrl2004_vectors_*.dat" // '*'
    means match anything
val stemsPath =
    "hdfs://localhost:9000/Reuters/stem.termid.idf.map.txt"
val outputPath = "hdfs://localhost:9000/Reuters/output"
```

Εδώ φαίνεται ότι χρησιμοποιούνται και οι 12 πυρήνες σε (σχεδόν) πλήρη ισχύ, αφού τέθηκε `.master("local[*]")`:



Και εδώ φαίνεται η εφαρμογή στο `History Server`:

3.5.1	local-1717839252687	Reuters	2024-06-08 12:34:10	2024-06-08 12:36:13	2.0 min

Το `Spark` χώρισε την εφαρμογή σε 4 `Jobs` (πώς αποφασίστηκε ο αριθμός 4;), φόρτωση κατηγοριών (0), όρων (1), ριζών όρων (2) και υπολογισμός `Jaccard Index` καθώς και αποθήκευση του αρχείου (3):

Completed Jobs (4)

Page: 1 — 1 Pages. Jump to 1 . Show 100 items in a page. Go

Job Id ▾	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
3	runJob at SparkHadoopWriter.scala:83 runJob at SparkHadoopWriter.scala:83	2024/06/08 12:34:35	1.6 min	9/9	103/103
2	collectAsMap at Reuters.scala:57 collectAsMap at Reuters.scala:57	2024/06/08 12:34:35	0.3 s	1/1	2/2
1	countByValue at Reuters.scala:54 countByValue at Reuters.scala:54	2024/06/08 12:34:18	17 s	2/2	26/26
0	countByValue at Reuters.scala:40 countByValue at Reuters.scala:40	2024/06/08 12:34:15	3 s	2/2	4/4

Page: 1 — 1 Pages. Jump to 1 . Show 100 items in a page. Go

Stages for All Jobs

Completed Stages: 14

Completed Stages (14)

Page: 1 — 1 Pages. Jump to 1 . Show 100 items in a page. Go

Stage Id ▾	Description		Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
13	runJob at SparkHadoopWriter.scala:83	+details	2024/06/08 12:36:08	5 s	13/13		46.8 MiB	18.9 MiB	
12	parallelize at Reuters.scala:61	+details	2024/06/08 12:34:35	0.2 s	12/12				519.2 KiB
11	map at Reuters.scala:71	+details	2024/06/08 12:36:07	0.7 s	13/13			12.7 MiB	17.8 MiB
10	parallelize at Reuters.scala:40	+details	2024/06/08 12:34:35	0.7 s	12/12				9.8 KiB
9	map at Reuters.scala:69	+details	2024/06/08 12:36:03	4 s	13/13			105.6 MiB	12.7 MiB
8	map at Reuters.scala:67	+details	2024/06/08 12:34:50	1.2 min	13/13			373.3 MiB	105.6 MiB
7	map at Reuters.scala:36	+details	2024/06/08 12:34:35	3 s	2/2	33.7 MiB			15.7 MiB
6	flatMap at Reuters.scala:49	+details	2024/06/08 12:34:35	14 s	13/13	1420.9 MiB			357.7 MiB
5	parallelize at Reuters.scala:54	+details	2024/06/08 12:34:35	0.2 s	12/12				526.4 KiB
4	collectAsMap at Reuters.scala:57	+details	2024/06/08 12:34:35	0.3 s	2/2	1393.0 KiB			
3	countByValue at Reuters.scala:54	+details	2024/06/08 12:34:34	0.3 s	13/13			3.1 MiB	
2	countByValue at Reuters.scala:54	+details	2024/06/08 12:34:18	17 s	13/13	1420.9 MiB			3.1 MiB
1	countByValue at Reuters.scala:40	+details	2024/06/08 12:34:17	0.2 s	2/2			2029.0 B	
0	countByValue at Reuters.scala:40	+details	2024/06/08 12:34:15	3 s	2/2	33.7 MiB			2029.0 B

Page: 1 — 1 Pages. Jump to 1 . Show 100 items in a page. Go

1.2 Φόρτωση δεδομένων

1.2.1 Κατηγορίες

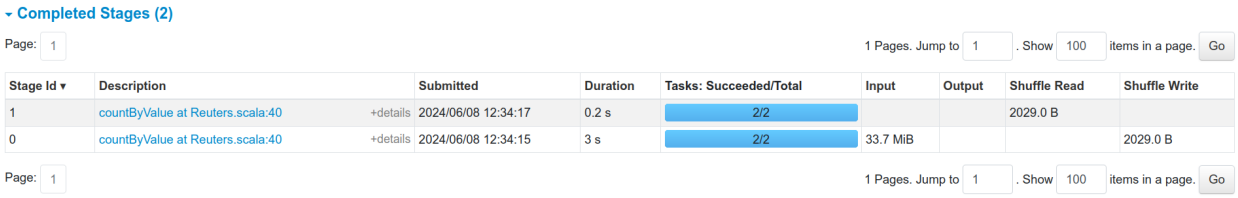
Κάθε γραμμή έχει ένα ζεύγος κατηγορίας - είδησης.

Αφότου γίνει `split`, εφαρμόζεται `map` στην γραμμή και μετατρέπεται σε ένα `tuple` με αντεστραμμένο το ζεύγος (για να γίνει το `join` μετέπειτα):

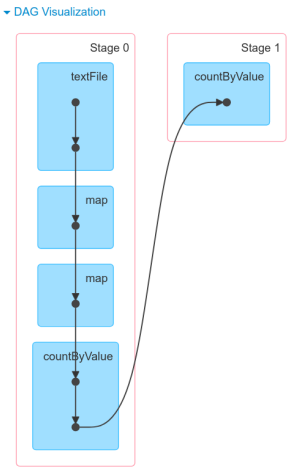
```
val categories = sc.textFile(categoriesPath).map { line =>
  val parts = line.split(" ")
  (parts(1).toInt, parts(0))
}
val catDocs =
    sc.parallelize((categories.map(_._2).countByValue()).toSeq)
```

Επιπλέον ορίστηκε το `catDocs` το οποίο μετατρέπεται σε `RDD` με την `parallelize` και κρατάει πληροφορία για το πόσες ειδήσεις έχει κάθε κατηγορία.

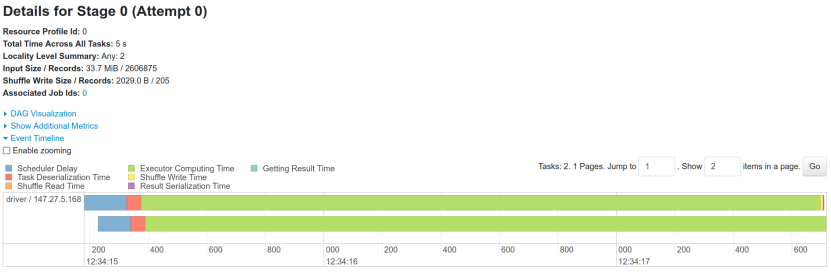
Εδώ δημιουργήθηκαν 2 `stages`, αφού εφαρμόζεται η `.countByValue()`, και είναι `wide operation`:

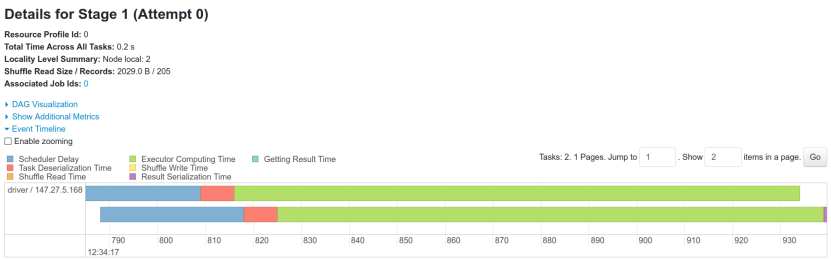


Ακολουθεί το `DAG visualization` για οπτικοποίηση:



Η παραλληλοποίηση φαίνεται και στα παρακάτω `figures`:





1.2.2 Όροι

Κάθε γραμμή έχει μία είδηση και ένα πλήθος από όρους.

Εδώ θα γίνουν δύο αντιστοιχίσεις.

Αρχικά, για κάθε γραμμή γίνεται η αντιστοίχιση είδησης με τους όρους της.

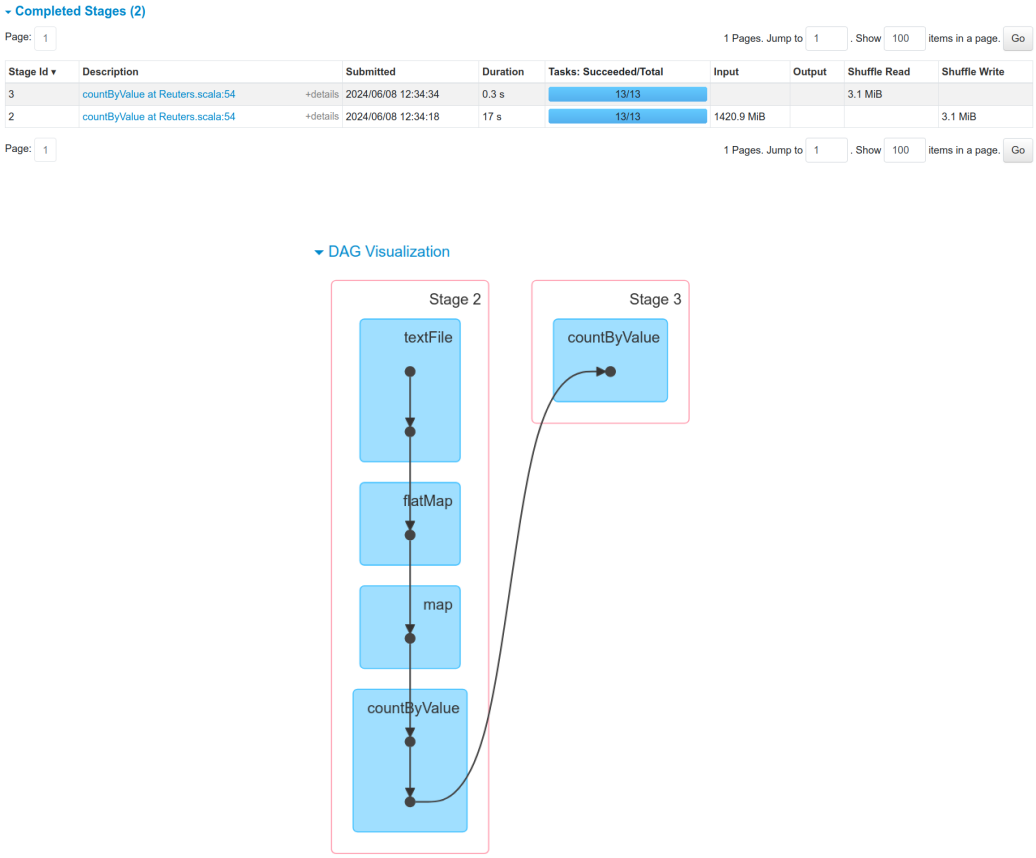
Οπότε υπάρχει μία δομή που περιέχει ζεύγη όρου - είδησης σε `tuples`.

Αυτό γίνεται για κάθε γραμμή οπότε προκύπτει δομή μέσα σε δομή. Για τον λόγο αυτό γίνεται το `flatMap`.

```
// loading the terms in a val
// terms = [(docId1, term11), ..., (docId1, term11),
//         (docId2, term21), ..., (docId2, term22),
//              .           .          .
//              .           .          .
//              .           .          .
//         ( docId ,  term1 ), ..., (docId ,    term   )]
val terms = sc.textFile(termsPath).flatMap { line =>
  val parts = line.split(" ")
  // BE CAREFUL AND DROP 2 NOT 1 !!!
  parts.drop(2).map( term => (parts(0).toInt,
      term.split(":")(0).toInt) )
}
val termDocs =
    sc.parallelize((terms.map(_._2).countByValue()).toSeq)
```

Επιπλέον ορίστηκε το `termDocs` το οποίο μετατρέπεται σε `RDD` με την `parallelize` και κρατάει πληροφορία για το πόσες ειδήσεις έχει κάθε όρος.

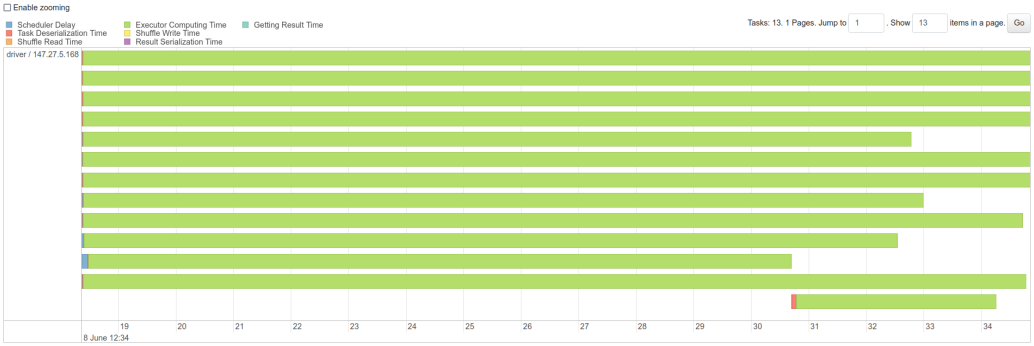
Τα `figures` στη λογική είναι ίδια με παραπάνω, αλλά αξίζει να σημειωθεί το (σχεδόν) εξαπλάσιο πλήθος από `nodes` (βλ. εδώ για εξήγηση):



Details for Stage 2 (Attempt 0)

Resource Profile Id: 0
Total Time Across All Tasks: 3.2 min
Locality Level Summary: Any: 13
Input Size / Records: 1420.9 MiB / 804414
Shuffle Write Size / Records: 3.1 MiB / 440578
Associated Job Ids: 1

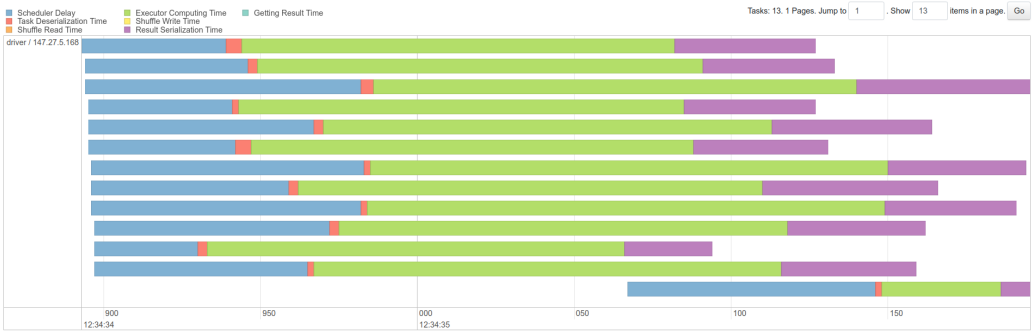
▸ DAG Visualization
▸ Show Additional Metrics
▾ Event Timeline
☐ Enable zooming



Details for Stage 3 (Attempt 0)

Resource Profile Id: 0
Total Time Across All Tasks: 2 s
Locality Level Summary: Node local: 13
Shuffle Read Size / Records: 3.1 MiB / 440578
Associated Job Ids: 1

▸ DAG Visualization
▸ Show Additional Metrics
▾ Event Timeline
☐ Enable zooming



1.2.3 Ρίζες όρων

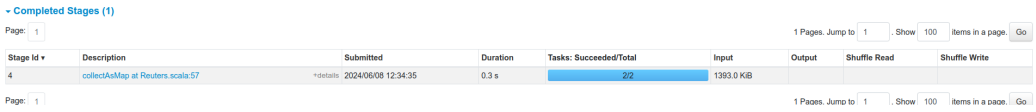
Το μόνο που αλλάζει στη συγκεκριμένη λογική σε σχέση με τα παραπάνω είναι το `.collectAsMap()`.

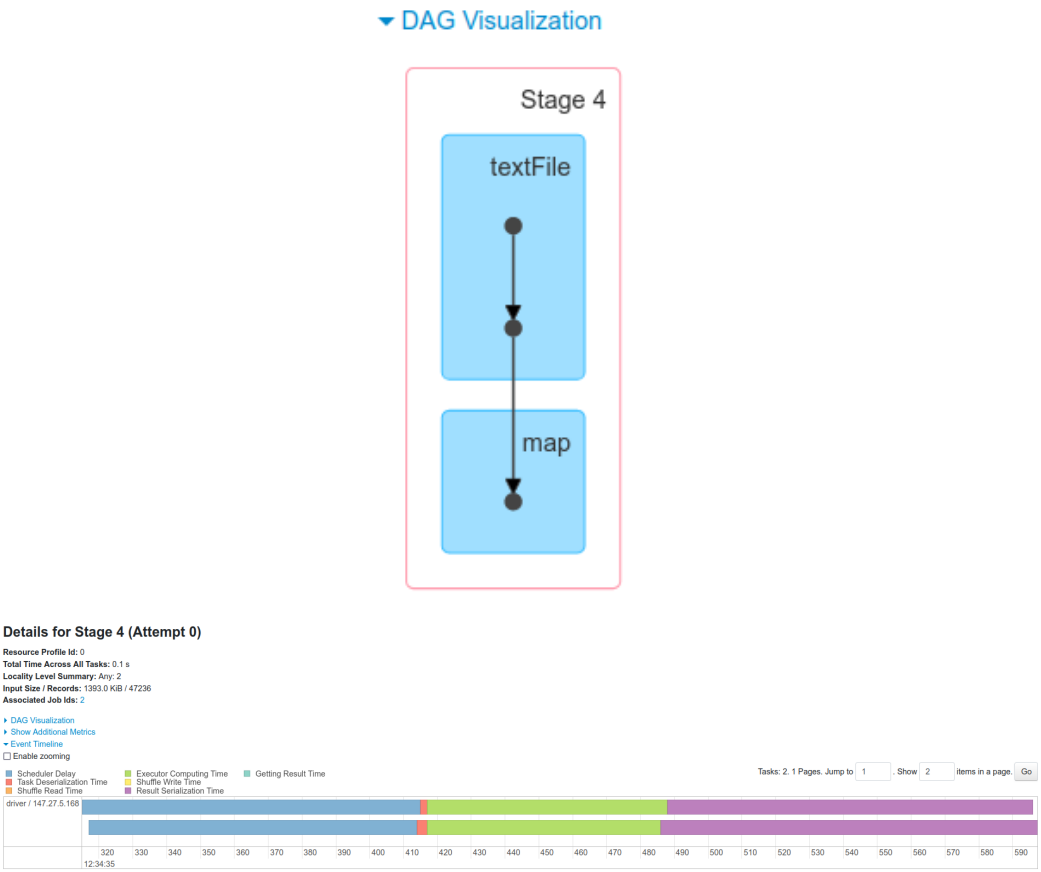
Βάσει του documentation του Spark:

> " Return the key-value pairs in this RDD to the master as a Map. Note: this method should only be used if the resulting data is expected to be small, as all the data is loaded into the driver's memory. "

```
// loading the stems in a val
val stems = sc.textFile(stemsPath).map { line =>
  val parts = line.split(" ")
  (parts(1).toInt, parts(0))
}.collectAsMap() // collected as Map [O(1) access]
val stemsTerms = sc.parallelize(stems.toSeq)
```

Το μοναδικό stage γίνεται παράλληλα:





1.3 Υπολογισμός Jaccard Index

Ο υπολογισμός της μετρικής γίνεται χρησιμοποιώντας τον τύπο της εκφώνησης:

$$J(T, C) = \frac{|\mathrm{DOC}(T) \cap \mathrm{DOC}(C)|}{|\mathrm{DOC}(T) \cup \mathrm{DOC}(C)|}$$

Ισχύει ότι:

$$|\mathrm{DOC}(T) \cup \mathrm{DOC}(C)| = |\mathrm{DOC}(T)| + |\mathrm{DOC}(C)| - |\mathrm{DOC}(T) \cap \mathrm{DOC}(C)|$$

Εδώ γίνεται το `join` μεταξύ κατηγοριών και όρων, βάσει των ειδήσεων.

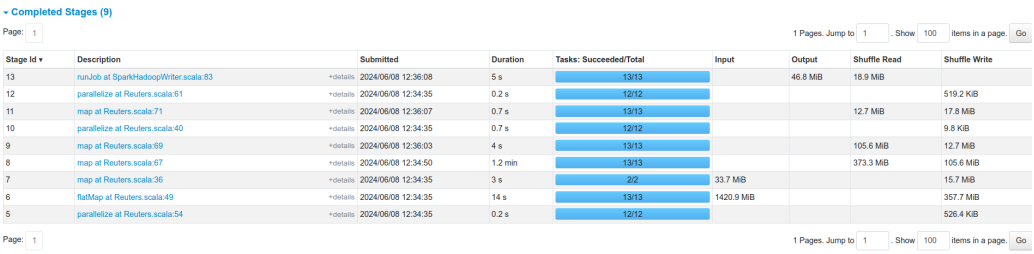
Έπειτα γίνεται `map` και αγνοούνται οι ειδήσεις, ενθυλακώνοντας την πληροφορία για το αν υπάρχει σύνδεση μεταξύ κατηγορίας και όρου - μέσω κοινής είδησης - βάζοντας 1 στο ζεύγος.

Το `.reduceByKey(_ + _)` αθροίζει τις μονάδες (1) για τα κοινά ζεύγη κατηγορίας - είδησης.

Τέλος, γίνονται τα `join` με τα αντίστοιχα πλήθη ειδήσεων για κάθε κατηγορία και όρο αντιστοίχως.

```
// resulting RDD[String] contains jaccard index for every category
// term pair using joins to calculate it
val result = categories
  .join(terms)
  .map { case (_, (c, t)) => ((c, t), 1) }
  .reduceByKey(_ + _)
  .map { case ((c, t), i) => (c, (t, i)) }
  .join(catDocs)
  .map { case (cat, ((term, count), catCount)) => (term, (cat,
     count, catCount)) }
  .join(termDocs)
  .join(stemsTerms)
  .map { case (_, (((cat, count, catCount), termCount), stem)) =>
    s"$cat;$stem;${count.toDouble / (catCount + termCount -
       count).toDouble}"
  }
```

Εννέα `stages`, λόγω της χρήσης τιμών `categories`, `terms` και των `wide operations join, reduceByKey`:



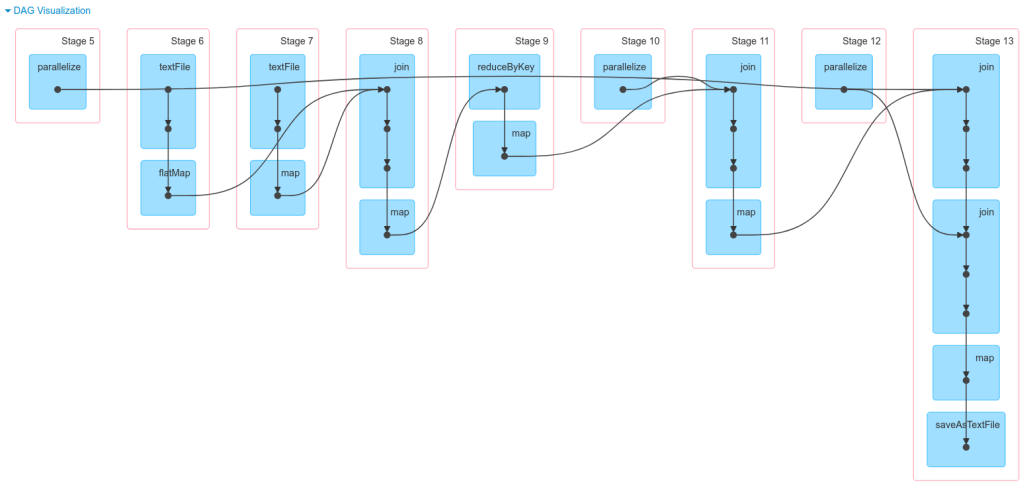
Completed Stages (9)

Page: 1 — 1 Pages. Jump to 1 . Show 100 items in a page. Go

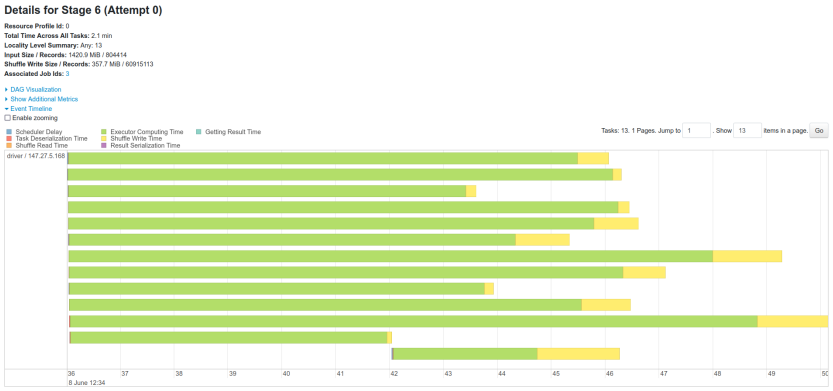
Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
13	runJob at SparkHadoopWriter.scala:83 +details	2024/06/08 12:36:08	5 s	13/13		46.8 MiB	18.9 MiB	
12	parallelize at Reuters.scala:61 +details	2024/06/08 12:34:35	0.2 s	12/12				519.2 KiB
11	map at Reuters.scala:71 +details	2024/06/08 12:36:07	0.7 s	13/13			12.7 MiB	17.8 MiB
10	parallelize at Reuters.scala:40 +details	2024/06/08 12:34:35	0.7 s	12/12				9.8 KiB
9	map at Reuters.scala:69 +details	2024/06/08 12:36:03	4 s	13/13			105.6 MiB	12.7 MiB
8	map at Reuters.scala:67 +details	2024/06/08 12:34:50	1.2 min	13/13			373.3 MiB	105.6 MiB
7	map at Reuters.scala:36 +details	2024/06/08 12:34:35	3 s	2/2	33.7 MiB			15.7 MiB
6	flatMap at Reuters.scala:49 +details	2024/06/08 12:34:35	14 s	13/13	1420.9 MiB			357.7 MiB
5	parallelize at Reuters.scala:54 +details	2024/06/08 12:34:35	0.2 s	12/12				526.4 KiB

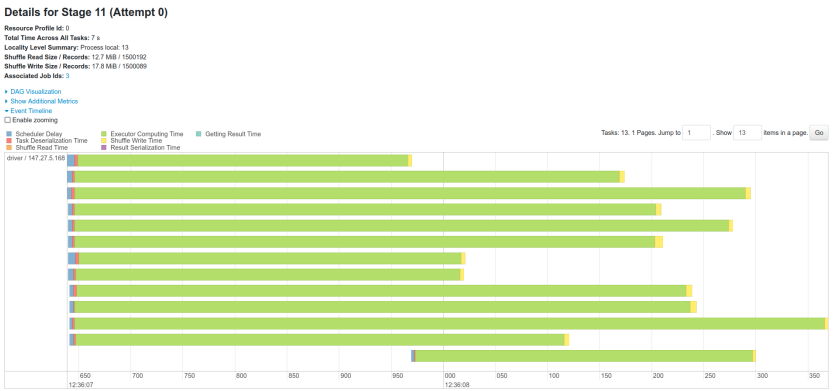
Page: 1 — 1 Pages. Jump to 1 . Show 100 items in a page. Go

DAG visualization:



Και δύο τυχαία stages ενδεικτικά:





1.4 Αποτελέσματα

Ο κώδικας για την αποθήκευση σε αρχείο στο `hdfs`:

```
result.saveAsTextFile(outputPath)
```

Παράδειγμα τυχαίων γραμμών του αρχείου εξόδου:

```
C152;storagetek;1.3679142044210987E-5
C15;storagetek;1.3175577749084297E-5
C181;storagetek;2.3048909786567094E-5
C182;storagetek;2.1353833013025838E-4
C21;storagetek;3.934684241589613E-5
C22;storagetek;3.262642740619902E-4
C17;storagetek;7.114905727499111E-5
GODD;kury;3.2310177705977385E-4
GCAT;kury;0.001178540532182097
C18;kury;1.318168122928593E-4
G151;kury;8.337965536409116E-4
M132;kury;3.6975411351451285E-5
C24;kury;3.082044011588486E-5
GPRO;kury;3.454231433506045E-4
E211;kury;1.8681113394358303E-4
GVIO;kury;6.077734220682529E-5
C32;kury;0.0012631578947368421
G154;kury;4.6008741660915573E-4
G155;kury;4.137360364087712E-4
E12;kury;1.825550403446639E-4
GDEF;kury;5.475851494907458E-4
```

```
C11;kury;1.2187195320116997E-4
G15;kury;2.3853823767950002E-4
CCAT;kury;3.930755805726325E-5
GOBIT;kury;8.795074758135445E-4
C151;kury;1.2167966611099619E-5
C13;kury;2.6523088348407288E-5
C41;kury;8.585164835164836E-5
MCAT;kury;4.875361386162749E-6
E21;kury;6.909099283756707E-5
GDIP;kury;2.3669261519040607E-4
GPOL;kury;2.9743679468113024E-4
ECAT;kury;6.655241834850174E-5
C15;kury;6.575573061192283E-6
```

Σε δοκιμαστικό κώδικα έγινε και `sort` βάσει `Jaccard Index` (ακολουθεί δείγμα των πρώτων 7 αποτελεσμάτων):

```
M11;index;0.495488489951332
G15;eu;0.46051515254352327
C15;net;0.4506494002806152
C151;net;0.443603202453445
G154;emu;0.4394870219628321
CCAT;compan;0.4153842904315215
C172;iss;0.4011808576755749
```

1.5 Cluster

Αρχικά αλλάζει το `build.sbt`

```
...
ThisBuild / scalaVersion := "2.11.8"
...
libraryDependencies ++= Seq(
    "org.apache.spark" %% "spark-core" % "2.3.1",
    "org.apache.spark" %% "spark-sql" % "2.3.1",
    "org.apache.hadoop" % "hadoop-client" % "3.3.1")
```

Και τα `configs`:

```
    // CLUSTER CONFIGS
    val spark = SparkSession.builder
        .appName("Reuters")
        .master("yarn")
        .config("spark.hadoop.fs.defaultFS",
           "hdfs://clu01.softnet.tuc.gr:8020")
        .config("spark.hadoop.yarn.resourcemanager.address",
           "http://clu01.softnet.tuc.gr:8189")
        .config("spark.hadoop.yarn.application.classpath",
           "$HADOOP_CONF_DIR,$HADOOP_COMMON_HOME/*,
        $HADOOP_COMMON_HOME/lib/*,$HADOOP_HDFS_HOME/*,
        $HADOOP_HDFS_HOME/lib/*,$HADOOP_MAPRED_HOME/*,
        $HADOOP_MAPRED_HOME/lib/*,$HADOOP_YARN_HOME/*,
        $HADOOP_YARN_HOME/lib/*")
        .getOrCreate()
    val sc = spark.sparkContext
    val hdfsURI = "hdfs://clu01.softnet.tuc.gr:8020"
    FileSystem.setDefaultUri(sc.hadoopConfiguration, hdfsURI)
    val hdfs = FileSystem.get(spark.sparkContext.hadoopConfiguration)
    val categoriesPath = "/user/chrisa/Reuters/rcv1-v2.topics.qrels"
    val termsPath = "/user/chrisa/Reuters/lyrl2004_vectors_*.dat" //
        '*' means match anything
    val stemsPath = "/user/chrisa/Reuters/stem.termid.idf.map.txt"
    val outputPath = "/user/fp19/output"
```

Σενάριο εφαρμογής 2: Ανάλυση τροχιάς πλοίων στο Αιγαίο Πέλαγος με χρήση Spark DFs (2 μονάδες)

2.1 Configurations

Ακολουθούν οι ρυθμίσεις για να τρέχει `locally` η εφαρμογή.

```
// LOCAL CONFIGS
val spark = SparkSession.builder()
  .master("local[*]") // local[*] means "use as many threads as
     the number of processors available to the Java virtual
     machine"
  .appName("Aegean")
  .getOrCreate()
val dataPath = "hdfs://localhost:9000/Aegean/nmea_aegean.logs"
val out1 = "hdfs://localhost:9000/Aegean/output2/out1"
val out2 = "hdfs://localhost:9000/Aegean/output2/out2"
val out3 = "hdfs://localhost:9000/Aegean/output2/out3"
val out4 = "hdfs://localhost:9000/Aegean/output2/out4"
val out5 = "hdfs://localhost:9000/Aegean/output2/out5"
```

Και εδώ φαίνεται η εφαρμογή στο `History Server`:

3.5.1	local-1718917362980	Aegean	2024-06-21 00:02:42	2024-06-21 00:03:09	27 s	ckapelonis	2024-06-21 00:03:09

Το `Spark` χώρισε την εφαρμογή σε 17 `Jobs` (πώς αποφασίστηκε ο αριθμός 17;).

Σε αυτό το σενάριο τα `actions` ήταν `csv` (δύο φορές λόγω του `.option("inferSchema", "true")`), `rdd`, `collect`, `count`:

Completed Jobs (18)

Page: 1

Job Id	Description
17	runJob at SparkHadoopWriter.scala:83 runJob at SparkHadoopWriter.scala:83
16	collect at Aegean.scala:93 collect at Aegean.scala:93
15	rdd at Aegean.scala:91 rdd at Aegean.scala:91
14	runJob at SparkHadoopWriter.scala:83 runJob at SparkHadoopWriter.scala:83
13	collect at Aegean.scala:83 collect at Aegean.scala:83
12	rdd at Aegean.scala:81 rdd at Aegean.scala:81
11	runJob at SparkHadoopWriter.scala:83 runJob at SparkHadoopWriter.scala:83
10	collect at Aegean.scala:75 collect at Aegean.scala:75
9	rdd at Aegean.scala:73 rdd at Aegean.scala:73
8	rdd at Aegean.scala:73 rdd at Aegean.scala:73
7	runJob at SparkHadoopWriter.scala:83 runJob at SparkHadoopWriter.scala:83
6	collect at Aegean.scala:61 collect at Aegean.scala:61
5	rdd at Aegean.scala:59 rdd at Aegean.scala:59
4	runJob at SparkHadoopWriter.scala:83 runJob at SparkHadoopWriter.scala:83
3	collect at Aegean.scala:51 collect at Aegean.scala:51
2	rdd at Aegean.scala:48 rdd at Aegean.scala:48
1	csv at Aegean.scala:41 csv at Aegean.scala:41
0	csv at Aegean.scala:41 csv at Aegean.scala:41

Completed Stages (18)

Page: 1

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total
30	runJob at SparkHadoopWriter.scala:83 +details	2024/06/21 00:03:08	0.5 s	1/1
28	collect at Aegean.scala:93 +details	2024/06/21 00:03:08	13 ms	1/1
26	rdd at Aegean.scala:91 +details	2024/06/21 00:03:07	1 s	12/12
25	runJob at SparkHadoopWriter.scala:83 +details	2024/06/21 00:03:06	0.5 s	1/1
23	collect at Aegean.scala:83 +details	2024/06/21 00:03:06	34 ms	1/1
21	rdd at Aegean.scala:81 +details	2024/06/21 00:03:04	2 s	12/12
20	runJob at SparkHadoopWriter.scala:83 +details	2024/06/21 00:03:04	49 ms	1/1
17	collect at Aegean.scala:75 +details	2024/06/21 00:03:04	29 ms	1/1
14	rdd at Aegean.scala:73 +details	2024/06/21 00:03:04	85 ms	1/1
12	rdd at Aegean.scala:73 +details	2024/06/21 00:03:00	4 s	12/12
11	runJob at SparkHadoopWriter.scala:83 +details	2024/06/21 00:02:59	0.6 s	1/1
9	collect at Aegean.scala:61 +details	2024/06/21 00:02:59	0.2 s	1/1
7	rdd at Aegean.scala:59 +details	2024/06/21 00:02:57	2 s	12/12
6	runJob at SparkHadoopWriter.scala:83 +details	2024/06/21 00:02:57	0.2 s	1/1
4	collect at Aegean.scala:51 +details	2024/06/21 00:02:56	0.1 s	1/1
2	rdd at Aegean.scala:48 +details	2024/06/21 00:02:52	4 s	12/12
1	csv at Aegean.scala:41 +details	2024/06/21 00:02:47	5 s	12/12
0	csv at Aegean.scala:41 +details	2024/06/21 00:02:46	0.2 s	1/1

Κάποια `stages` έγιναν `skip` (βλ. εδώ):

Skipped Stages (13)

Page: 1 — 1 Pages. Jump to 1 . Show 100 items in a page. Go

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
29	rdd at Aegean.scala:91 +details	Unknown	Unknown	0/12				
27	rdd at Aegean.scala:91 +details	Unknown	Unknown	0/12				
24	rdd at Aegean.scala:81 +details	Unknown	Unknown	0/12				
22	rdd at Aegean.scala:81 +details	Unknown	Unknown	0/12				
19	rdd at Aegean.scala:73 +details	Unknown	Unknown	0/1				
18	rdd at Aegean.scala:73 +details	Unknown	Unknown	0/12				
16	rdd at Aegean.scala:73 +details	Unknown	Unknown	0/1				
15	rdd at Aegean.scala:73 +details	Unknown	Unknown	0/12				
13	rdd at Aegean.scala:73 +details	Unknown	Unknown	0/12				
10	rdd at Aegean.scala:59 +details	Unknown	Unknown	0/12				
8	rdd at Aegean.scala:59 +details	Unknown	Unknown	0/12				
5	rdd at Aegean.scala:48 +details	Unknown	Unknown	0/12				
3	rdd at Aegean.scala:48 +details	Unknown	Unknown	0/12				

Η λογική είναι παρόμοια με το πρώτο σενάριο οπότε θα αναλυθούν κάποια ενδεικτικά `stages`.

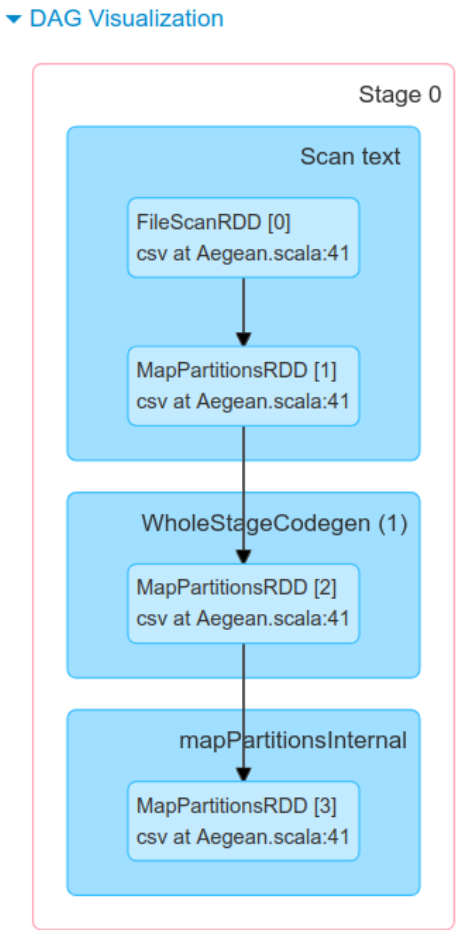
Εδώ, το πρώτο `stage` είναι η φόρτωση δεδομένων `csv` μορφής στη γραμμή 41:

Completed Stages (1)

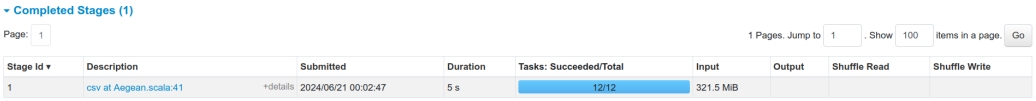
Page: 1 — 1 Pages. Jump to 1 . Show 100 items in a page. Go

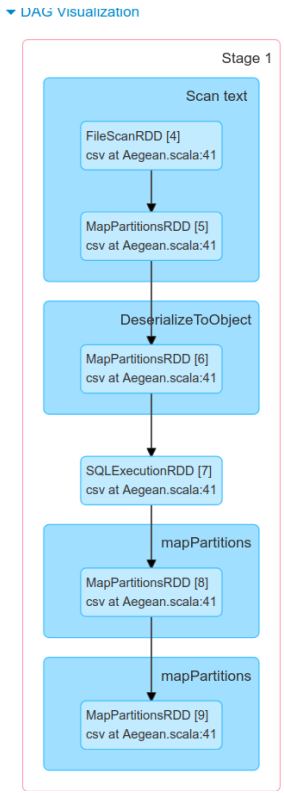
Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
0	csv at Aegean.scala:41 +details	2024/06/21 00:02:46	0.2 s	1/1	64.0 KiB			

Η ανάλυση DAG:

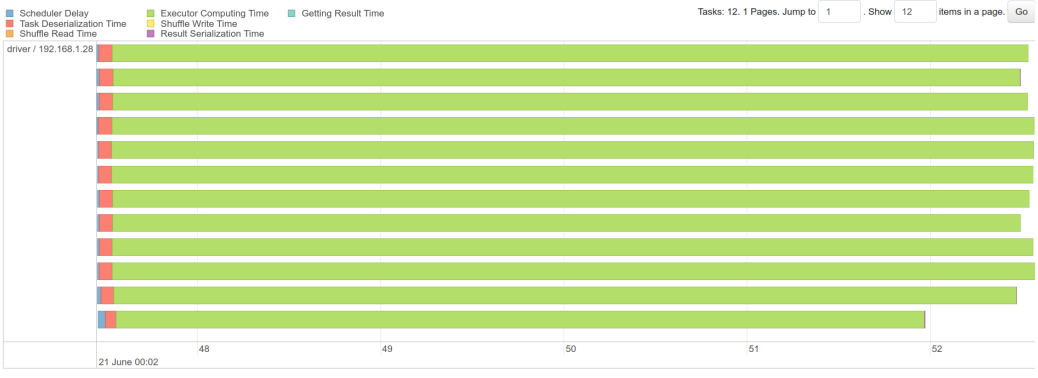


Ομοίως, με το δεύτερο:

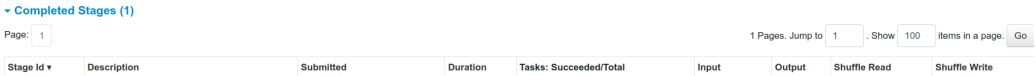


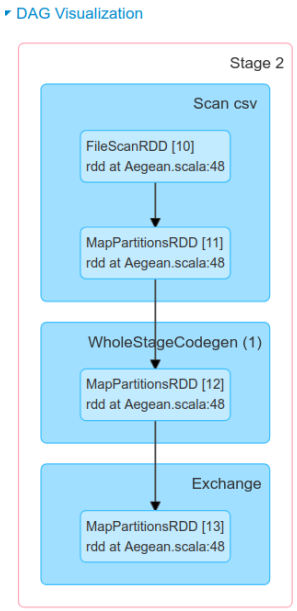


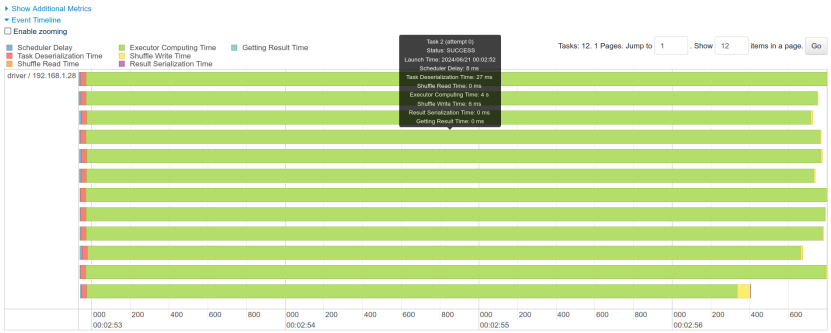
Κι εδώ φαίνεται η παραλληλοποίηση:



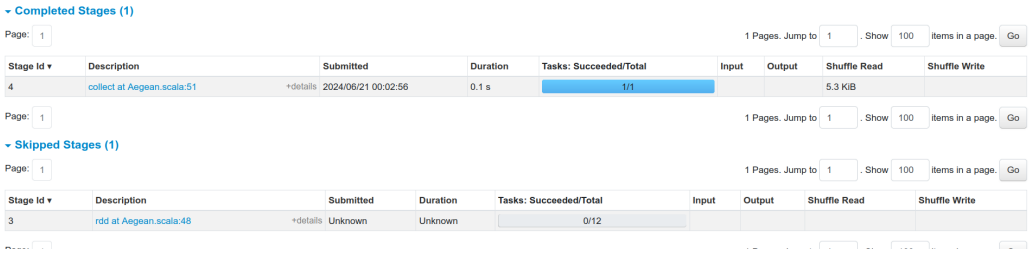
Για το τρίτο stage:

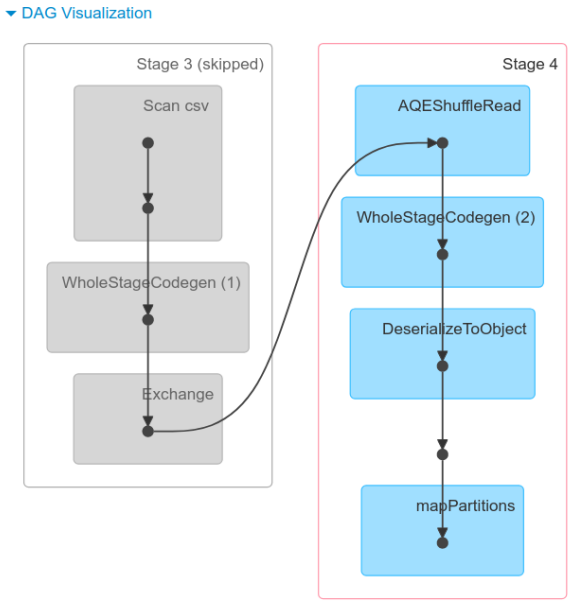




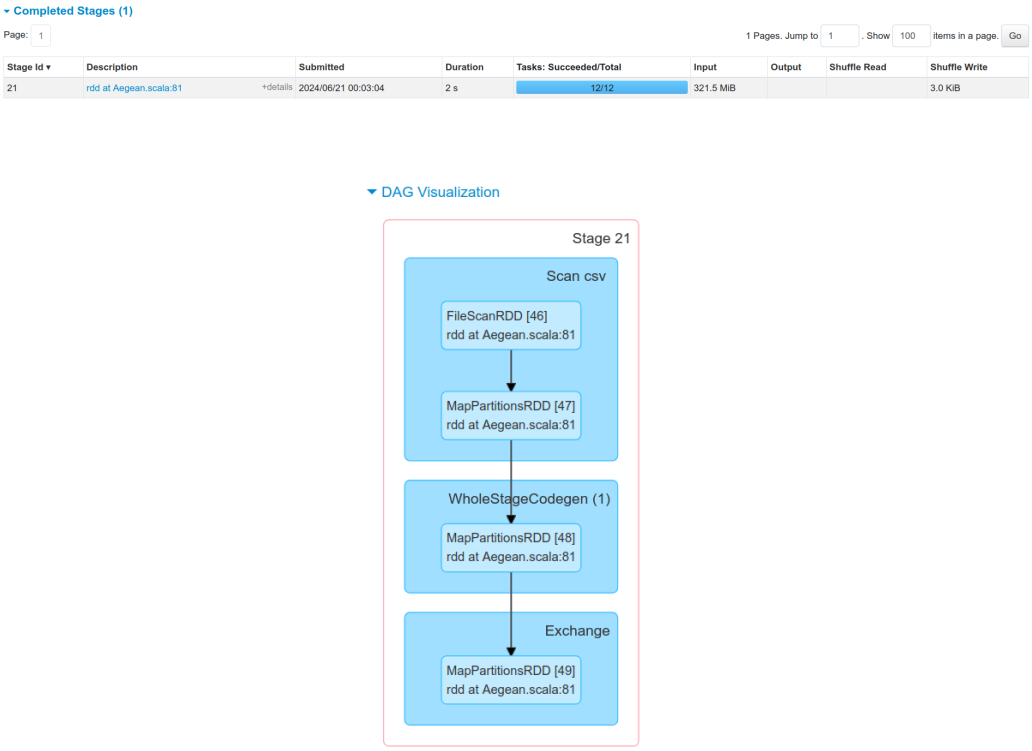


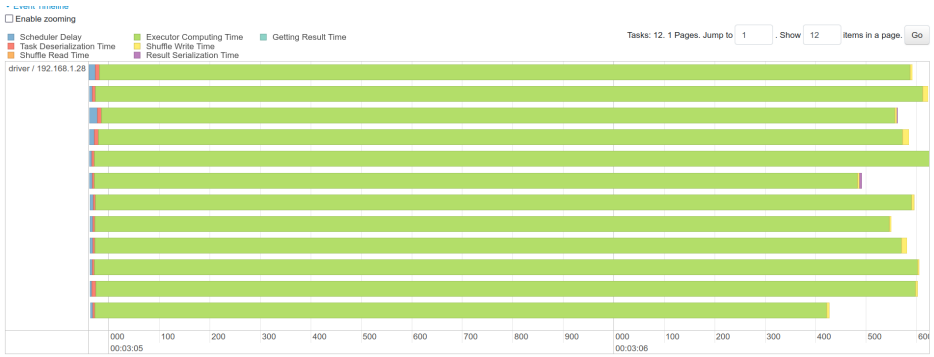
Κι εδώ φαίνεται κι ένα παράδειγμα σε κάποιο `job` που έγινε παράλειψη ενός `stage`, ως περιττό:





Τέλος, ένα από τα τελευταία `stages`:





2.2 Φόρτωση δεδομένων

Εδώ γίνεται η φόρτωση των δεδομένων στο `DataFrame` με όνομα `dataset` και τιμή τα `logs` με τις απαραίτητες μόνο στήλες.

Υπολογισμοί που θα χρειάζονταν για μετά έγιναν αμέσως εδώ για να μην κρατιέται επιπλέον, άχρηστη πληροφορία.

Η επιλογή `inferSchema` βοηθάει στο να συμπεραίνεται αυτόματα το schema κάθε στήλης:

```
// Reading the Aegean logs from hdfs
val dataset = spark.read
  .option("header", "true") // first row is column names
  .option("inferSchema", "true") // automatically infer the schema
     of each column
  .csv(dataPath)
  .withColumn("date", to_date(col("timestamp")))
  .withColumn("abs_diff", abs(col("heading") -
     col("courseoverground")))
  .select("station", "date", "mmsi", "abs_diff",
     "speedoverground", "status") // select only necessary columns
```

2.3 Ερωτήματα

2.3.1 Q1

Εδώ χρειάστηκε το `timestamp` χωρίς την ακριβή ώρα, άρα χρησιμοποιήθηκε το `date`, όπως ορίστηκε κατά τη φόρτωση των `logs`.

Έγινε η ομαδοποίηση βάσει τον σταθμό και την μέρα κι έπειτα έγινε η καταμέτρηση:

```
// Question 1: grouping by station and date since we want per
    station per day
val q1 = dataset
  .groupBy("station", "date")
  .count()
  .rdd
q1.collect().take(30).foreach(println)
q1.saveAsTextFile(out1)
```

Το αποτέλεσμα (μόνο 30 πρώτες γραμμές):

```
[10001,2024-04-17,14078]
[10001,2024-04-18,4531]
[10003,2024-04-19,3726]
[10004,2024-04-18,4861]
[10004,2024-04-19,378]
[8006,2024-04-16,62625]
[10002,2024-04-18,213]
[10002,2024-04-20,743]
[10003,2024-04-17,42794]
[10002,2024-04-19,1025]
[10002,2024-04-17,430]
[10003,2024-04-20,4606]
[10003,2024-04-18,2691]
[10002,2024-04-16,257]
[10004,2024-04-20,504772]
[10004,2024-04-17,83189]
[8006,2024-04-17,34207]
[10003,2024-04-21,2948]
[10004,2024-04-21,160945]
[10003,2024-04-22,5584]
[10004,2024-04-22,496801]
[10001,2024-04-22,36278]
[10004,2024-04-23,157595]
[10001,2024-04-24,23425]
[10002,2024-04-23,606]
[10004,2024-04-24,108955]
[10002,2024-04-24,832]
[10003,2024-04-23,18435]
[10003,2024-04-24,763]
[10001,2024-04-23,25280]
```

2.3.2 Q2

Εδώ η ομαδοποίηση έγινε βάσει το `vessel_id` και επιλέχθηκε το μεγαλύτερο `count`:

```
// Question 2: grouping and counting based on #mmsi, then taking
    the max
val q2 = dataset
  .groupBy("mmsi")
  .count()
  .orderBy(desc("count"))
  .limit(1)
  .rdd
q2.collect().foreach(println)
q2.saveAsTextFile(out2)
```

Το αποτέλεσμα:

```
[256002039,125459]
```

2.3.3 Q3

Το πιο περίπλοκο ερώτημα.

Εδώ, αφότου έγινε το φιλτράρισμα και πάρθηκαν μόνο οι ζητούμενοι σταθμοί, έγινε η ομαδοποίηση βάσει `vessel_id, date` σε συνάρτηση με την `agg` συνάρτηση που μάζεψε σε ένα σύνολο τους σταθμούς και τη μέση τιμή ταχύτητας σε σχέση με το έδαφος.

Σε αυτό όμως το σημείο πρέπει να μείνουν μόνο όσα πλοία βρίσκονταν την ίδια μέρα **και** στους δύο σταθμούς, πράγμα που εφαρμόζεται στο `.filter(size(col("stations")) === 2)`:

```
// Question 3:
val q3 = dataset
  .filter(col("station") === 8006 || col("station") === 10003) //
     get rid of the rest of the stations
  .groupBy("mmsi", "date") // grouping by mmsi and date to find...
  .agg(
    collect_set("station").alias("stations"),
    avg("speedoverground").alias("avg_SOG")
  ) // ...the stations foreach vessel on the same date and average
     their SOG's
  .filter(size(col("stations")) === 2) // take only those both at
     8006 and 10003
  .select(avg("avg_SOG").alias("average_SOG")) // average the avg's
  .rdd
q3.collect().take(30).foreach(println)
q3.saveAsTextFile(out3)
```

Το αποτέλεσμα:

```
[0.30748089310156396]
```

2.3.4 Q4

Εδώ αφότου έγινε η ομαδοποίηση κατά σταθμό, υπολογίστηκε η μέση τιμή της διαφοράς τροχιών (η διαφορά έχει ήδη τοποθετηθεί σε στήλη κατά τη φόρτωση δεδομένων):

```
// Question 4: per station -> group by station
val q4 = dataset
  .groupBy("station")
  .agg(avg("abs_diff").alias("avg_diff")) // difference already
     calculated
  .rdd
q4.collect().take(30).foreach(println)
q4.saveAsTextFile(out4)
```

Το αποτέλεσμα ανά σταθμό:

```
[8006,96.13389891771453]
[10002,98.36212078595254]
[10001,104.57120025147312]
[10003,113.64403149039288]
[10004,97.39718926398415]
```

2.3.5 Q5

Εδώ η λογική είναι ίδια με το ερώτημα 2:

```
// Question 5: grouping by status and counting occurences of
    status_i, then sorting and taking top 3
val q5 = dataset
  .groupBy("status")
  .count()
  .orderBy(desc("count"))
  .limit(3)
  .rdd
q5.collect().foreach(println)
q5.saveAsTextFile(out5)
```

Οι τρεις συχνότερες καταστάσεις και ο αριθμός εμφάνισής τους:

```
[0,2486924] // underway using engine
[15,568507] // undefined
[5,412089] // moored
```

2.4 Cluster

Αρχικά αλλάζει το `build.sbt`

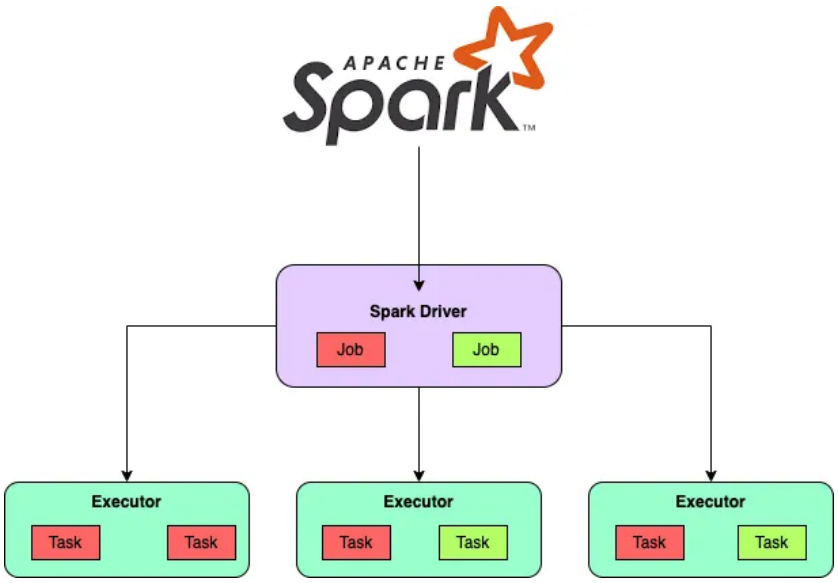
```
...
ThisBuild / scalaVersion := "2.11.8"
...
libraryDependencies ++= Seq(
    "org.apache.spark" %% "spark-core" % "2.3.1",
    "org.apache.spark" %% "spark-sql" % "2.3.1",
    "org.apache.hadoop" % "hadoop-client" % "3.3.1")
```

Και τα `configs`:

```
  // CLUSTER CONFIGS
  val spark = SparkSession.builder
    .appName("Aegean")
    .master("yarn")
    .config("spark.hadoop.fs.defaultFS",
       "hdfs://clu01.softnet.tuc.gr:8020")
    .config("spark.hadoop.yarn.resourcemanager.address",
       "http://clu01.softnet.tuc.gr:8189")
    .config("spark.hadoop.yarn.application.classpath",
    "$HADOOP_CONF_DIR,$HADOOP_COMMON_HOME/*,
    $HADOOP_COMMON_HOME/lib/*,$HADOOP_HDFS_HOME/*,
    $HADOOP_HDFS_HOME/lib/*,$HADOOP_MAPRED_HOME/*,
    $HADOOP_MAPRED_HOME/lib/*,$HADOOP_YARN_HOME/*,
    $HADOOP_YARN_HOME/lib/*")
    .getOrCreate()
  val sc = spark.sparkContext
  val hdfsURI = "hdfs://clu01.softnet.tuc.gr:8020"
  FileSystem.setDefaultUri(sc.hadoopConfiguration, hdfsURI)
  val dataPath = "/user/chrisa/nmea_aegean/nmea_aegean.logs"
  val out1 = "/user/fp19/output2/out1"
  val out2 = "/user/fp19/output2/out2"
  val out3 = "/user/fp19/output2/out3"
  val out4 = "/user/fp19/output2/out4"
  val out5 = "/user/fp19/output2/out5"
```

Πώς εκτελείται ο κώδικας στο Spark;

Μια εικόνα χίλιες λέξεις:

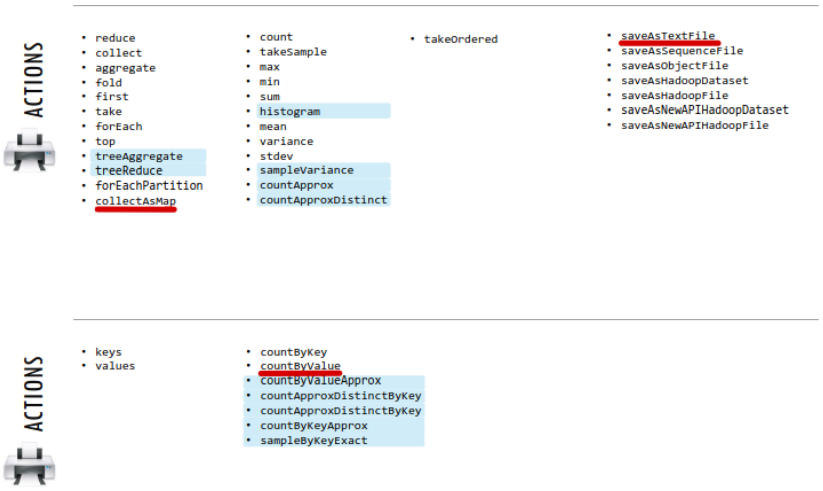


3.1 Jobs

Όπως αναφέρεται και στις διαλέξεις, όταν υποβάλλεται μία εφαρμογή στο `Spark`, τη χωρίζει σε έναν αριθμό από `jobs` ανάλογα με τον αριθμό των `actions`.

- **Job**
  - **The number of jobs that will be submitted depend on the number of "actions" (to be presented later on) included in our Spark Program**

Για παράδειγμα, στο πρώτο σενάριο τα `actions` είναι 4, άρα 4 και τα `jobs`:

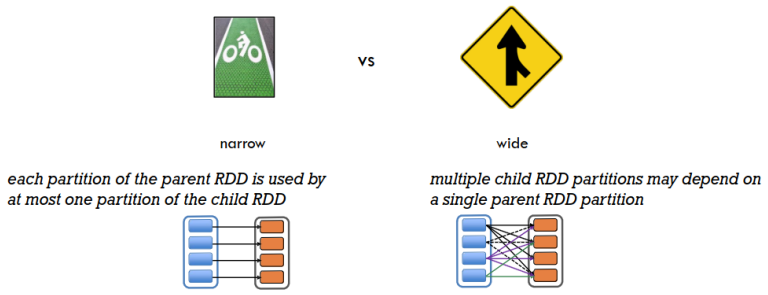


3.2 Stages

Κάθε `job` χωρίζεται με τη σειρά του σε `stages`, συνήθως από `narrow` ή `shuffle operations`.

- **Stage**
  - **Set of transf ormations that can be pipelined and executed by a single independent worker. Usually it is app the transformations between "read", "shuffle", "action"" "save"**

Εδώ σημειώνεται και η διαφορά μεταξύ `narrow` και `wide operation`:



Κατά κανόνα, όποτε συναντάται `wide operation` υπάρχει και νέο `stage`.

3.3 Tasks

Τέλος, θα γίνει αναφορά στα `tasks`, την πιο μικρή μονάδα στο `Spark` που εκτελεί έργο.

- **Task**
  - **Execution of the stage on a single data partition. Basic unit of scheduling**

Το σημαντικό είναι ότι αντιπροσωπεύει ένα `operation` που μπορεί να εκτελεστεί πάνω σε ένα υποσύνολο δεδομένων.

Κάθε `task` εκτελείται από `worker nodes` **παράλληλα**. Τα `tasks` έχουν άμεση σύνδεση με τον αριθμό των `partitions` στα οποία χωρίζεται ένα `RDD`.

Άρα περισσότερα δεδομένα σημαίνουν περισσότερα `partitions` άρα και περισσότερα `tasks`, χωρίς αυτό να ισχύει αυστηρά και χωρίς εξαιρέσεις.

Ένα παράδειγμα στην συγκεκριμένη εφαρμογή είναι οι κατηγορίες σε σύγκριση με τους όρους.

Τα αρχεία με τους όρους είναι πολύ περισσότερα και μεγαλύτερα.

Το παρακάτω cheat sheet συγγράφηκε κατά την περάτωση της άσκησης:

```
user = fp19
password = *********

local!!!!!!!!!!!!!!!!!!!!!!!!!!!!!:
*connect to vpn*
ssh fp19@clu04.softnet.tuc.gr
scp
    /home/ckapelonis/IdeaProjects/Reuters/out/artifacts/Reuters_jar/Reuters.jar
    fp19@clu04.softnet.tuc.gr:/home/fp19/
scp
    /home/ckapelonis/IdeaProjects/Aegean/out/artifacts/Aegean_jar/Aegean.jar
    fp19@clu04.softnet.tuc.gr:/home/fp19/

ssh:
rm ~/Reuters.jar
rm -r ~/output
hdfs dfs -rm -r /user/fp19/output

rm ~/Aegean.jar
rm -r ~/output2
hdfs dfs -rm -r /user/fp19/output2

//run the app
cd /usr/hdp/current/spark2-client
./bin/spark-submit --class Reuters --master yarn --deploy-mode
    cluster ~/Reuters.jar

./bin/spark-submit --class Aegean --master yarn --deploy-mode
    cluster ~/Aegean.jar

hdfs dfs -copyToLocal /user/fp19/output ~
hdfs dfs -copyToLocal /user/fp19/output2 ~
```

Πηγές

https://stackoverflow.com/questions/24029873/
how-to-read-multiple-text-files-into-a-single-rdd

https://spark.apache.org/docs/latest/api/scala/org/apache/
spark/rdd/PairRDDFunctions.html#collectAsMap():
scala.collection.Map[K,V]

https://spark.apache.org/docs/1.5.1/api/java/org/apache/spark/
SparkContext.html#broadcast(T,%20scala.reflect.ClassTag)

https://pratikbarjatya.medium.com/
demystifying-spark-jobs-stages-and-tasks-a-simplified-guide-f35da5ab4aa6

https://stackoverflow.com/questions/34580662/
what-does-stage-skipped-mean-in-apache-spark-web-ui

https://stackoverflow.com/questions/56927329/
spark-option-inferschema-vs-header-true

https://stackoverflow.com/questions/67339570/
is-spark-read-csv-an-action-or-transformation

https://spark.apache.org/docs/latest/sql-data-sources-csv.html

https://stackoverflow.com/questions/44708629/
is-dataset-rdd-an-action-or-transformation

https://stackoverflow.com/questions/44708629/
is-dataset-rdd-an-action-or-transformation