Relazione BigData

Carlo La Viola 461959, Davinder Kumar 460825, gruppo: Kucha

May 19, 2017

Introduzione

Tutti i Job compreso l'esercizio facoltativo sono stati eseguiti in locale e su AWS. La macchina locale ha le seguenti specifiche:
RAM di 8 GByte
Processore i5 da 2,8GHz.
Per quanto riguarda l'esecuzione remota si è fatto utilizzo della configurazione di default di AWS utilizzando quindi 1 master e 2 core nodes. Per valutare al meglio le potenzialità di AWS, le esecuzioni sono state fatte con input di grandezza superiore rispetto ai dataset forniti.
I nuovi file sono stati creati variando il timestamp. Caricati su AWS i 4 dataset; i nuovi file vengono generati a partire da questi tramite l'esecuzione di un software appositamente sviluppato.
Quindi i nuovi file di dimensione maggiore a quelli forniti non sono caricati ma creati su AWS e la grandezza massima del file generato è stata di 23GB. A parità di input si sono valutate le prestazioni su AWS rispetto al locale. Si è giunti alla conclusione che Spark e Hive su tutti i Job scalano meglio su AWS.
Per quanto riguarda Hadoop le prestazioni sono migliori su AWS per il Job1 mentre sono migliori in locale per il Job2 e il Job3. Per Hadoop su Job2 riteniamo che le pretazioni miglioreranno su AWS se l'input viene ulteriormente aumentato di dimensione. Invece per Hadoop su Job3 non abbiamo motivazioni su tale comportamente (si è fatto uso di 2 map e 2 reduce) in quanto ci aspettavamo una situazione opposta.

Note

1. I tempi di esecuzione per Hadoop e Spark sono stati presi tramite un "timer" che è stato fatto partire all'inzio del main e stoppato alla sua fine; quelli di Hive sono stati invece ricavati dal file di log, sommando gli output delle query.

2. I primi 4 punti di tutti i grafici riportati in seguito riguardano i 4 dataset forniti mentre il quinto punto del grafico è relativo al file ottenuto dalla concatenazione di questi dataset.

3. Le prime righe dei file di output presenti in questa relazione sono relative ad esecuzioni fatte solamente sul primo file (1999-2006.csv), mentre i file di output allegati sono relativi alle esecuzioni fatte su tutti i 4 dataset forniti.

4. Nel file 2011-2012.csv ci sono degli UserId che iniziavano con caratteri *#-oc*. Abbiamo interpretato come normali questi tipi di UserID e quindi non è stata fatta alcuna trasformazione del campo.

1 Job 1: Top 5 Products

1.1 Hadoop

Pseudo

```
Map(line){
   tokens = read and split line by "\t";
   timestamp = tokens[7]
   yearMonth = convert(timestamp)
   key = yearMonth + tokens[1] //tokens[1] is prodID
   value = tokens[6]
   return (key, value); //key is yearMonth and prodID and value is the score
}

Shuffle and Sort

//In reduce we have defined a map structure where YearMonth is the key, innerMap is the
    value
map(key = YearMonthProdID, value=innerMap);
//innerMap is an auxiliary map created for each key in map and it stores all products for
    an avg
innerMap(key=average, value=list(prodID));

Reduce(key, value){
   do average for each YearMonth and ProdId
   (yearMonth, prodID) = split(key); //after split we obtain yearMonth and prodID
   if (!map.contains(yearMonth))
      then add element in map; //element is yearMonth, innerMap
   else if (!map.get(yearMonth).contains(average))
      then add element to innerMap; //element is avg, prodID
   else
      add only prodId to innerMap.value; //append product to list of products
}

//in cleanUp we iterate on map and print top 5 products
CleanUp(context){
   print top 5 products;
}
```

Abbiamo notato nella codifica Java, che a causa delle strutture dati utilizzate, in locale, il programma sollevava errore di *OutOfMemory*. Abbiamo successivamente capito che il problema era dovuto al salvataggio di troppi oggetti di tipo String (ProdID) in un *TreeSet*. Per ovviare al problema e quindi per limitare l'occupazione di memoria si è scelto di utilizzare un *ArrayList* e le stringhe sono state salvate in esso come *Byte[]*. Facendo questa scelta è stato possibile aumentare la dimensione di input da processare.

1.2 Hive

Code

```
CREATE TABLE IF NOT EXISTS reviews (
   id STRING,
   productId STRING,
   userId STRING,
   profileName STRING,
   hNum STRING,
   hDen STRING,
   score STRING,
   time STRING,
   summary STRING,
   text STRING)
   ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t';

LOAD DATA LOCAL INPATH '${INPUT_DIR}' INTO TABLE reviews;

add jar ${CUSTOM_JAR_PATH};
CREATE TEMPORARY FUNCTION unix_date AS 'it.uniroma3.kucha.Unix2Date';
CREATE TEMPORARY FUNCTION ranking AS 'it.uniroma3.kucha.Rank';
CREATE TABLE resultEs1 AS
   SELECT ps.tm, ps.productId, ps.avg FROM (
      SELECT unix_date(time) AS tm, productId, ROUND(AVG(CAST(score AS FLOAT)),3) AS avg
          FROM reviews GROUP BY unix_date(time), productId
   )ps
   ORDER BY ps.tm, ps.avg DESC;

SELECT productId,avg from resultEs1
WHERE ranking(tm,avg) < 5;
```

Sono state definite due funzioni in Java per convertire il timestamp in data e per filtrare i top5 o i top10.

1.3 Spark

Pseudo

```
   tokens = read and split line by "\t";
   timestamp = tokens[7];
   yearMonth = convert(timestamp);
   create JavaPairRDD of ((yearMonth prodID), score);
   create JavaPairRDD of ((yearMonth, prodID), avg); //for each (yearMonth, prodID) we have
       avg as sum(score)/number of score
   create JavaPairRDD of ((yearMonth, avg), prodID)
   group and sort by key;
   create JavaPairRDD of (yearMonth, AvgProdIDs)
   group by key;
   for each yearMonth print top 5 element
      yearMonth, avg , list of products
```

1.4 Output

Di seguito vengono riportate le prime righe di output sul file 1999-2006.csv:

```
Hadoop
199910 0006641040 5.0
199912 B00004CI84 5.0
199912 B00004CXX9 5.0
199912 B00004RYGX 5.0
200001 B00002N8SM 5.0
200001 B00004CXX9 3.666
200001 B00004CI84 3.0
200001 B00004RYGX 3.0
200002 B00004CI84 4.0
200002 B00004CXX9 4.0
200002 B00004RYGX 4.0
200006 B00002Z754 5.0
200006 B00004CI84 5.0
200006 B00004CXX9 5.0
200006 B00004RYGX 5.0
200007 B00004RAMX 5.0
200008 B00004CI84 5.0
200008 B00004CXX9 5.0
200008 B00004RYGX 5.0
200008 B00004S1C5 5.0
200008 B00004S1C6 5.0
200010 B00004CI84 5.0
200010 B00004CXX9 5.0
200010 B00004RYGX 5.0

Hive
199910  0006641040 5.0
199912  B00004CI84 5.0
199912  B00004RYGX 5.0
199912  B00004CXX9 5.0
200001  B00002N8SM 5.0
200001  B00004CXX9 3.667
200001  B00004CI84 3.0
200001  B00004RYGX 3.0
200002  B00004CXX9 4.0
200002  B00004RYGX 4.0
200002  B00004CI84 4.0
200006  B00004CI84 5.0
200006  B00002Z754 5.0
200006  B00004RYGX 5.0
200006  B00004CXX9 5.0
200007  B00004RAMX 5.0
200008  B00004CXX9 5.0
200008  B00004CI84 5.0
200008  B00004RYGX 5.0
200008  B00004S1C5 5.0
200008  B00004S1C6 5.0
200010  B00004CI84 5.0
200010  B00004CXX9 5.0
200010  B00004RYGX 5.0

Spark
199910 5.0 [0006641040]
```

```
199912 5.0 [B00004CI84, B00004CXX9, B00004RYGX]
200001 5.0 [B00002N8SM]
200001 3.666 [B00004CXX9]
200001 3.0 [B00004CI84, B00004RYGX]
200002 4.0 [B00004CI84, B00004CXX9, B00004RYGX]
200006 5.0 [B00002Z754, B00004CXX9, B00004RYGX, B00004CI84]
200007 5.0 [B00004RAMX]
200008 5.0 [B00004CI84, B00004S1C5, B00004CXX9, B00004RYGX, B00004S1C6]
200010 5.0 [B00004CXX9, B00004RYGX, B00004CI84]
```

1.5 Risultati

Tutti i grafici riportati hanno il **tempo** in secondi sull'asse **y** e la dimensione dell'**input** in MB sull'asse **x**.

Input [MB]	HadoopTop5 Locale	HadoopTop5 AWS
5,2	3	30
27,3	4	29
73,6	5	30
192,9	6	33
299,1	8	34
3580	51	89
6570	91,12	131
9560	138	191
12550	228	227
14040	317,399	246

Figure 1: Hadoop Job1 Tabella

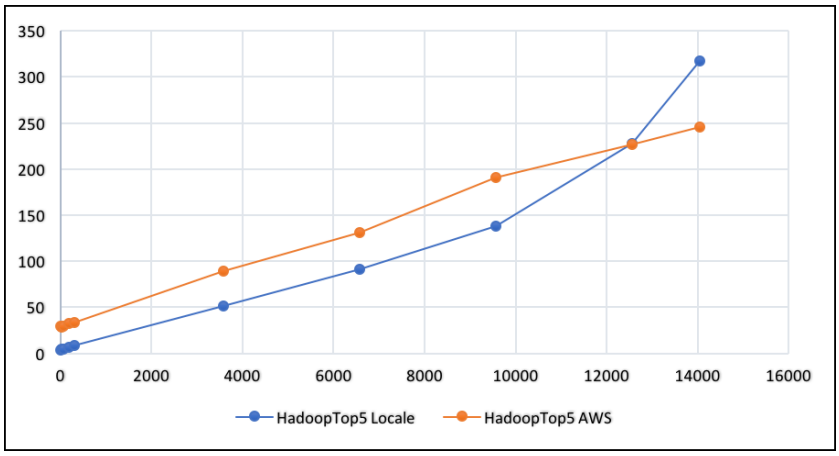

Figure 2: Hadoop Job1

L'input è stato incrementato gradualmente fino a 14000 MB. Da 10000 MB in poi si può notare che le prestazioni cominciano a migliorare su AWS. In locale con input pari a 15000 MB l'esecuzione terminava con l'errore "OutOfMemory" poichè a parità di yearMonth e Avg, il numero di prodotti da tenere in considerazione è troppo elevato. Il problema è quindi dovuto al fatto che i nuovi file vengono generati variado il timestamp.

Input [MB]	HiveTop5 locale	HiveTop5 AWS
5,2	7,068	14,999
27,3	8,068	13,199
73,6	9,283	20,735
192,9	13,005	24,398
299,1	14,792	39
3580	118,34	106,679
6570	195	160,75
9560	317	204
12550	377,585	246
15530	491	287

Figure 3: Hive Job1 Tabella

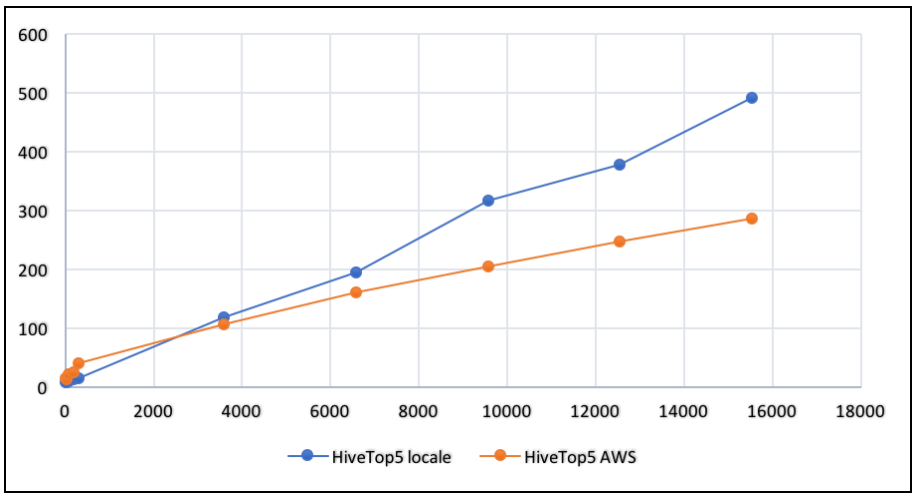

Figure 4: Hive Job1
I primi 4 punti del grafico sono relativi ai dataset forniti. Dal grafico si nota che all'aumentare dell'input l'esecuzione su AWS migliora in termini di prestazioni rispetto all'esecuzione locale.

Input [MB]	SparkTop5 Local	SparkTop5 AWS
5,2	4,544	24
27,3	5,429	26
73,6	6,569	29
192,9	8,912	32
299,1	10	33
3580	64	57
6570	107	78
9560	135	99
12550	184	116
15530	215	131

Figure 5: Spark Job1 Tabella

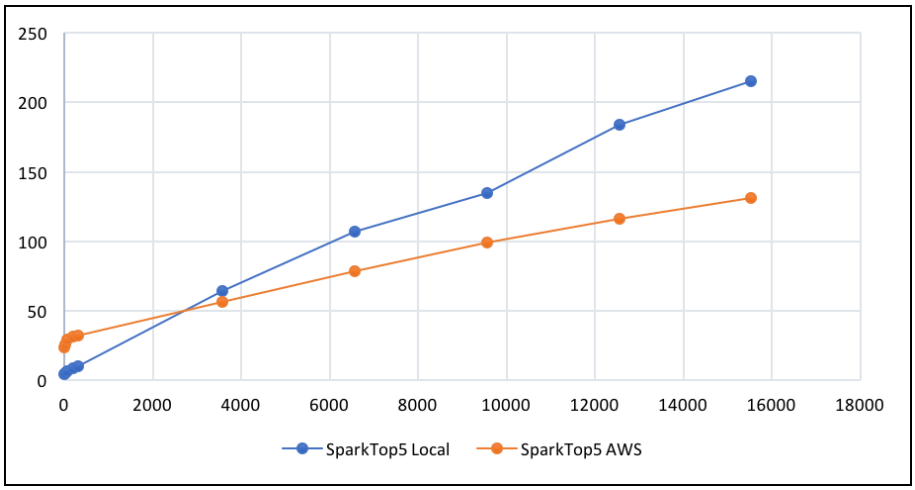

Figure 6: Spark Job1

2 Job 2: Top 10 Products per User

2.1 Hadoop

Pseudo

```
Map(line){
   tokens = read and split line by "\t";
   key = tokens [2] + tokens[1] // userID and prodID
   value = tokens[6] //score
   return (key, value); //key is userID prodID, value is the score
   }

Shuffle and Sort

    //In reduce we have defined a map structure where UserID is the key,innerMap is the
        value
map(key=userID, value=innerMap);
   //innerMap is an auxiliary map created for each key in map and it stores all products
       for an avg
innerMap(key=average, value=list(prodID));


Reduce(key, value){
   do average for each userID prodId;
   (userID, prodID) = split(key);
   if (!map.contains(userID))
       then add element in map;
   else if (!map.get(userID).contains(average))
      then add elements to innerMap;
   else
      add only prodId to innerMap.value; //append product to list of products
}

//in cleanUp we iterate on map and print top 10 products
CleanUp(context){
   print top 10 products;
}
```

In questo caso l'uso di TreeSet non ha provocato il problema che si era avuto con Hadoop su Job1. Ovvero nell'esercizio precedente il TreeSet portava all'errore OutOfMemory perchè a partità di AnnoMese e Avg si avevano troppi prodID da salvare in un Set. Qui invece a parit di UserID e Avg il numero di prodID da inserire in un Set limitato.

2.2 Hive

Code

```
CREATE TABLE resultEs2 AS
SELECT ps.userId, ps.productId, ps.avg FROM (
   SELECT userId, productId, ROUND(AVG(CAST(score AS FLOAT)),3) AS avg FROM reviews GROUP
       BY userId, productId
   ) ps
   ORDER BY ps.userId, ps.avg DESC;

SELECT productId, avg FROM resultES2 WHERE ranking(userId,avg) < 10;
```

2.3 Spark

Pseudo

```
tokens = read and split line by "\t";
create JavaPairRDD of ((userID prodID), score);
create JavaPairRDD of ((userID prodID), avg); //for each (userID, prodID) we have avg as
    sum(score)/number of score
create JavaPairRDD of ((userID avg), prodID);
group and sort by key;
create JavaPairRDD of (userID, AvgProdIDs)
group by key
for each userID print top 10 element
  userID, avg , list of products
```

2.4 Output

Di seguito vengono riportate le prime righe di output sul file 1999-2006.csv:

```
Hadoop
A100CY9WRC18I2 B000CQG84Y 1.0
A101CCC619GN4S B00017L1UK 5.0
A101VS17YZ5ZEJ B0004LW990 5.0
A103OZ75AVET1Y B000CBOR60 5.0
A1048CYU0OV4O8 B00004CI84 5.0
A1048CYU0OV4O8 B00004CXX9 5.0
A1048CYU0OV4O8 B00004RYGX 5.0
A105981PIJDJUU B000FFLHSY 4.0
A106E0DP6X12NW B0001ES9F8 1.0
A106E0DP6X12NW B0007NOWMM 1.0
A106MCEFKHCTX9 B000DZFMEQ 5.0
A106X6HMD3NE76 B0002QEKPI 5.0
A107OAJUDTZXTC B0007SNZQ6 5.0
A107OAJUDTZXTC B000FDMLUO 5.0
A107OAJUDTZXTC B000CROPGQ 2.0
A108KZJAD7SME B00014FT06 4.0
A1097WLPUXJYXK B0000DZ0Q4 5.0
A10A8TD5C67LBT B0002UW3JO 5.0
A10B113DVKWX4Q B000FKL0D6 5.0
A10B6QI9B6YZOE B000EPOC28 5.0
A10CMEPEV19WAB B000FNB3AI 5.0
A10CMEPEV19WAB B000FNEX8C 5.0
A10CMEPEV19WAB B000FNH1C2 5.0
A10CMEPEV19WAB B000FVWJAI 5.0
A10DAMI04QFCBP B0000TU98E 5.0
A10FBA937MA91B B0007IQQV2 5.0
A10FOB6FBXPJY5 B0000UVRXY 5.0
A10G14BNGMVQV B000084F04 5.0
A10GV9VYNLIXUQ B00008DFK5 5.0
A10GV9VYNLIXUQ B0002DHNXC 5.0
A10GV9VYNLIXUQ B0009YD7P2 5.0
A10GV9VYNLIXUQ B000SP1CWW 5.0

Hive
A100CY9WRC18I2 B000CQG84Y 1.0
A101CCC619GN4S B00017L1UK 5.0
A101VS17YZ5ZEJ B0004LW990 5.0
A103OZ75AVET1Y B000CBOR60 5.0
```

```
A1048CYU0OV4O8 B00004CI84 5.0
A1048CYU0OV4O8 B00004CXX9 5.0
A1048CYU0OV4O8 B00004RYGX 5.0
A105981PIJDJUU B000FFLHSY 4.0
A106E0DP6X12NW B0001ES9F8 1.0
A106E0DP6X12NW B0007NOWMM 1.0
A106MCEFKHCTX9 B000DZFMEQ 5.0
A106X6HMD3NE76 B0002QEKPI 5.0
A107OAJUDTZXTC B0007SNZQ6 5.0
A107OAJUDTZXTC B000FDMLUO 5.0
A107OAJUDTZXTC B000CROPGQ 2.0
A108KZJAD7SME B00014FT06 4.0
A1097WLPUXJYXK B0000DZ0Q4 5.0
A10A8TD5C67LBT B0002UW3JO 5.0
A10B113DVKWX4Q B000FKL0D6 5.0
A10B6QI9B6YZOE B000EPOC28 5.0
A10CMEPEV19WAB B000FNB3AI 5.0
A10CMEPEV19WAB B000FNEX8C 5.0
A10CMEPEV19WAB B000FNH1C2 5.0
A10CMEPEV19WAB B000FVWJAI 5.0
A10DAMI04QFCBP B0000TU98E 5.0
A10FBA937MA91B B0007IQQV2 5.0
A10FOB6FBXPJY5 B0000UVRXY 5.0
A10G14BNGMVQV B000084F04 5.0
A10GV9VYNLIXUQ B00008DFK5 5.0
A10GV9VYNLIXUQ B0002DHNXC 5.0
A10GV9VYNLIXUQ B0009YD7P2 5.0
A10GV9VYNLIXUQ B000SP1CWW 5.0

Spark
A100CY9WRC18I2 1.0 [B000CQG84Y]
A101CCC619GN4S 5.0 [B00017L1UK]
A101VS17YZ5ZEJ 5.0 [B0004LW990]
A103OZ75AVET1Y 5.0 [B000CBOR60]
A1048CYU0OV4O8 5.0 [B00004CXX9, B00004CI84, B00004RYGX]
A105981PIJDJUU 4.0 [B000FFLHSY]
A106E0DP6X12NW 1.0 [B0007NOWMM, B0001ES9F8]
A106MCEFKHCTX9 5.0 [B000DZFMEQ]
A106X6HMD3NE76 5.0 [B0002QEKPI]
A107OAJUDTZXTC 5.0 [B0007SNZQ6, B000FDMLUO]
A107OAJUDTZXTC 2.0 [B000CROPGQ]
A108KZJAD7SME 4.0 [B00014FT06]
A1097WLPUXJYXK 5.0 [B0000DZ0Q4]
A10A8TD5C67LBT 5.0 [B0002UW3JO]
A10B113DVKWX4Q 5.0 [B000FKL0D6]
A10B6QI9B6YZOE 5.0 [B000EPOC28]
A10CMEPEV19WAB 5.0 [B000FNH1C2, B000FNEX8C, B000FNB3AI, B000FVWJAI]
A10DAMI04QFCBP 5.0 [B0000TU98E]
A10FBA937MA91B 5.0 [B0007IQQV2]
A10FOB6FBXPJY5 5.0 [B0000UVRXY]
A10G14BNGMVQV 5.0 [B000084F04]
A10GV9VYNLIXUQ 5.0 [B00008DFK5, B0009YD7P2, B000SP1CWW, B0002DHNXC]
```

2.5 Risultati

Tutti i grafici riportati hanno il **tempo** in secondi sull'asse **y** e la dimensione dell'**input** in MB sull'asse **x**.

Input [MB]	hadoopTop10 locale	hadoopTop10 AWS
5,2	3,095	27
27,3	4,201	28
73,6	5,285	30
192,9	7,233	31
299,1	9,44	30
3580	44,27	79,74
6570	78	118
9560	111	168,08
12550	136	207
15530	174	235,94
18520	219	284,21
23000	261	363

Figure 7: Hadoop Job2 Tabella

Figure 8: Hadoop Job2

Dal grafico si puó notare che l'esecuzione locale è migliore che su AWS in termini di prestazioni. Questo comportamento è giustificabile dal fatto che per queste dimensioni di input in locale non si manifestano problemi per la gestione di strutture come la Map utilizzata. A parità di UserID e Avg, il numero di prodotti che si possono avere è infatti contenuto e non comporta OutofMemory.

Input [MB]	HiveTop10 locale	HiveTop10 AWS
5,2	6,2	13,893
27,3	8,151	15,293
73,6	10,5	19,093
192,9	12,3	21,561
299,1	18	29,771
3580	89	92,257
6570	142,2	132,539
9560	198,871	168,374
12550	266,425	201,347
15530	338,7	237,214

Figure 9: Hive Job2 Tabella

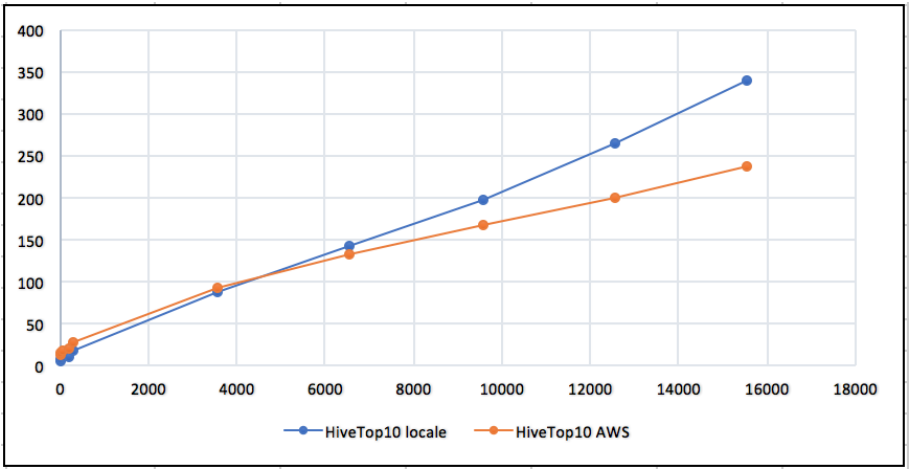

Figure 10: Hive Job2

Input [MB]	sparkTop10 locale	sparkTop10 AWS
5,2	4,887	24
27,3	6,576	28
73,6	8,265	29
192,9	12,641	35
299,1	16	39
3580	49	51
6570	80	64
9560	109	79
12550	130	89
15530	159	103

Figure 11: Spark Job2 Tabella

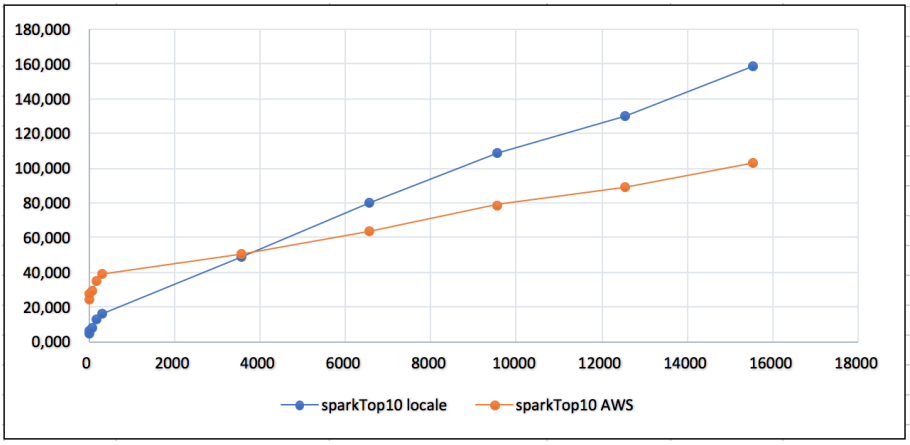

Figure 12: Spark Job2

3 Job 3: Simil preferences of 2 Users

3.1 Hadoop

Pseudo

```
Map1(line){
   tokens = read and split line by "\t"
   filter line per score >= 4
   key = tokens[1]; //tokens[1] is the prodID
   value = tokens[2]; //tokens[2] is the userID
   return (key, value); //prodID, userID
}

Shuffle and Sort Map1

Reduce1(key, value){ //key is prodID and value is all userID that prefer this product
    return all pairs of userID and the prodID
    //example: ((usr1,usr2)&(usr1,usr3)&(usr1,usrn), prodID)
}

Map2(line){ // line is output of reduce1
   tokens = read and split line by "&" //tokens contains pairs of userID
   for each pair (usr_i,usr_j) in key
      return (usr_i,usr_j), prodID
}

Shuffle and Sort Map2

Reduce2(key, value){ // key is a pair of userID and value are prodID
    if value.length >= 3
      writeOnFile (key, value)
}
```

3.2 Hive

Code

```
CREATE TABLE resultEs3 AS
SELECT r1.userId AS user1, r2.userId AS user2, COUNT(DISTINCT(r1.productId)) AS cont,
    collect_set(r1.productId) as prodList
FROM reviews r1 JOIN reviews r2 ON r1.productId = r2.productId
WHERE r1.score>3 AND r2.score>3 AND r1.userId > r2.userId
GROUP BY r1.userId, r2.userId;

SELECT user1, user2, prodList FROM resultEs3 WHERE cont>2 ORDER BY user1, user2;
```

3.3 Spark

Pseudo

```
tokens = read and split line by "\t";
create JavaRDD of (inputFile) where score >= 4 //score is tokens[6]
prodIDUserID = JavaPairRDD of (prodID, userID) //from the JavaRDD we create JavaPairRDD
join(prodIDUserID, prodIDUserID);
filter on userID1 < userID2; //filters duplicates and create couple user with prodID
group by key; //obtains ((user1 user2), list(prodID) )
filter on values.length >= 3;
sort by key;
for each (userID1, userID2) print
   userID1, userID2 , list of products;
```

3.4 Output

Di seguito vengono riportate le prime righe di output per il file 1999-2006.csv:

```
Hadoop
A1048CYU0OV4O8,A157XTSMJH9XA4 [B00004RYGX, B00004CXX9, B00004CI84]
A1048CYU0OV4O8,A19JYLHD94K94D [B00004CI84, B00004RYGX, B00004CXX9]
A1048CYU0OV4O8,A1BZEGSNBB7DVS [B00004RYGX, B00004CI84, B00004CXX9]
A1048CYU0OV4O8,A1CAA94EOP0J2S [B00004CXX9, B00004CI84, B00004RYGX]
A1048CYU0OV4O8,A1CZICCYP2M5PX [B00004CI84, B00004CXX9, B00004RYGX]
A1048CYU0OV4O8,A1DU58OZJNPUHV [B00004CXX9, B00004RYGX, B00004CI84]
A1048CYU0OV4O8,A1E5AVR7QJN8HF [B00004RYGX, B00004CXX9, B00004CI84]
A1048CYU0OV4O8,A1FJOY14X3MUHE [B00004CI84, B00004CXX9, B00004RYGX]
A1048CYU0OV4O8,A1GB1Q193DNFGR [B00004CXX9, B00004RYGX, B00004CI84]
A1048CYU0OV4O8,A1HWMNSQF14MP8 [B00004CI84, B00004RYGX, B00004CXX9]
A1048CYU0OV4O8,A1JZV9MCT6KOX4 [B00004CXX9, B00004CI84, B00004RYGX]
A1048CYU0OV4O8,A1NVW4NUXPOSJB [B00004RYGX, B00004CI84, B00004CXX9]
A1048CYU0OV4O8,A1OBPHRXHZF8P6 [B00004CI84, B00004CXX9, B00004RYGX]
A1048CYU0OV4O8,A1OP3SQP78M1PP [B00004CXX9, B00004CI84, B00004RYGX]
A1048CYU0OV4O8,A1RSXP7MB772E3 [B00004RYGX, B00004CXX9, B00004CI84]
A1048CYU0OV4O8,A1SWVKJIQWW33K [B00004RYGX, B00004CI84, B00004CXX9]
A1048CYU0OV4O8,A1TW9ZGRDQQZ2Y [B00004CXX9, B00004RYGX, B00004CI84]
A1048CYU0OV4O8,A1VTP9QUHAOTXR [B00004CXX9, B00004CI84, B00004RYGX]
A1048CYU0OV4O8,A1ZHO86GZYL5MZ [B00004CXX9, B00004RYGX, B00004CI84]
A1048CYU0OV4O8,A1ZPALFR6ZYKRH [B00004CXX9, B00004CI84, B00004RYGX]
A1048CYU0OV4O8,A212YFPHGXOGMK [B00004CI84, B00004RYGX, B00004CXX9]
A1048CYU0OV4O8,A21GKWZJL8BVJ7 [B00004RYGX, B00004CI84, B00004CXX9]
A1048CYU0OV4O8,A23GFTVIETX7DS [B00004CXX9, B00004RYGX, B00004CI84]
A1048CYU0OV4O8,A23QOAXJSWIBS6 [B00004CI84, B00004RYGX, B00004CXX9]
A1048CYU0OV4O8,A2A6NH6DPE0VXR [B00004CI84, B00004RYGX, B00004CXX9]
A1048CYU0OV4O8,A2DEE7F9XKP3ZR [B00004RYGX, B00004CI84, B00004CXX9]
A1048CYU0OV4O8,A2E1UPR4LZJSF2 [B00004CI84, B00004CXX9, B00004RYGX]
A1048CYU0OV4O8,A2HIZRVOKXKZ52 [B00004CI84, B00004CXX9, B00004RYGX]
A1048CYU0OV4O8,A2JTZI7Y58X9W5 [B00004CI84, B00004RYGX, B00004CXX9]
A1048CYU0OV4O8,A2KJX73VG7RRCN [B00004RYGX, B00004CI84, B00004CXX9]
A1048CYU0OV4O8,A2N2C3DZ47WO6C [B00004CXX9, B00004CI84, B00004RYGX]
A1048CYU0OV4O8,A2NUHWMHA9XNKV [B00004CI84, B00004CXX9, B00004RYGX]
A1048CYU0OV4O8,A2QOI90Y1HNTP9 [B00004CI84, B00004CXX9, B00004RYGX]
A1048CYU0OV4O8,A2RLGG5XB5ID34 [B00004CI84, B00004CXX9, B00004RYGX]
A1048CYU0OV4O8,A2SXUINNJD8R25 [B00004CXX9, B00004RYGX, B00004CI84]
A1048CYU0OV4O8,A2WTQL7T70Z1H1 [B00004CXX9, B00004RYGX, B00004CI84]
A1048CYU0OV4O8,A30CE2CDDBVNDE [B00004CI84, B00004CXX9, B00004RYGX]
A1048CYU0OV4O8,A31RM5QU797HPJ [B00004CI84, B00004CXX9, B00004RYGX]
```

```
A1048CYUOOV4O8,A32JKNQ6BABMQ2 [B00004CI84, B00004CXX9, B00004RYGX]
A1048CYUOOV4O8,A344SMIA5JECGM [B00004RYGX, B00004CXX9, B00004CI84]
A1048CYUOOV4O8,A34NBH479RB0E [B00004CI84, B00004CXX9, B00004RYGX]
A1048CYUOOV4O8,A3C3BAQDZWH5YE [B00004CI84, B00004RYGX, B00004CXX9]
A1048CYUOOV4O8,A3ICHEO78NFE4Y [B00004CXX9, B00004RYGX, B00004CI84]
A1048CYUOOV4O8,A3K3YJWVON54ZO [B00004RYGX, B00004CI84, B00004CXX9]
A1048CYUOOV4O8,A3R2YBOWTTBOIJ [B00004CXX9, B00004RYGX, B00004CI84]
A1048CYUOOV4O8,A48VQ893Z2KQR [B00004CI84, B00004RYGX, B00004CXX9]
A1048CYUOOV4O8,A4RUNO1NO4827 [B00004RYGX, B00004CXX9, B00004CI84]
A1048CYUOOV4O8,AAI57M3OXP5NK [B00004CXX9, B00004CI84, B00004RYGX]
A1048CYUOOV4O8,ABUO5VJAR81ZZ [B00004CI84, B00004CXX9, B00004RYGX]
A1048CYUOOV4O8,ACJR7EQF9S6FP [B00004CXX9, B00004RYGX, B00004CI84]
A1048CYUOOV4O8,ADIDQRLLR4KBQ [B00004RYGX, B00004CI84, B00004CXX9]
A1048CYUOOV4O8,AEPJYNONAX9N4 [B00004RYGX, B00004CI84, B00004CXX9]
A1048CYUOOV4O8,API663PFYRQCP [B00004RYGX, B00004CI84, B00004CXX9]
A1048CYUOOV4O8,ASCVAUPIB31OD [B00004CI84, B00004CXX9, B00004RYGX]
A1048CYUOOV4O8,AW3FTPCBPVQNB [B00004CI84, B00004CXX9, B00004RYGX]
A10CMEPEV19WAB,A1DS27DOV1NQJO [B000FNH1C2, B000FNEX8C, B000FNB3AI]
A10CMEPEV19WAB,A1I7RRDDARP414 [B000FNH1C2, B000FNB3AI, B000FNEX8C]
A10CMEPEV19WAB,A1SYAQVM7E8VFP [B000FNB3AI, B000FNEX8C, B000FNH1C2]
```

Hive

```
A1048CYUOOV4O8 A157XTSMJH9XA4 ["B00004CI84","B00004CXX9","B00004RYGX"]
A1048CYUOOV4O8 A19JYLHD94K94D ["B00004CI84","B00004CXX9","B00004RYGX"]
A1048CYUOOV4O8 A1BZEGSNBB7DVS ["B00004CI84","B00004CXX9","B00004RYGX"]
A1048CYUOOV4O8 A1CAA94EOPOJ2S ["B00004CI84","B00004CXX9","B00004RYGX"]
A1048CYUOOV4O8 A1CZICCYP2M5PX ["B00004CI84","B00004CXX9","B00004RYGX"]
A1048CYUOOV4O8 A1DU58OZJNPUHV ["B00004CI84","B00004CXX9","B00004RYGX"]
A1048CYUOOV4O8 A1E5AVR7QJN8HF ["B00004CI84","B00004CXX9","B00004RYGX"]
A1048CYUOOV4O8 A1FJOY14X3MUHE ["B00004CI84","B00004CXX9","B00004RYGX"]
A1048CYUOOV4O8 A1GB1Q193DNFGR ["B00004CI84","B00004CXX9","B00004RYGX"]
A1048CYUOOV4O8 A1HWMNSQF14MP8 ["B00004CI84","B00004CXX9","B00004RYGX"]
A1048CYUOOV4O8 A1JZV9MCT6KOX4 ["B00004CI84","B00004CXX9","B00004RYGX"]
A1048CYUOOV4O8 A1NVW4NUXPOSJB ["B00004CI84","B00004CXX9","B00004RYGX"]
A1048CYUOOV4O8 A1OBPHRXHZF8P6 ["B00004CI84","B00004CXX9","B00004RYGX"]
A1048CYUOOV4O8 A1OP3SQP78M1PP ["B00004CI84","B00004CXX9","B00004RYGX"]
A1048CYUOOV4O8 A1RSXP7MB772E3 ["B00004CI84","B00004CXX9","B00004RYGX"]
A1048CYUOOV4O8 A1SWVKJIQWW33K ["B00004CI84","B00004CXX9","B00004RYGX"]
A1048CYUOOV4O8 A1TW9ZGRDQQZ2Y ["B00004CI84","B00004CXX9","B00004RYGX"]
A1048CYUOOV4O8 A1VTP9QUHAOTXR ["B00004CI84","B00004CXX9","B00004RYGX"]
A1048CYUOOV4O8 A1ZHO86GZYL5MZ ["B00004CI84","B00004CXX9","B00004RYGX"]
A1048CYUOOV4O8 A1ZPALFR6ZYKRH ["B00004CI84","B00004CXX9","B00004RYGX"]
A1048CYUOOV4O8 A212YFPHGXOGMK ["B00004CI84","B00004CXX9","B00004RYGX"]
A1048CYUOOV4O8 A21GKWZJL8BVJ7 ["B00004CI84","B00004CXX9","B00004RYGX"]
A1048CYUOOV4O8 A23GFTVIETX7DS ["B00004CI84","B00004CXX9","B00004RYGX"]
A1048CYUOOV4O8 A23QOAXJSWIBS6 ["B00004CI84","B00004CXX9","B00004RYGX"]
A1048CYUOOV4O8 A2A6NH6DPEOVXR ["B00004CI84","B00004CXX9","B00004RYGX"]
A1048CYUOOV4O8 A2DEE7F9XKP3ZR ["B00004CI84","B00004CXX9","B00004RYGX"]
A1048CYUOOV4O8 A2E1UPR4LZJSF2 ["B00004CI84","B00004CXX9","B00004RYGX"]
A1048CYUOOV4O8 A2HIZRVOKXKZ52 ["B00004CI84","B00004CXX9","B00004RYGX"]
A1048CYUOOV4O8 A2JTZI7Y58X9W5 ["B00004CI84","B00004CXX9","B00004RYGX"]
A1048CYUOOV4O8 A2KJX73VG7RRCN ["B00004CI84","B00004CXX9","B00004RYGX"]
A1048CYUOOV4O8 A2N2C3DZ47WO6C ["B00004CI84","B00004CXX9","B00004RYGX"]
A1048CYUOOV4O8 A2NUHWMHA9XNKV ["B00004CI84","B00004CXX9","B00004RYGX"]
A1048CYUOOV4O8 A2QOI9OY1HNTP9 ["B00004CI84","B00004CXX9","B00004RYGX"]
A1048CYUOOV4O8 A2RLGG5XB5ID34 ["B00004CI84","B00004CXX9","B00004RYGX"]
A1048CYUOOV4O8 A2SXUINNJD8R25 ["B00004CI84","B00004CXX9","B00004RYGX"]
A1048CYUOOV4O8 A2WTQL7T7OZ1H1 ["B00004CI84","B00004CXX9","B00004RYGX"]
```

```
A1048CYUOOV4O8 A3OCE2CDDBVNDE ["B00004CI84","B00004CXX9","B00004RYGX"]
A1048CYUOOV4O8 A31RM5QU797HPJ ["B00004CI84","B00004CXX9","B00004RYGX"]
A1048CYUOOV4O8 A32JKNQ6BABMQ2 ["B00004CI84","B00004CXX9","B00004RYGX"]
A1048CYUOOV4O8 A344SMIA5JECGM ["B00004CI84","B00004CXX9","B00004RYGX"]
A1048CYUOOV4O8 A34NBH479RBOE ["B00004CI84","B00004CXX9","B00004RYGX"]
A1048CYUOOV4O8 A3C3BAQDZWH5YE ["B00004CI84","B00004CXX9","B00004RYGX"]
A1048CYUOOV4O8 A3ICHEO78NFE4Y ["B00004CI84","B00004CXX9","B00004RYGX"]
A1048CYUOOV4O8 A3K3YJWVON54ZO ["B00004CI84","B00004CXX9","B00004RYGX"]
A1048CYUOOV4O8 A3R2YBOWTTBOIJ ["B00004CI84","B00004CXX9","B00004RYGX"]
A1048CYUOOV4O8 A48VQ893Z2KQR ["B00004CI84","B00004CXX9","B00004RYGX"]
A1048CYUOOV4O8 A4RUNO1NO4827 ["B00004CI84","B00004CXX9","B00004RYGX"]
A1048CYUOOV4O8 AAI57M3OXP5NK ["B00004CI84","B00004CXX9","B00004RYGX"]
A1048CYUOOV4O8 ABUO5VJAR81ZZ ["B00004CI84","B00004CXX9","B00004RYGX"]
A1048CYUOOV4O8 ACJR7EQF9S6FP ["B00004CI84","B00004CXX9","B00004RYGX"]
A1048CYUOOV4O8 ADIDQRLLR4KBQ ["B00004CI84","B00004CXX9","B00004RYGX"]
A1048CYUOOV4O8 AEPJYNONAX9N4 ["B00004CI84","B00004CXX9","B00004RYGX"]
A1048CYUOOV4O8 API663PFYRQCP ["B00004CI84","B00004CXX9","B00004RYGX"]
A1048CYUOOV4O8 ASCVAUPIB31OD ["B00004CI84","B00004CXX9","B00004RYGX"]
A1048CYUOOV4O8 AW3FTPCBPVQNB ["B00004CI84","B00004CXX9","B00004RYGX"]
A10CMEPEV19WAB A1DS27DOV1NQJO ["B000FNB3AI","B000FNEX8C","B000FNH1C2"]
A10CMEPEV19WAB A1I7RRDDARP414 ["B000FNB3AI","B000FNEX8C","B000FNH1C2"]
A10CMEPEV19WAB A1SYAQVM7E8VFP ["B000FNB3AI","B000FNEX8C","B000FNH1C2"]
```

Spark

```
(A1048CYUOOV4O8 A157XTSMJH9XA4,[B00004RYGX, B00004CXX9, B00004CI84])
(A1048CYUOOV4O8 A19JYLHD94K94D,[B00004RYGX, B00004CXX9, B00004CI84])
(A1048CYUOOV4O8 A1BZEGSNBB7DVS,[B00004RYGX, B00004CXX9, B00004CI84])
(A1048CYUOOV4O8 A1CAA94EOPOJ2S,[B00004RYGX, B00004CXX9, B00004CI84])
(A1048CYUOOV4O8 A1CZICCYP2M5PX,[B00004RYGX, B00004CXX9, B00004CI84])
(A1048CYUOOV4O8 A1DU58OZJNPUHV,[B00004RYGX, B00004CXX9, B00004CI84])
(A1048CYUOOV4O8 A1E5AVR7QJN8HF,[B00004RYGX, B00004CXX9, B00004CI84])
(A1048CYUOOV4O8 A1FJOY14X3MUHE,[B00004RYGX, B00004CXX9, B00004CI84])
(A1048CYUOOV4O8 A1GB1Q193DNFGR,[B00004RYGX, B00004CXX9, B00004CI84])
(A1048CYUOOV4O8 A1HWMNSQF14MP8,[B00004RYGX, B00004CXX9, B00004CI84])
(A1048CYUOOV4O8 A1JZV9MCT6KOX4,[B00004RYGX, B00004CXX9, B00004CI84])
(A1048CYUOOV4O8 A1NVW4NUXPOSJB,[B00004RYGX, B00004CXX9, B00004CI84])
(A1048CYUOOV4O8 A1OBPHRXHZF8P6,[B00004RYGX, B00004CXX9, B00004CI84])
(A1048CYUOOV4O8 A1OP3SQP78M1PP,[B00004RYGX, B00004CXX9, B00004CI84])
(A1048CYUOOV4O8 A1RSXP7MB772E3,[B00004RYGX, B00004CXX9, B00004CI84])
(A1048CYUOOV4O8 A1SWVKJIQWW33K,[B00004RYGX, B00004CXX9, B00004CI84])
(A1048CYUOOV4O8 A1TW9ZGRDQQZ2Y,[B00004RYGX, B00004CXX9, B00004CI84])
(A1048CYUOOV4O8 A1VTP9QUHAOTXR,[B00004RYGX, B00004CXX9, B00004CI84])
(A1048CYUOOV4O8 A1ZHO86GZYL5MZ,[B00004RYGX, B00004CXX9, B00004CI84])
(A1048CYUOOV4O8 A1ZPALFR6ZYKRH,[B00004RYGX, B00004CXX9, B00004CI84])
(A1048CYUOOV4O8 A212YFPHGXOGMK,[B00004RYGX, B00004CXX9, B00004CI84])
(A1048CYUOOV4O8 A21GKWZJL8BVJ7,[B00004RYGX, B00004CXX9, B00004CI84])
(A1048CYUOOV4O8 A23GFTVIETX7DS,[B00004RYGX, B00004CXX9, B00004CI84])
(A1048CYUOOV4O8 A23QOAXJSWIBS6,[B00004RYGX, B00004CXX9, B00004CI84])
(A1048CYUOOV4O8 A2A6NH6DPEOVXR,[B00004RYGX, B00004CXX9, B00004CI84])
(A1048CYUOOV4O8 A2DEE7F9XKP3ZR,[B00004RYGX, B00004CXX9, B00004CI84])
(A1048CYUOOV4O8 A2E1UPR4LZJSF2,[B00004RYGX, B00004CXX9, B00004CI84])
(A1048CYUOOV4O8 A2HIZRVOKXKZ52,[B00004RYGX, B00004CXX9, B00004CI84])
(A1048CYUOOV4O8 A2JTZI7Y58X9W5,[B00004RYGX, B00004CXX9, B00004CI84])
(A1048CYUOOV4O8 A2KJX73VG7RRCN,[B00004RYGX, B00004CXX9, B00004CI84])
(A1048CYUOOV4O8 A2N2C3DZ47WO6C,[B00004RYGX, B00004CXX9, B00004CI84])
(A1048CYUOOV4O8 A2NUHWMHA9XNKV,[B00004RYGX, B00004CXX9, B00004CI84])
(A1048CYUOOV4O8 A2QOI9OY1HNTP9,[B00004RYGX, B00004CXX9, B00004CI84])
(A1048CYUOOV4O8 A2RLGG5XB5ID34,[B00004RYGX, B00004CXX9, B00004CI84])
(A1048CYUOOV4O8 A2SXUINNJD8R25,[B00004RYGX, B00004CXX9, B00004CI84])
```

```
(A1048CYU0OV4O8 A2WTQL7T70Z1H1,[B00004RYGX, B00004CXX9, B00004CI84])
(A1048CYU0OV4O8 A30CE2CDDBVNDE,[B00004RYGX, B00004CXX9, B00004CI84])
(A1048CYU0OV4O8 A31RM5QU797HPJ,[B00004RYGX, B00004CXX9, B00004CI84])
(A1048CYU0OV4O8 A32JKNQ6BABMQ2,[B00004RYGX, B00004CXX9, B00004CI84])
(A1048CYU0OV4O8 A344SMIA5JECGM,[B00004RYGX, B00004CXX9, B00004CI84])
(A1048CYU0OV4O8 A34NBH479RBOE,[B00004RYGX, B00004CXX9, B00004CI84])
(A1048CYU0OV4O8 A3C3BAQDZWH5YE,[B00004RYGX, B00004CXX9, B00004CI84])
(A1048CYU0OV4O8 A3ICHEO78NFE4Y,[B00004RYGX, B00004CXX9, B00004CI84])
(A1048CYU0OV4O8 A3K3YJWV0N54ZO,[B00004RYGX, B00004CXX9, B00004CI84])
(A1048CYU0OV4O8 A3R2YB0WTTB0IJ,[B00004RYGX, B00004CXX9, B00004CI84])
(A1048CYU0OV4O8 A48VQ893Z2KQR,[B00004RYGX, B00004CXX9, B00004CI84])
(A1048CYU0OV4O8 A4RUN01NO4827,[B00004RYGX, B00004CXX9, B00004CI84])
(A1048CYU0OV4O8 AAI57M3OXP5NK,[B00004RYGX, B00004CXX9, B00004CI84])
(A1048CYU0OV4O8 ABU05VJAR81ZZ,[B00004RYGX, B00004CXX9, B00004CI84])
(A1048CYU0OV4O8 ACJR7EQF9S6FP,[B00004RYGX, B00004CXX9, B00004CI84])
(A1048CYU0OV4O8 ADIDQRLLR4KBQ,[B00004RYGX, B00004CXX9, B00004CI84])
(A1048CYU0OV4O8 AEPJYN0NAX9N4,[B00004RYGX, B00004CXX9, B00004CI84])
(A1048CYU0OV4O8 API663PFYRQCP,[B00004RYGX, B00004CXX9, B00004CI84])
(A1048CYU0OV4O8 ASCVAUPIB31OD,[B00004RYGX, B00004CXX9, B00004CI84])
(A1048CYU0OV4O8 AW3FTPCBPVQNB,[B00004RYGX, B00004CXX9, B00004CI84])
(A10CMEPEV19WAB A1DS27D0V1NQJ0,[B000FNH1C2, B000FNEX8C, B000FNB3AI])
(A10CMEPEV19WAB A1I7RRDDARP414,[B000FNH1C2, B000FNEX8C, B000FNB3AI])
(A10CMEPEV19WAB A1SYAQVM7E8VFP,[B000FNH1C2, B000FNEX8C, B000FNB3AI])
```

3.5 Risultati

Tutti i grafici riportati hanno il **tempo** in secondi sull'asse **y** e la dimensione dell'**input** in MB sull'asse **x**.

Input [MB]	hadoopGustiAffini locale	hadoopGustiAffini AWS
5,2	4	60
27,3	7	60
73,6	12	63
192,9	41	72
299,1	63	77
3580	96	137
6570	112,48	157
9560	143	205
12550	162	249
15530	188	279

Figure 13: Hadoop Job3 Tabella

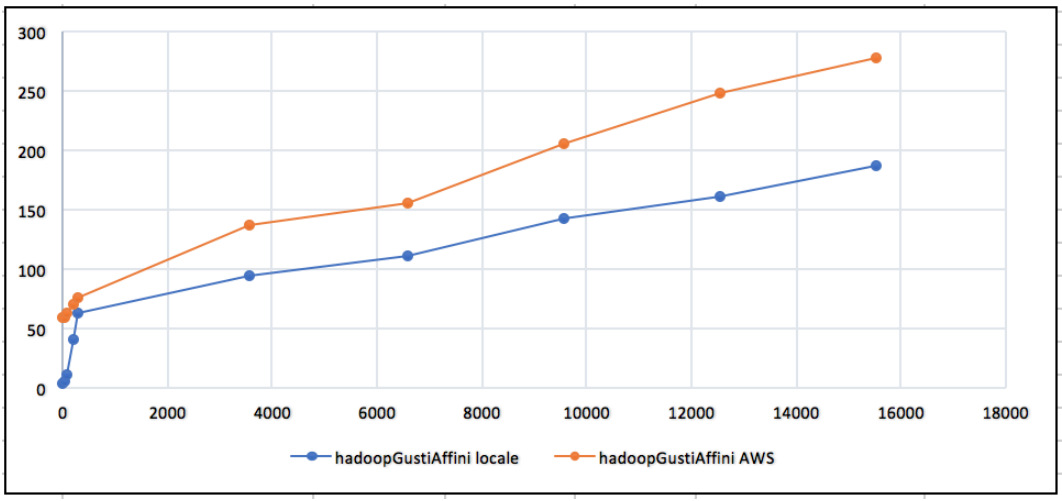

Figure 14: Hadoop Job3

Si è notatato che per dimensioni di input in tabella riportate, hadoop è milgiore in termini di prestazioni in locale rispetto al AWS. In questa implementazione si fatto utilizzo di 2 map e 2 reduce.

Input [MB]	HiveGustiAffini locale	HiveGustiAffini AWS
5,2	18,295	16,451
27,3	23	25,972
73,6	43,513	41,224
192,9	90,737	110,13
299,1	190,142	194,554
394,7	279,753	216,317

Figure 15: Hive Job3 Tabella

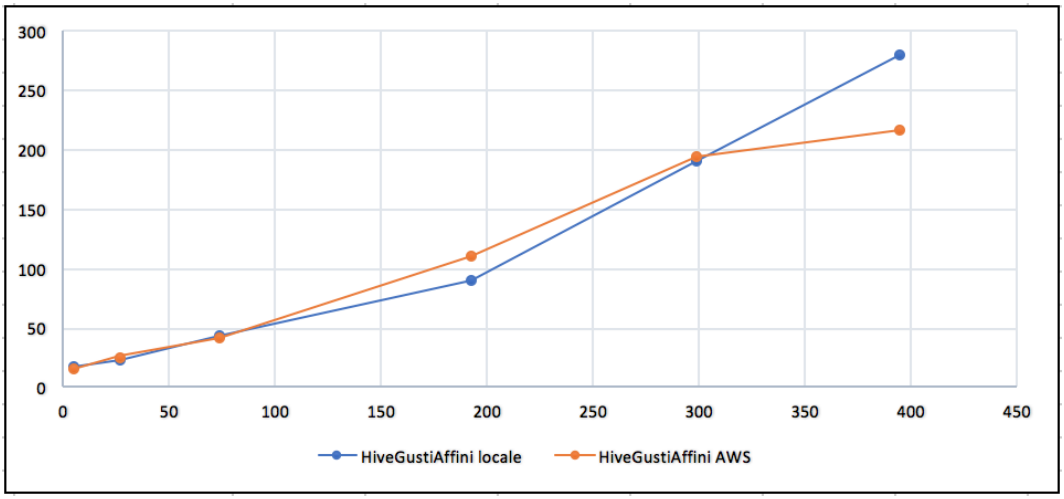

Figure 16: Hive Job3

Input [MB]	sparkGustiAffini locale	sparkGustiAffini AWS
5,2	5	26,25
27,3	7	30
73,6	13	35
192,9	53	49,31
299,1	101	60

Figure 17: Spark Job3 Tabella

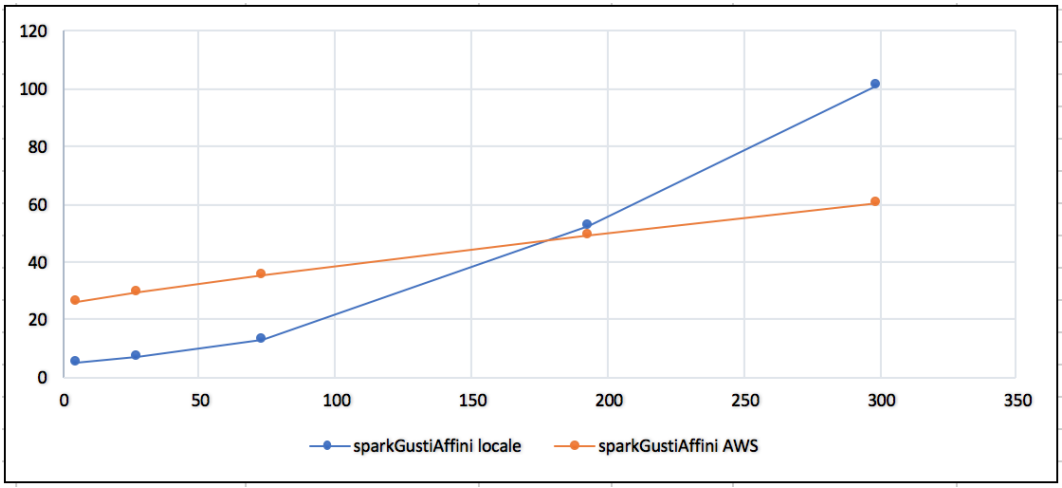

Figure 18: Spark Job3

4 Performance delle 3 tecnologie a confronto

Di seguito vengono riportati i grafici che confrontano le prestazioni di Hadoop, Hive e Spark, relaivamente allo stesso Job con esecuzioni in locale e su AWS. In generale risulta evidente che le prestazioni peggiori si hanno con Hive e le migliori con Spark.

Tutti i grafici riportati hanno il **tempo** in secondi sull'asse **y** e la dimensione dell'**input** in MB sull'asse **x**.

4.1 Job 1

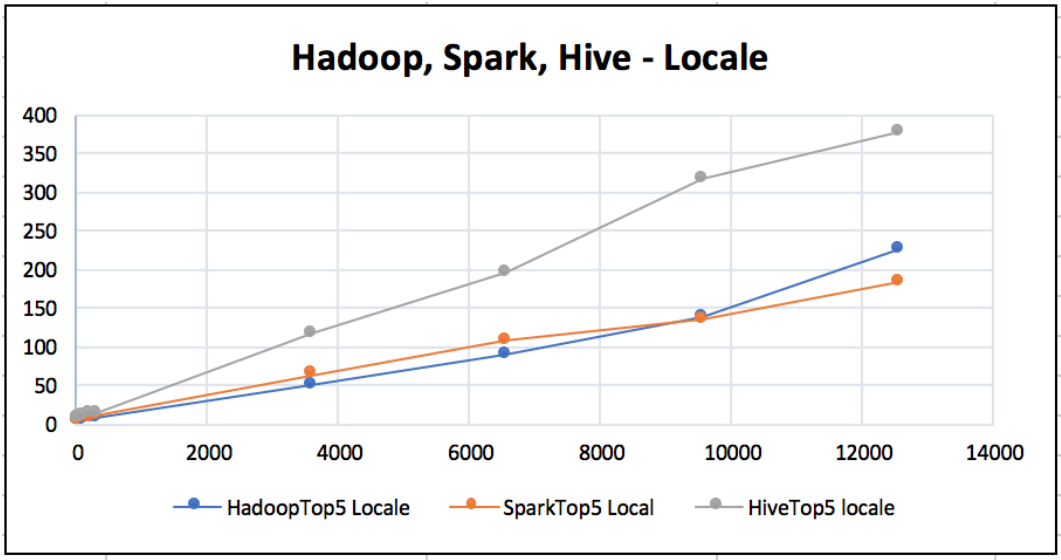

Figure 19: Performance Job 1

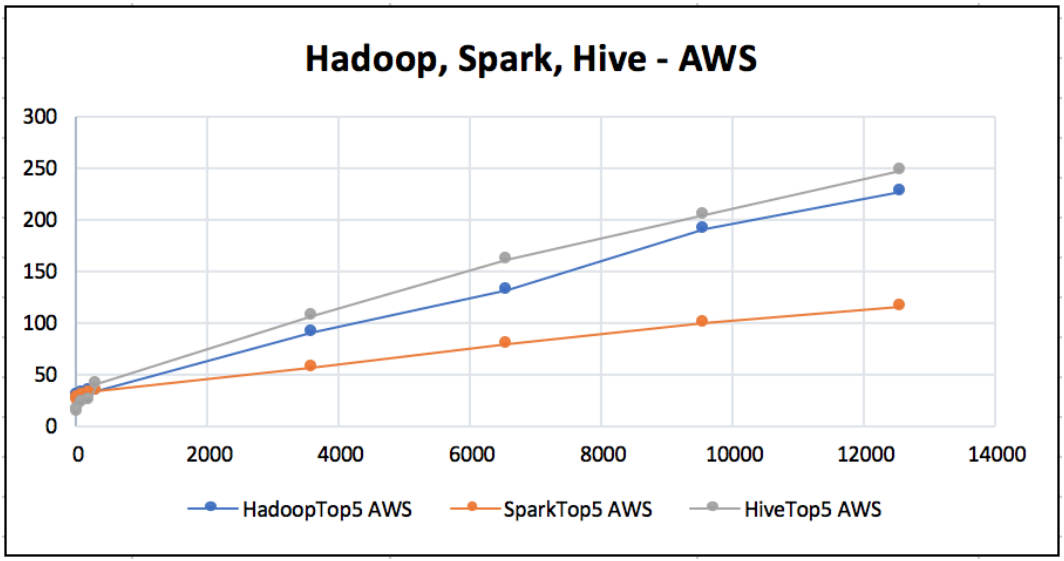

Figure 20: Performance Job 1

4.2 Job 2

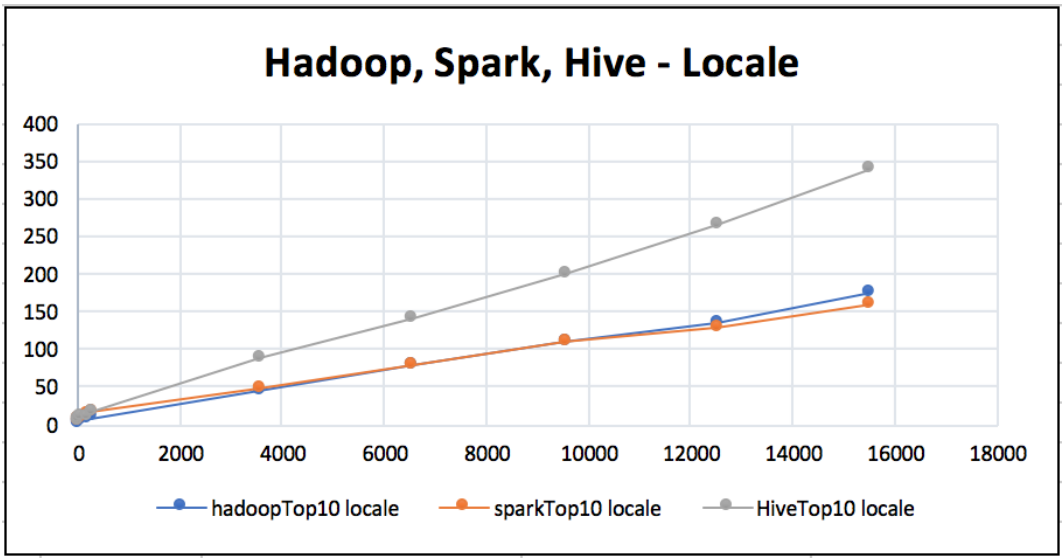

Figure 21: Performance Job 2

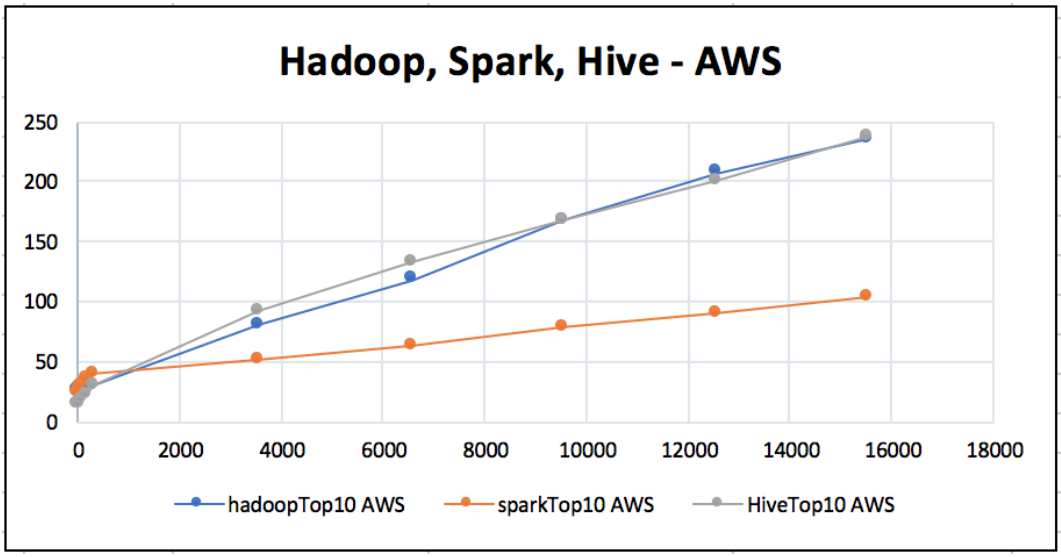

Figure 22: Performance Job 2

4.3 Job 3

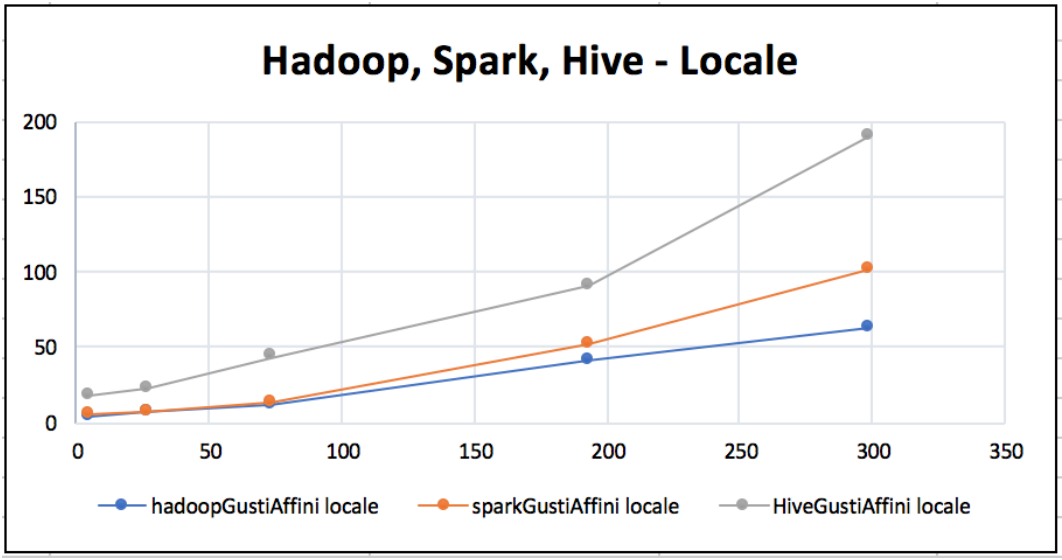

Figure 23: Performance Job 3

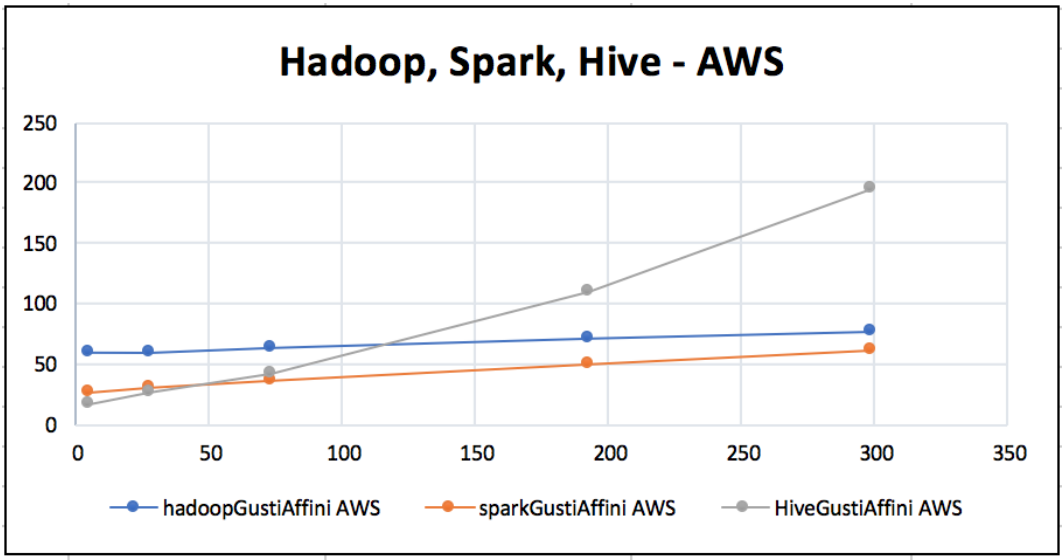

Figure 24: Performance Job 3