In [1]:
```
import pyspark as spark
```

In [2]:
```
sc = spark.SparkContext()
```

```
24/10/14 13:49:44 WARN Utils: Your hostname, HP-Victus resolves to a loopback address: 127.0.1.1;
using 10.255.255.254 instead (on interface lo)
24/10/14 13:49:44 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
24/10/14 13:49:45 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform...
using builtin-java classes where applicable
```

Word Count

In [3]:
```
with open('words.txt', 'w') as f:
    text = '''Install spark and pyspark (ubuntu only). Run a spark shell and test the installation.
            Run the wordcount program that you did using hadoop using pyspark. \
            Use the movielens dataset available in the LMS theory page and \
            try to find out for each movie, how are the ratings distributed.'''
    f.write(text)
```

The `counts` contains the text from `text_file` split into words and converted into a tuple of `(word, count)` format and then reduced to give the count of each word.

In [4]:
```
text_file = sc.textFile("words.txt")
counts = text_file.flatMap(lambda line: line.split(" ")).map(lambda word: (word, 1)).reduceByKey(la
output = counts.collect()
```

In [5]:
```
output
```

Out[5]:
```
[('Install', 1),
 ('(ubuntu', 1),
 ('shell', 1),
 ('test', 1),
 ('installation.', 1),
 ('', 36),
 ('using', 2),
 ('hadoop', 1),
 ('Use', 1),
 ('in', 1),
 ('page', 1),
 ('try', 1),
 ('out', 1),
 ('movie,', 1),
 ('are', 1),
 ('spark', 2),
 ('and', 3),
 ('pyspark', 1),
 ('only).', 1),
 ('Run', 2),
 ('a', 1),
 ('the', 5),
 ('wordcount', 1),
 ('program', 1),
 ('that', 1),
 ('you', 1),
 ('did', 1),
 ('pyspark.', 1),
 ('movielens', 1),
 ('dataset', 1),
 ('available', 1),
 ('LMS', 1),
 ('theory', 1),
 ('\\', 1),
 ('to', 1),
 ('find', 1),
 ('for', 1),
 ('each', 1),
 ('how', 1),
 ('ratings', 1),
 ('distributed.', 1)]
```

Movie Lens

The movies contains the MovieLens dataset stored in the RDD format with the columns corresponding to `UserID`, `MovieID`, `Rating`, and `TimeStamp`.

In [6]:
```
movies = sc.textFile("u.data")

movies.take(5)
```

Out[6]:
```
['196\t242\t3\t881250949',
 '186\t302\t3\t891717742',
 '22\t377\t1\t878887116',
 '244\t51\t2\t880606923',
 '166\t346\t1\t886397596']
```

The `movies_grp` contains the data split into lists of lists, grouped by `MovieID` and `Rating`, and the count of each group.

In [7]:
```
movies_grp = movies.map(lambda x: x.split('\t')).groupBy(lambda x: (x[1], x[2])).mapValues(len).col
movies_grp.sort()
```

In [8]:
```
movies_grp
```

Out[8]:
```
[(('1', '1'), 8),
 (('1', '2'), 27),
 (('1', '3'), 96),
 (('1', '4'), 202),
 (('1', '5'), 119),
 (('10', '1'), 2),
 (('10', '2'), 7),
 (('10', '3'), 21),
 (('10', '4'), 33),
 (('10', '5'), 26),
 (('100', '1'), 14),
 (('100', '2'), 18),
 (('100', '3'), 70),
 (('100', '4'), 179),
 (('100', '5'), 227),
 (('1000', '2'), 2),
 (('1000', '3'), 6),
 (('1000', '4'), 2),
 (('1001', '1'), 10),
```