

Names: Bob Skowron, Jason Walker

Keys: rskowron, jwalker

SVN: jwalker: https://svn.seas.wustl.edu/repositories/jwalker/cse427s_f17/

1. a. `mydata.map(lambda line: line.split(' ')).filter(lambda fields: "html" in fields[6]).keyBy(lambda fields: (fields[0] + "/" + fields[2])).keys().saveAsTextFile("loudacre-weblogs.html")`

```
hadoop fs -cat loudacre-weblogs.html/part-00000 | head -10
```

```
3.94.78.5/69827
19.38.140.62/21475
129.133.56.105/2489
217.150.149.167/4712
209.151.12.34/45922
184.97.84.245/144
233.60.251.2/33908
160.134.139.204/51340
19.209.18.222/13392
230.220.223.28/12643
```

- b. Total lines: 1,079,891; Number of lines with HTML requests: 474,360

Code:

Total Lines: `mydata.map(lambda line: line.split(' ')).keyBy(lambda fields: (fields[0] + "/" + fields[2])).count()`

Lines with HTML: `mydata.map(lambda line: line.split(' ')).filter(lambda fields: "html" in fields[6]).keyBy(lambda fields: (fields[0] + "/" + fields[2])).count()`

Names: Bob Skowron, Jason Walker

Keys: rskowron, jwalker

SVN: jwalker: https://svn.seas.wustl.edu/repositories/jwalker/cse427s_f17/

2. a. The keys are the paths to the files. The value is the entire contents of the file.

```
mydata.keys().take(2)
[u'hdfs://localhost:8020/loudacre/activations/2008-10.xml',
 u'hdfs://localhost:8020/loudacre/activations/2008-11.xml']
```

- b. flatMap()

```
import xml.etree.ElementTree as ElementTree
```

```
def getactivations(s):
    filetree = ElementTree.fromstring(s)
    return filetree.getiterator('activation')
```

```
xmldata = mydata.flatMap(lambda fields: getactivations(fields[1]))
```

- c.

```
def getmodel(activation):
    return activation.find('model').text
def getaccount(activation):
    return activation.find('account-number').text
```

```
xmldata.map(lambda activation: getaccount(activation) + ":" + getmodel(activation)).saveAsTextFile("/loudacre/account
models")
```

Names: Bob Skowron, Jason Walker

Keys: rskowron, jwalker

SVN: jwalker: https://svn.seas.wustl.edu/repositories/jwalker/cse427s_f17/

3.
 - a. Spark works in stages. Pipelining is a feature of Spark that when possible, Spark will optimize transformations not separated by shuffles by executing them in a single task. This allows it to skip adding another stage and it can execute those steps on a single cluster node. There are several benefits. First, no intermediate records or RDDs have to be stored. Second, it will only process the requisite transformations for the data required. We do not have to apply all transformations to all rows if they end up filtered later on.
 - b. Maps and filters can be pipelined together. Any transformation that does not require a shuffle of the data should be able to be pipelined.