Homework 3

Akshay Mishra 011476673

Q1. For *each* job:

At high level, describe the logic of your MapReduce job

- Input directory on the VM =
- Output directory on the VM =
- # of map tasks =
- # of reduce tasks =

Your screenshots (specified later) must match this info.

Answer:

Mapper: This class takes care of running mapper jobs where every line of input in all input files is split based on space character, which is in-line with the structure of the provided files. The mapper retrieves appropriate element, which belongs to the URL field as per the log format. For every URL, the mapper associates count of 1, and passes this to reducer as its input.

Reducer: Reducer basically sums up the count for every URL and stores it in the form of URL against its count. As an additional responsibility through its cleanup method, the reducer also sorts the results based on URL count using Java collections APIs. The sorting functionality could have been achieved using another MapReduce job, but choosing overriding cleanup function over another MapReduce job is merely a design choice.

- Input directory on the VM = (inside HDFS) /tmp/wordcount/in
- Output directory on the VM =(inside HDFS) /tmp/wordcount/out
- # of map tasks = 3
- # of reduce tasks = 1

Q2. Use the aforementioned example hit count and sorted output to illustrate step by step *how* your jobs perform the work correctly.

Answer:

Below is the Mapper class:

```
public class MapperAkshayMishra673 extends Mapper<Object, Text, Text, IntWritable> {

        private final static IntWritable ONE = new IntWritable(1);
        private Text word = new Text();

        public void map(Object key, Text value, Context context
                ) throws IOException, InterruptedException {
           String[] split = value.toString().split(" ");
           for(int i=0; i<split.length; i++){
              if (i==6)
                 word.set(split[i]);
                 context.write(word, ONE);
           }
        }
}
```

- Mapper class will take input from the log files provided as per the job configuration. As per job configuration, it'll read file from /tmp/wordcount/in dir in HDFS.
- The Mapper will split every line based on space character as below.
- For example,
  198.104.162.38 - - [01/Jul/1995:23:59:49 -0400] "GET /images/NASA-logosmall.gif HTTP/1.0" 200 786 ➔

  ["198.104.162.38" "-" "-" "[01/Jul/1995:23:59:49 -0400]" "GET" "/images/NASA-logosmall.gif" "HTTP/1.0" "200" "786"]
- The mapper will associate count of 1 with every input and will simply ignore rest of the fields. So, it will throw output in the form of key, value such that key is URL in the line and value is count '1'. The input format is Object & Text, Output format is text, IntWritable.
- IntWritable is a class defined in Hadoop Library.

```
public class ReducerAkshayMishra673 extends Reducer<Text,IntWritable,Text,IntWritable> {

   //private IntWritable result = new IntWritable();
   TreeMap<Text,IntWritable> result = new TreeMap<Text, IntWritable>();

   public void reduce(Text key, Iterable<IntWritable> values,  Context context) throws
IOException, InterruptedException {

         int sum = 0;
         for (IntWritable val : values) {
            sum += val.get();
         }
         result.put(new Text(key),new IntWritable(sum));
      }

   @Override
   protected void cleanup(Context context) throws IOException, InterruptedException {
      Set<Map.Entry<Text, IntWritable>> set = result.entrySet();

      List<Map.Entry<Text, IntWritable>> list = new ArrayList<Map.Entry<Text,IntWritable>>(set);

      Collections.sort(list,
                                    new Comparator<Map.Entry<Text, IntWritable>>() {
                                    public int compare( Map.Entry<Text, IntWritable> o1,
                                    Map.Entry<Text,IntWritable> o2 ) {
                              return ((o2.getValue()).compareTo( o1.getValue()) * -1);
                           }
                        });

                        for(Map.Entry<Text,IntWritable> entry:list){
                           context.write(entry.getKey(),entry.getValue());
                        }
            }
}
```

- The reducer class basically performs two functions:
    - Reducing the input provided by Mapper
    - Sorting it based on value or the URL count in the context of above example

- Reducing: Reduction is performed by adding the count of 1s to get a total count for every URL and then storing them inside a map. It is stored in order to further pass it to the cleanup method, wherein we have defined the logic of sorting.
- Sorting: The sorting logic involves storing the map.entrySet() into a set from which we are going to create a list. The list, which is a part of powerful Collection framework in Java can be sorted using Collection APIs.

- We have to define a new Comparator for this to work as Comparing Map<Map.Entry<Text, IntWritable>> is not a built-in operation in Java.

- Once sorting is done, the Reducer gives its output by writing to context, which is the final output of our reducer.
- It is written to HDFS in /tmp/wordcount/out dir as specified by the job configuration.

Q3. Enclose screenshot s for *each* of the following steps for *each* job:

The JAR file is stored in the /tmp directory, in order to run the program, please run below command:

```
hadoop jar /tmp/hw3AkshayMishra673.jar com.cmpe282.hw3AkshayMishra673.WCAkshayMishra673
/tmp/wordcount/in /tmp/wordcount/out
```

Ensure that the /tmp/wordcount/in is present and contains all weblog files.

And /tmp/wordcount/out is not present, it'll be created by the program

• Before job execution, show input directory on VM o hadoop fs - ls <inputDir>

```
[cloudera@quickstart ~]$ hadoop fs -ls /tmp/wordcount
Found 1 items
drwxr-xr-x   - cloudera supergroup          0 2017-11-13 11:59 /tmp/wordcount/in
[cloudera@quickstart ~]$
```

• During execution, capture output from "hadoop jar ... ". In particular, highlight the # of map tasks and # of reduce tasks.

```
~/workspace/hw3cmpe282AkshayMishra673 — builder@rh7v-intel64-90-java-stress-1: ~ — -bash    ~/workspace/cmpe282AkshayMishra — cloudera@quickstart:~ — ssh cloudera@130.65.159.146
[cloudera@quickstart ~]$ hadoop jar /tmp/hw3AkshayMishra673.jar com.cmpe282.hw3AkshayMishra673.WCAkshayMishra673 /tmp/wordcount/in /tmp/wordcount/out
17/11/13 12:24:47 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
17/11/13 12:24:48 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your applicat
ion with ToolRunner to remedy this.
17/11/13 12:24:48 INFO input.FileInputFormat: Total input paths to process : 3
17/11/13 12:24:48 INFO mapreduce.JobSubmitter: number of splits:3
17/11/13 12:24:48 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1510600702085_0003
17/11/13 12:24:49 INFO impl.YarnClientImpl: Submitted application application_1510600702085_0003
17/11/13 12:24:49 INFO mapreduce.Job: The url to track the job: http://quickstart.cloudera:8088/proxy/application_1510600702085_0003/
17/11/13 12:24:49 INFO mapreduce.Job: Running job: job_1510600702085_0003
17/11/13 12:24:56 INFO mapreduce.Job: Job job_1510600702085_0003 running in uber mode : false
17/11/13 12:24:56 INFO mapreduce.Job:  map 0% reduce 0%
17/11/13 12:25:11 INFO mapreduce.Job:  map 33% reduce 0%
17/11/13 12:25:12 INFO mapreduce.Job:  map 67% reduce 0%
17/11/13 12:25:14 INFO mapreduce.Job:  map 100% reduce 0%
17/11/13 12:25:27 INFO mapreduce.Job:  map 100% reduce 100%
17/11/13 12:25:27 INFO mapreduce.Job: Job job_1510600702085_0003 completed successfully
17/11/13 12:25:27 INFO mapreduce.Job: Counters: 50
        File System Counters
                FILE: Number of bytes read=84119100
                FILE: Number of bytes written=168722449
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=23613580
                HDFS: Number of bytes written=239660
                HDFS: Number of read operations=12
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Killed map tasks=1
                Launched map tasks=3
                Launched reduce tasks=1
                Data-local map tasks=3
                Total time spent by all maps in occupied slots (ms)=40946
                Total time spent by all reduces in occupied slots (ms)=13016
                Total time spent by all map tasks (ms)=40946
                Total time spent by all reduce tasks (ms)=13016
                Total vcore-seconds taken by all map tasks=40946
                Total vcore-seconds taken by all reduce tasks=13016
                Total megabyte-seconds taken by all map tasks=41928704
                Total megabyte-seconds taken by all reduce tasks=13328384
        Map-Reduce Framework
                Map input records=214563
                Map output records=2145675
                Map output bytes=79827744
                Map output materialized bytes=84119112
                Input split bytes=399
```

Map Tasks = 3,
Reduce Tasks=1

• After job execution, show output directory on VM o hadoop fs - ls <outputDir>

```
[cloudera@quickstart ~]$ hadoop fs -ls /tmp/wordcount/
Found 2 items
drwxr-xr-x   - cloudera supergroup          0 2017-11-13 11:59 /tmp/wordcount/in
drwxr-xr-x   - cloudera supergroup          0 2017-11-13 12:47 /tmp/wordcount/out
[cloudera@quickstart ~]$ hadoop fs -ls /tmp/wordcount/out
Found 2 items
-rw-r--r--   1 cloudera supergroup          0 2017-11-13 12:47 /tmp/wordcount/out/_SUCCESS
-rw-r--r--   1 cloudera supergroup     239660 2017-11-13 12:47 /tmp/wordcount/out/part-r-00000
[cloudera@quickstart ~]$
```

• After execution, for *each* part - r - * output file, show first few lines with head and last few lines with tail:

o hadoop fs - cat <output File> | head

```
[cloudera@quickstart ~]$ hadoop fs -cat /tmp/wordcount/out/part-r-00000 | head
/%7Eadverts/ibm/ad1/banner.gif  1
/.      1
/./     1
/.pub.win3.winvn        1
//elv/elvhead2.gif      1
//elv/elvhead3.gif      1
//elv/endball.gif       1
//elv/vidpicp.htm       1
//elv/whnew.htm 1
//history/apollo/apollo-13/apollo-13html        1
cat: Unable to write to output stream.
```

o hadoop fs - cat <outputFile> | tail

```
[cloudera@quickstart ~]$ hadoop fs -cat /tmp/wordcount/out/part-r-00000 | tail
/shuttle/missions/sts-71/images/images.html     4645
/images/WORLD-logosmall.gif     4956
/shuttle/missions/sts-71/sts-71-patch-small.gif 5003
/images/USA-logosmall.gif       5075
/images/MOSAIC-logosmall.gif    5099
/images/ksclogo-medium.gif      5146
/shuttle/countdown/     8470
/shuttle/countdown/count.gif    8646
/images/KSC-logosmall.gif       11262
/images/NASA-logosmall.gif      12970
```

Q 4. Comment on the performance and scalability of the 2nd MapReduce job. Discuss if there is any way to improve its performance and scalability.

Answer:

Since I am using only one MapReduce Job, instead of second separate MapReduce Job, I'll discuss the scalability and performance of the sorting function in my only MapReduce job.

MapReduce Aspect:
If we were using second MapReduce job, then it is worth noting that it is generally a better approach to keep the data in sorted state using let's say BigTable concepts, etc. In short, it is more efficient to sort the data once during insertion that to sort them at the time of each MapReduce query. Use of combiner and more no of reducers should be able to perform data parallel reducing response time.

Sorting Aspect:
I am using Collections.sort() API from powerful Collection framework in Java, which is basically using list.sort() method. The current solution indirectly uses MergeSort through Collections.sort() method. MergeSort is fast, stable and guarantees O(nlogn) efficiency, although it takes extra space of o(n). We could implement our own QuickSort to sort based on the values, which would eliminate auxiliary space requirement.

Scalability of the current sorting implementation can be improved by adding more powerful machines to the hadoop cluster if we were to run the program for larger files. The faster processer with higher RAM would help to scale the application.

EXTRA CREDIT

"wc.py"

```
import sys

from pyspark import SparkContext, SparkConf

if __name__ == "__main__":

# create Spark context with Spark configuration
  conf = SparkConf().setAppName("Spark Count")
  sc = SparkContext(conf=conf)


#to count the words in a file hdfs:/// of file:/// or localfile "./samplefile.txt"
rdd=sc.textFile("hdfs://localhost:8020/tmp/wordcount/in/")#.map(lambda x: x[0]).collect()
#or you can initialize with your list
#v1='Hi hi hi bye bye bye word count'
#rdd=sc.parallelize([v1])

#print rdd.take(10)


wordcounts=rdd.map(lambda l: l.split()[6]) \
          .map(lambda w:(w,1)) \
          .reduceByKey(lambda a,b:a+b) \
          .map(lambda (a,b):(b,a)) \
          .sortByKey(ascending=True)

output = wordcounts.collect()

for (count,word) in output:
    print("%s: %i" % (word,count))
```

1. Save above code in current dir as "wc.py"
2. Command to run the apache spark job:

```
spark-submit --master yarn --deploy-mode client --executor-memory 1g --name wordcount --
conf "spark.app.id=wordcount" wc.py
```

Explanation:

Below are the sequence of actions happening in the python file.

1. Initialize a SparkContext
2. Read textfiles from specified path into the RDD
3. Split them and take only URL from the log lines
4. Associate count of 1 with every URL
5. Reduce based on key i.e. URL here
6. Swap key,value pair to value,key to facilitate sorting using SortByKey function
7. Collect the output
8. Print the output

Pre-requisites:

1. Install JDK, Scala to set up the Apache Spark environment.
2. Install spark, add it to the PATH

Create below dir structure in hdfs /tmp dir.

Copy all weblog files to form a dir structure as below

/tmp
    /wordcount
        /in
            weblog-1995-7-1.txt
            weblog-1995-7-2.txt
            weblog-1995-7-3.txt

Screenshots:

Program terminal output:

```
17/11/14 17:09:33 INFO mapred.FileInputFormat: Total input paths to process : 3
17/11/14 17:09:33 INFO spark.SparkContext: Starting job: sortByKey at /home/cloudera/wc.py:25
17/11/14 17:09:33 INFO scheduler.DAGScheduler: Registering RDD 3 (reduceByKey at /home/cloudera/wc.py:23)
17/11/14 17:09:33 INFO scheduler.DAGScheduler: Got job 0 (sortByKey at /home/cloudera/wc.py:25) with 3 output partitions
17/11/14 17:09:33 INFO scheduler.DAGScheduler: Final stage: ResultStage 1 (sortByKey at /home/cloudera/wc.py:25)
17/11/14 17:09:33 INFO scheduler.DAGScheduler: Parents of final stage: List(ShuffleMapStage 0)
17/11/14 17:09:33 INFO scheduler.DAGScheduler: Missing parents: List(ShuffleMapStage 0)
17/11/14 17:09:34 INFO scheduler.DAGScheduler: Submitting ShuffleMapStage 0 (PairwiseRDD[3] at reduceByKey at /home/cloudera/wc.py:23), which has no missing parents
17/11/14 17:09:34 INFO storage.MemoryStore: Block broadcast_1 stored as values in memory (estimated size 8.1 KB, free 530.1 MB)
17/11/14 17:09:34 INFO storage.MemoryStore: Block broadcast_1_piece0 stored as bytes in memory (estimated size 5.1 KB, free 530.1 MB)
17/11/14 17:09:34 INFO storage.BlockManagerInfo: Added broadcast_1_piece0 in memory on 130.65.159.146:57975 (size: 5.1 KB, free: 530.3 MB)
17/11/14 17:09:34 INFO spark.SparkContext: Created broadcast 1 from broadcast at DAGScheduler.scala:1006
17/11/14 17:09:34 INFO scheduler.DAGScheduler: Submitting 3 missing tasks from ShuffleMapStage 0 (PairwiseRDD[3] at reduceByKey at /home/cloudera/wc.py:23)
17/11/14 17:09:34 INFO cluster.YarnScheduler: Adding task set 0.0 with 3 tasks
17/11/14 17:09:34 INFO scheduler.TaskSetManager: Starting task 0.0 in stage 0.0 (TID 0, 130.65.159.146, executor 1, partition 0, RACK_LOCAL, 2149 bytes)
17/11/14 17:09:34 INFO scheduler.TaskSetManager: Starting task 1.0 in stage 0.0 (TID 1, 130.65.159.146, executor 2, partition 1, RACK_LOCAL, 2149 bytes)
17/11/14 17:09:34 INFO storage.BlockManagerInfo: Added broadcast_1_piece0 in memory on 130.65.159.146:32827 (size: 5.1 KB, free: 530.3 MB)
17/11/14 17:09:34 INFO storage.BlockManagerInfo: Added broadcast_1_piece0 in memory on 130.65.159.146:52006 (size: 5.1 KB, free: 530.3 MB)
17/11/14 17:09:34 INFO storage.BlockManagerInfo: Added broadcast_0_piece0 in memory on 130.65.159.146:52006 (size: 23.2 KB, free: 530.3 MB)
17/11/14 17:09:34 INFO storage.BlockManagerInfo: Added broadcast_0_piece0 in memory on 130.65.159.146:32827 (size: 23.2 KB, free: 530.3 MB)
17/11/14 17:09:40 INFO scheduler.TaskSetManager: Starting task 2.0 in stage 0.0 (TID 2, 130.65.159.146, executor 1, partition 2, RACK_LOCAL, 2149 bytes)
17/11/14 17:09:40 INFO scheduler.TaskSetManager: Finished task 0.0 in stage 0.0 (TID 0) in 6358 ms on 130.65.159.146 (executor 1) (1/3)
17/11/14 17:09:40 INFO scheduler.TaskSetManager: Finished task 1.0 in stage 0.0 (TID 1) in 6341 ms on 130.65.159.146 (executor 2) (2/3)
17/11/14 17:09:40 INFO scheduler.TaskSetManager: Finished task 2.0 in stage 0.0 (TID 2) in 625 ms on 130.65.159.146 (executor 1) (3/3)
17/11/14 17:09:40 INFO scheduler.DAGScheduler: ShuffleMapStage 0 (reduceByKey at /home/cloudera/wc.py:23) finished in 6.857 s
17/11/14 17:09:40 INFO scheduler.DAGScheduler: looking for newly runnable stages
17/11/14 17:09:40 INFO scheduler.DAGScheduler: running: Set()
17/11/14 17:09:40 INFO scheduler.DAGScheduler: waiting: Set(ResultStage 1)
17/11/14 17:09:40 INFO scheduler.DAGScheduler: failed: Set()
17/11/14 17:09:40 INFO scheduler.DAGScheduler: Submitting ResultStage 1 (PythonRDD[6] at sortByKey at /home/cloudera/wc.py:25), which has no missing parents
17/11/14 17:09:40 INFO cluster.YarnScheduler: Removed TaskSet 0.0, whose tasks have all completed, from pool
17/11/14 17:09:40 INFO storage.MemoryStore: Block broadcast_2 stored as values in memory (estimated size 6.4 KB, free 530.0 MB)
17/11/14 17:09:41 INFO storage.MemoryStore: Block broadcast_2_piece0 stored as bytes in memory (estimated size 4.0 KB, free 530.0 MB)
17/11/14 17:09:41 INFO storage.BlockManagerInfo: Added broadcast_2_piece0 in memory on 130.65.159.146:57975 (size: 4.0 KB, free: 530.2 MB)
17/11/14 17:09:41 INFO spark.SparkContext: Created broadcast 2 from broadcast at DAGScheduler.scala:1006
17/11/14 17:09:41 INFO scheduler.DAGScheduler: Submitting 3 missing tasks from ResultStage 1 (PythonRDD[6] at sortByKey at /home/cloudera/wc.py:25)
17/11/14 17:09:41 INFO cluster.YarnScheduler: Adding task set 1.0 with 3 tasks
17/11/14 17:09:41 INFO scheduler.TaskSetManager: Starting task 0.0 in stage 1.0 (TID 3, 130.65.159.146, executor 1, partition 0, NODE_LOCAL, 1894 bytes)
17/11/14 17:09:41 INFO scheduler.TaskSetManager: Starting task 1.0 in stage 1.0 (TID 4, 130.65.159.146, executor 2, partition 1, NODE_LOCAL, 1894 bytes)
17/11/14 17:09:41 INFO storage.BlockManagerInfo: Added broadcast_2_piece0 in memory on 130.65.159.146:32827 (size: 4.0 KB, free: 530.2 MB)
17/11/14 17:09:41 INFO storage.BlockManagerInfo: Added broadcast_2_piece0 in memory on 130.65.159.146:52006 (size: 4.0 KB, free: 530.2 MB)
17/11/14 17:09:41 INFO spark.MapOutputTrackerMasterEndpoint: Asked to send map output locations for shuffle 0 to 130.65.159.146:43544
17/11/14 17:09:41 INFO spark.MapOutputTrackerMasterEndpoint: Asked to send map output locations for shuffle 0 to 130.65.159.146:43549
17/11/14 17:09:41 INFO spark.MapOutputTrackerMaster: Size of output statuses for shuffle 0 is 167 bytes
17/11/14 17:09:41 INFO scheduler.TaskSetManager: Starting task 2.0 in stage 1.0 (TID 5, 130.65.159.146, executor 1, partition 2, NODE_LOCAL, 1894 bytes)
17/11/14 17:09:41 INFO scheduler.TaskSetManager: Finished task 0.0 in stage 1.0 (TID 3) in 909 ms on 130.65.159.146 (executor 1) (1/3)
17/11/14 17:09:41 INFO scheduler.TaskSetManager: Finished task 1.0 in stage 1.0 (TID 4) in 913 ms on 130.65.159.146 (executor 2) (2/3)
17/11/14 17:09:41 INFO scheduler.TaskSetManager: Finished task 2.0 in stage 1.0 (TID 5) in 44 ms on 130.65.159.146 (executor 1) (3/3)
17/11/14 17:09:41 INFO cluster.YarnScheduler: Removed TaskSet 1.0, whose tasks have all completed, from pool
17/11/14 17:09:41 INFO scheduler.DAGScheduler: ResultStage 1 (sortByKey at /home/cloudera/wc.py:25) finished in 0.945 s
17/11/14 17:09:41 INFO scheduler.DAGScheduler: Job 0 finished: sortByKey at /home/cloudera/wc.py:25, took 8.032896 s
17/11/14 17:09:42 INFO spark.SparkContext: Starting job: sortByKey at /home/cloudera/wc.py:25
17/11/14 17:09:42 INFO scheduler.DAGScheduler: Got job 1 (sortByKey at /home/cloudera/wc.py:25) with 3 output partitions
17/11/14 17:09:42 INFO scheduler.DAGScheduler: Final stage: ResultStage 3 (sortByKey at /home/cloudera/wc.py:25)
17/11/14 17:09:42 INFO scheduler.DAGScheduler: Parents of final stage: List(ShuffleMapStage 2)
17/11/14 17:09:42 INFO scheduler.DAGScheduler: Missing parents: List()
17/11/14 17:09:42 INFO scheduler.DAGScheduler: Submitting ResultStage 3 (PythonRDD[7] at sortByKey at /home/cloudera/wc.py:25), which has no missing parents
17/11/14 17:09:42 INFO storage.MemoryStore: Block broadcast_3 stored as values in memory (estimated size 6.3 KB, free 530.0 MB)
17/11/14 17:09:42 INFO storage.MemoryStore: Block broadcast_3_piece0 stored as bytes in memory (estimated size 4.0 KB, free 530.0 MB)
17/11/14 17:09:42 INFO storage.BlockManagerInfo: Added broadcast_3_piece0 in memory on 130.65.159.146:57975 (size: 4.0 KB, free: 530.2 MB)
```

Sample initial lines of output

```
/shuttle/technology/sts-newsref/106_operatns.txt: 1
/htbin/imagemap/paylproc?160,286: 1
/software/winvn/vinvn.html: 1
/shuttle/countdown/video/livevideo.jpeg": 1
/news/sci.space.news/archive/sci-space-news-25-jan-1995-00.txt: 1
/cgi-bin/imagemap/countdown?320,277: 1
/shuttle/technology/sts-newsref/spacelab.html": 1
/htbin/wais.pl?GLO-2: 1
/shuttle/missions/41-g/images/84HC520.GIF: 1
/htbin/wais.pl?cusseeme: 1
/cgi-bin/imagemap/countdown?239,156: 1
/history/apollo/as-202/sounds/: 1
/cgi-bin/imagemap/countdown?104,156: 1
/shuttle/missions/status/r89-183: 1
/cgi-bin/imagemap/countdown?89,246: 1
/shuttle/missions/sts-71/sts-71-press-kit.txt": 1
/cgi-bin/imagemap/countdown?269,195: 1
/cgi-bin/imagemap/fr?187,134: 1
/htbin/wais.pl?DMOS: 1
/cgi-bin/imagemap/countdown?95,105: 1
/htbin/wais.pl?mir+and+layout: 1
/history/apollo/apollo-13/apollo-13-info.hmtl: 1
/biomed/threat/gif/manateefin.gif: 1
/cgi-bin/imagemap/countdown?333,174: 1
/cgi-bin/imagemap/countdown?88,168: 1
/htbin/wais.pl?waste: 1
```

Sample terminal lines of output:

```
/shuttle/missions/sts-71/movies/movies.html: 1221
/history/apollo/apollo-13/apollo-13-patch-small.gif: 1256
/history/apollo/images/apollo-logo.gif: 1490
/history/apollo/apollo-13/apollo-13.html: 1535
/history/apollo/apollo.html: 1545
/history/apollo/images/footprint-small.gif: 1627
/history/apollo/images/footprint-logo.gif: 1678
/icons/blank.xbm: 1752
/icons/menu.xbm: 1753
/images/launchmedium.gif: 2236
/shuttle/missions/missions.html: 2757
/shuttle/countdown/liftoff.html: 3173
/: 3247
/ksc.html: 3379
/images/ksclogosmall.gif: 3546
/shuttle/countdown/video/livevideo.gif: 3590
/shuttle/missions/sts-71/mission-sts-71.html: 4287
/images/launch-logo.gif: 4514
/shuttle/missions/sts-71/images/images.html: 4645
/images/WORLD-logosmall.gif: 4956
/shuttle/missions/sts-71/sts-71-patch-small.gif: 5003
/images/USA-logosmall.gif: 5075
/images/MOSAIC-logosmall.gif: 5099
/images/ksclogo-medium.gif: 5146
/shuttle/countdown/: 8470
/shuttle/countdown/count.gif: 8646
/images/KSC-logosmall.gif: 11262
/images/NASA-logosmall.gif: 12970
17/11/14 17:09:42 INFO spark.SparkContext: Invoking stop() from shutdown hook
17/11/14 17:09:42 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/metrics/json,null}
17/11/14 17:09:42 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/stages/stage/kill,null}
17/11/14 17:09:42 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/api,null}
17/11/14 17:09:42 INFO handler.ContextHandler: stopped o.s.j.s.ServletContextHandler{/,null}
```