HW#2

UFID - 16446159

Algorithm

Here is the map-reduce input/output used in page rank algorithm(RankMapReducer.java).

- From map, I am emitting link as key and title of the page,pagerank and outlink as value. Now reducer will get the output from all the maps where key was this link. Value in the reducer input will be combined value of, all the nodes pointing to this node and their respective page rank. In reducer we will use the following formula to calculate page rank –

  Rank contributed by a individual node –> rank(node)/degree(node), then we will sum up all of these calculated ranks to get something called say – sum.

  Now, use below formula –

  sum = lambda * sum + 1 - lambda;

  to get final rank.

Map:

- Input:
  - key: index.html
  - value: <pagerank> 1.html 2.html...
  - Output for each outlink:
  - key: "1.html"
  - value: "index.html <pagerank> <number of outlinks>"

Reduce

- Input:
  - Key: "1.html"
  - Value: "index.html 0.5 23"
    Value: "2.html 2.4 2"
    Value: ...
- Output:
  - Key: "1.html"
  - Value: "<new pagerank> index.html 2.html..."

After that, the graph information is output using another simple MapReduce.

We also retain original graph structure to be used in next iteration.

- To get top 10 Nodes with high page rank, I have used another job SortMapReducer.java. In mapper of SortMapReduce, I am emitting (100-pagerank) as key and node as value. In reducer, we will sorted key-value pair sorted based on keys. Then we will emit (100-key) as key and node as value. We will emit key-value pair, 10 times to get top 10 page ranks. Below is graphical represention of map-reduce function of Sorter.

Map:

- Input:
  - Key: “index.html”
  - Value: “<pagerank> <outlinks>”
- Output:
  - Key: “<pagerank>”
  - Value: “index.html”

- 3rd job in program is to generate additional graph information(AdditionalPropertyMapperReducer.java). Input to this program would be the original graph structure which we retained. Now in mapper of this job, we will emit “addinfo” as key and their degree as value. In reducer input we will get degree for all the nodes in one reduce call. Then we will do simple operations to get all the additional info.

Convergence criteria-

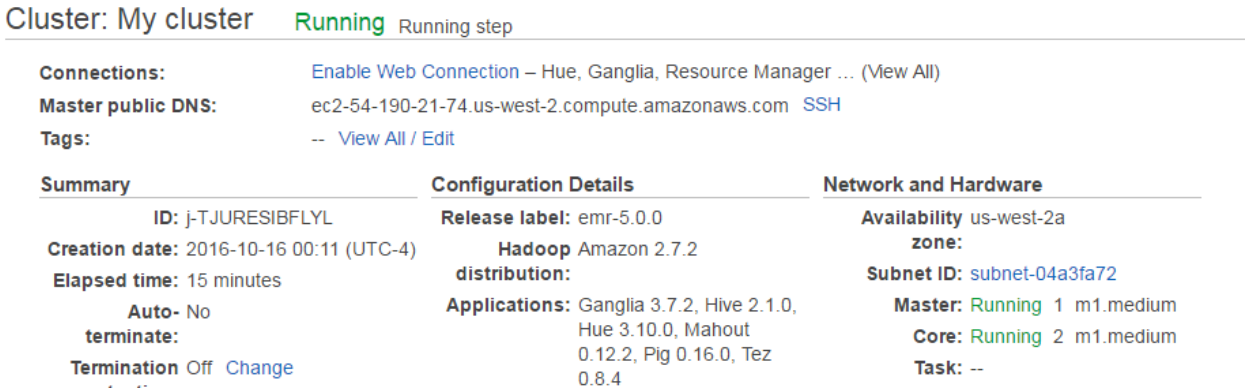
- If top 10 ranks don’t change for 3 iterations.
- If page rank remain constant for 3 consecutive iterations.

of iteration to converge

- Small – 13
- Medium – 11
- Large – 16

AWS EMR cluster run

Created EMR cluster with 1, m1.medium master instance and , m1.medium slave instance.



AWS s3 bucket-

All Buckets / cloudhw21

Name	Storage Class	Size
cloudhw.jar	Standard	34.7 MB
prog2-sample-large.txt	Standard	193.7 KB
prog2-sample-medium.txt	Standard	28.9 KB
prog2-sample-small.txt	Standard	11.1 KB

Adding steps –

For small file –

s-1KONPFJ43X2XP Custom JAR Running 2016-10-16 00:26 (UTC-4) 1 minute View logs

JAR location: s3://cloudhw21/cloudhw.jar
Main class: None
Arguments: s3://cloudhw21/prog2-sample-small.txt s3://cloudhw21/output-small 13
Action on failure: Continue

For medium file –

s-1KUBQUDGZ8ZMZ Custom JAR Completed 2016-10-16 00:45 (UTC-4) 15 minutes

JAR location: s3://cloudhw21/cloudhw.jar
Main class: None
Arguments: s3://cloudhw21/prog2-sample-medium.txt s3://cloudhw21/output-medium 11
Action on failure: Continue

For large file –

s-F3CW9MMZO5NJ Custom JAR Completed 2016-10-16 01:01 (UTC-4) 21 minutes

JAR location: s3://cloudhw21/cloudhw.jar
Main class: None
Arguments: s3://cloudhw21/prog2-sample-large.txt s3://cloudhw21/output-large 16
Action on failure: Continue

Results –

AWS results link –

Generated folders -

All Buckets / cloudhw21

Name	Storage Class	Size
cloudhw.jar	Standard	34.7 MB
output-large	--	--
output-largeadditionalInfo	--	--
output-medium	--	--
output-mediumadditionalInfo	--	--
output-small	--	--
output-smalladditionalInfo	--	--
prog2-sample-large.txt	Standard	193.7 KB
prog2-sample-medium.txt	Standard	28.9 KB
prog2-sample-small.txt	Standard	11.1 KB

Generated output files –

For small file –

https://s3-us-west-2.amazonaws.com/cloudhw21/output-small/part-00000

https://s3-us-west-2.amazonaws.com/cloudhw21/output-small/part-00001

https://s3-us-west-2.amazonaws.com/cloudhw21/output-small/part-00002

Additional info -

https://s3-us-west-2.amazonaws.com/cloudhw21/output-smalladditionalInfo/part-00001

For medium file –

https://s3-us-west-2.amazonaws.com/cloudhw21/output-medium/part-00000

https://s3-us-west-2.amazonaws.com/cloudhw21/output-medium/part-00001

https://s3-us-west-2.amazonaws.com/cloudhw21/output-medium/part-00002

Additional Info -

https://s3-us-west-2.amazonaws.com/cloudhw21/output-mediumadditionalInfo/part-00001

For large file-

https://s3-us-west-2.amazonaws.com/cloudhw21/output-large/part-00000

https://s3-us-west-2.amazonaws.com/cloudhw21/output-large/part-00001

https://s3-us-west-2.amazonaws.com/cloudhw21/output-large/part-00002

Additional info –

https://s3-us-west-2.amazonaws.com/cloudhw21/output-largeadditionalInfo/part-00001

Google cloud run –

Cluster creation -

cluster-1

Overview Jobs VM Instances Configuration

Edit

Name	cluster-1
Zone	us-east1-c
Master node	
Machine type	n1-standard-1 (1 vCPU, 3.75 GB memory)
Primary disk size	500 GB
Worker nodes	2
Machine type	n1-standard-1 (1 vCPU, 3.75 GB memory)
Primary disk size	500 GB
Local SSDs	0
Preemptible worker nodes	0
Cloud Storage staging bucket	dataproc-5e5ada84-9369-4c09-96c8-b3c430eceb02-us
Network	default
Image version	1.1.7
Created	15 Oct 2016, 23:45:16

Google cloud bucket –

Buckets / cloudhwbucket / input

Name	Size	Type	Last modified
cloudhw.jar	34.79 MB	binary/octet-stream	16/10/2016, 00:53
prog2-sample-large.txt	193.79 KB	text/plain	16/10/2016, 00:49
prog2-sample-medium.txt	28.91 KB	text/plain	16/10/2016, 00:49
prog2-sample-small.txt	11.14 KB	text/plain	15/10/2016, 20:23

Adding jobs –

For small files -

7636e524-32c6-47ad-8e06-237870d4a658

Start time: 16 Oct 2016, 00:54:19 Elapsed time: 45 min 50 sec Status: Succeeded

Output Configuration

Cluster	cluster-1
Job type	Hadoop
Jar files	
Main class or jar	gs://cloudhwbucket/input/cloudhw.jar
Arguments	gs://cloudhwbucket/input/prog2-sample-small.txt gs://cloudhwbucket/output-small 13

Equivalent REST

For medium file –

21d3b873-72e1-4057-8428-5ec0214c56df

Start time: 16 Oct 2016, 11:35:14 Elapsed time: 37 min 24 sec Status: Succeeded

Output Configuration

Cluster	cluster-1
Job type	Hadoop
Jar files	
Main class or jar	gs://cloudhwbucket/input/cloudhw.jar
Arguments	gs://cloudhwbucket/input/prog2-sample-medium.txt gs://cloudhwbucket/output-medium 11

Equivalent REST

For large file –

07c56cae-8c59-414d-84c7-3dc9d057f25b

Start time: 16 Oct 2016, 12:04:34 Elapsed time: 55 min 11 sec Status: Succeeded

Output Configuration

Cluster	cluster-2
Job type	Hadoop
Jar files	
Main class or jar	gs://cloudhwbucket/input/cloudhw.jar
Arguments	gs://cloudhwbucket/input/prog2-sample-large.txt gs://cloudhwbucket/output-large 16

Equivalent REST

<u>Results on google cloud –</u>

Buckets / cloudhwbucket

Name
input/
output-large/
output-largeadditionalInfo/
output-medium/
output-mediumadditionalInfo/
output-small/
output-smalladditionalInfo/

Small file –

https://storage.googleapis.com/cloudhwbucket/output-small/part-00000

https://storage.googleapis.com/cloudhwbucket/output-small/part-00001

https://storage.googleapis.com/cloudhwbucket/output-small/part-00002

https://storage.googleapis.com/cloudhwbucket/output-small/part-00003

https://storage.googleapis.com/cloudhwbucket/output-small/part-00004

https://storage.googleapis.com/cloudhwbucket/output-small/part-00005

https://storage.googleapis.com/cloudhwbucket/output-small/part-00006

https://storage.googleapis.com/cloudhwbucket/output-smalladditionalInfo/part-00006

Medium file –

https://storage.googleapis.com/cloudhwbucket/output-medium/part-00000

https://storage.googleapis.com/cloudhwbucket/output-medium/part-00001

https://storage.googleapis.com/cloudhwbucket/output-medium/part-00002

https://storage.googleapis.com/cloudhwbucket/output-medium/part-00003

https://storage.googleapis.com/cloudhwbucket/output-medium/part-00004

https://storage.googleapis.com/cloudhwbucket/output-medium/part-00005

https://storage.googleapis.com/cloudhwbucket/output-medium/part-00006

https://storage.googleapis.com/cloudhwbucket/output-medium/part-00007

https://storage.googleapis.com/cloudhwbucket/output-mediumadditionalInfo/part-00006

Large file –

https://storage.googleapis.com/cloudhwbucket/output-large/part-00000

https://storage.googleapis.com/cloudhwbucket/output-large/part-00001

https://storage.googleapis.com/cloudhwbucket/output-large/part-00002

https://storage.googleapis.com/cloudhwbucket/output-large/part-00003

https://storage.googleapis.com/cloudhwbucket/output-large/part-00004

https://storage.googleapis.com/cloudhwbucket/output-large/part-00005

https://storage.googleapis.com/cloudhwbucket/output-large/part-00006

https://storage.googleapis.com/cloudhwbucket/output-large/part-00007

https://storage.googleapis.com/cloudhwbucket/output-largeadditionalInfo/part-00006

Google cloud vs. Amazon web services -

Cloud Type (machine gen)	Small Input	Medium Input	Large Input
Google Cloud (n1-standard-1)	45 min 50 sec	37 min 24 sec	55 min 11 sec
Amazon EMR (m1.med)	19 minutes	15 minutes	21 minutes

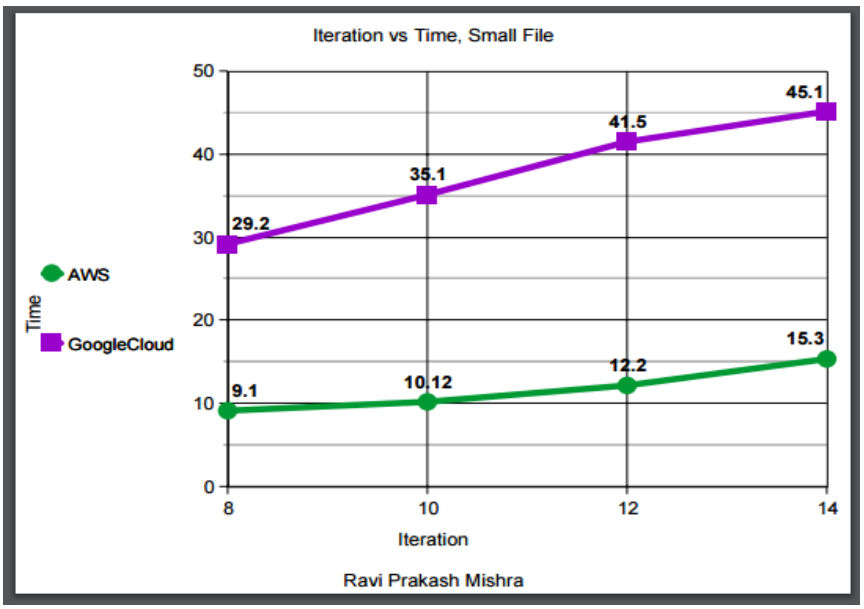
First 10 iterations time -

Cloud Type (machine gen)	Small Input	Medium Input	Large Input
Amazon EMR(m1.med)	10 min 12 sec	11 min 16 sec	10 min 01 sec
Google Cloud(n1-standard-1)	30 min 48 sec	27 min 55 sec	29 min 27 sec

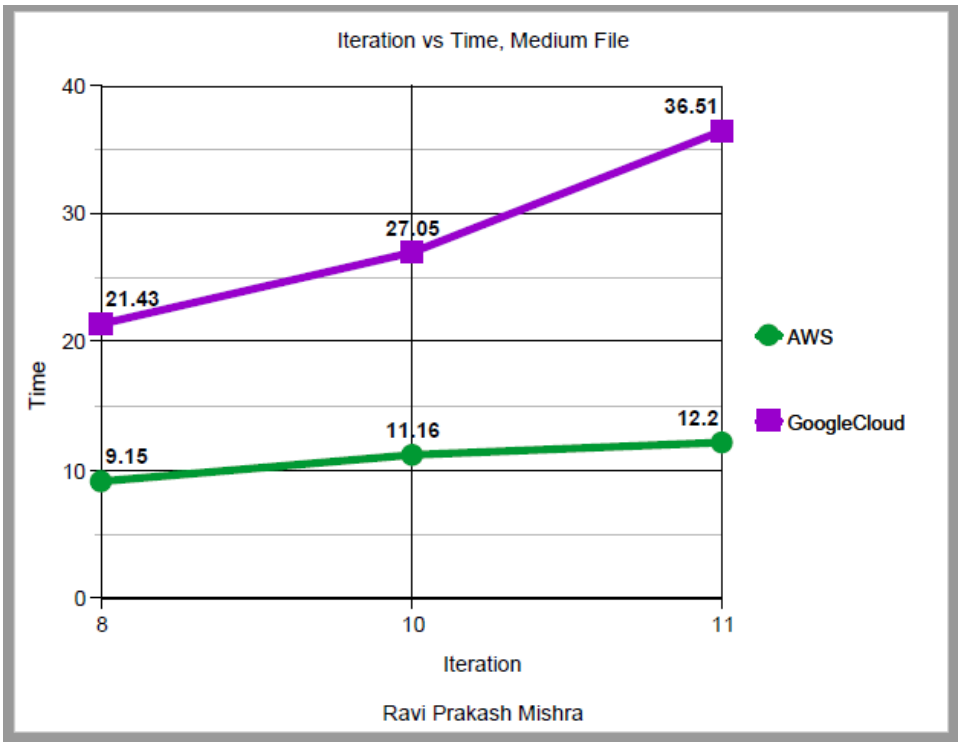
Iteration vs time –

Below are the graphs for the small, medium and large files on AWS and google cloud. It is evident from the graphs that iteraion progresses linearily with time.

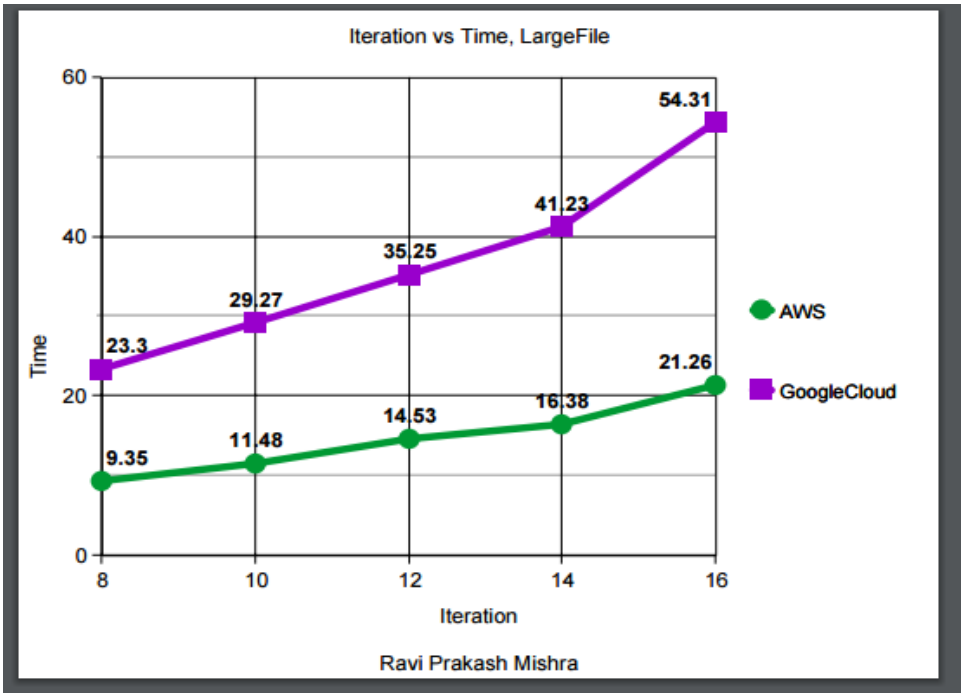
Small File



Medium File



Large File



Result from local hadoop execution –

- **For small file**

prog2-sample-sm | MyPageRank.java | part-00000

```
1.2404  104524212055442757665907965243560045101
0.893   823061567661945876296903500839674733 94
0.8847  134661996234159808488375170070473187582
0.8733  155346617108560808581184142629329729230
0.833   227109448702302113507903604759918416063
0.7807  168673252469550579557899067836420932546
0.7306  579793706157419286092830050343252200 88
0.6764  176495987834825257450737101676186061 07
0.5908  976680205381248088380653562678728878 71
0.5131  133772998861810280463554697803246893667
```

prog2-sample-sm | MyPageRank.java | part-00000 | part-00000

```
AddInfo
Number of Nodes: 94
Number of Edges: 195
Minimum Degree: 0
Maximum Degree: 5
Average Degree: 2
```

- **For medium file**

prog2-sample-sm | MyPageRank.java | part-00000

```
1.2805  111981443422667599916101641267414970874
0.7044  304426760625152844155987234180143550 61
0.5928  217182398344717121985059912345853998316
0.5148  104105844697470013276372331783894076726
0.5059  643632821489458762108903368728657553 43
0.4697  255141271871887572604204954207769279563
0.4468  298690743135077500802007851608046438995
0.438   116480772629362012002460626777081605400
0.4249  303806832053566290572095716352649981643
0.4135  148511838361064104411653673322648403910
```

prog2-sample-sm MyPageRank.java part-00000

```
AddInfo
Number of Nodes: 317
Number of Edges: 430
Minimum Degree: 0
Maximum Degree: 5
Average Degree: 1
```

- **For large file**

prog2-sample-sm MyPageRank.java part-00000

```
5.5828  640329419637502236015056967871231384 45
5.1618  11933741243794013314488192320804988 2442
3.7877  29441828984030197332267216930039492 4184
3.4797  28451023925191004642759305748644918 5085
3.4065  97008356640547621048472268562410523068
3.3935  17129678229587134900557165283241780 36
3.311   15647161731364441982668626578918440 2299
3.2787  21491768659455923649754762253345725 8166
2.7651  25947979394195914996030993368286605 9975
2.7118  41465870706060606689699857814850788091
```

prog2-sample-sm MyPageRank.java part-00000 part-00000

```
AddInfo
Number of Nodes: 1459
Number of Edges: 3545
Minimum Degree: 0
Maximum Degree: 5
Average Degree: 2
```

How to run program -

Create an EMR/dataproc cluster at-least with 3 VM instances. Upload input jar file clouhw.jar in s3/googlecloud bucket and all input files. Now, once EMR/dataproc cluster is up add step. In arguments first argument must be the input file path in s3, second argument must be output folder and third and last argument must be no. of iterations. If no of iterations are not provided, then it will take iteration number as 16.

GitHubLink

https://github.com/raviprakashmishra/PageRank/tree/master/HW