CSC 555: HW4

Name: Nachiketh Reddy
ID: 2117731

QUESTION 1:

Bloom Filter Consider a Bloom filter with 16 slots that uses 2 hash functions. To define the index position to set in the array, we will use the following procedure: Generate a hash value using SHA256 on your depaulid and a number. For example, to generate hash value of number x, you will need to run: Use the bloomfilter.py file shared in the class: (a) Add the strings '1', '3', '5', . . ., '19' to this Bloom filter and write down which bits have been set. (b) Test the resulting Bloom filter for the values '2', '4', '6', . . ., '20'. Do you get any false positives, and if so, what are they?

QUESTION 1

In [10]:

```python
import hashlib

class BloomFilter:
    def __init__(self, size, hash_functions):
        self.size = size  # Size of the filter (number of bits)
        self.hash_functions = hash_functions
        self.bit_array = [False] * size

    def add(self, element):
        for i in range(self.hash_functions):
            index = self._hash(element, i)
            self.bit_array[index] = True

    def contains(self, element):
        for i in range(self.hash_functions):
            index = self._hash(element, i)
            if not self.bit_array[index]:
                return False
        return True

    def _hash(self, element, i):
        # Generate hash value using SHA256 using DePaul ID and element
        hash_value = hashlib.sha256(bytes(f"nparamah {element}", encoding='utf-8')).hexdigest()
        if i == 0:
            hash_function = int(hash_value[0], 16)
        else:
            hash_function = int(hash_value[1], 16)
        # Return the hash value modulo the size of the filter
        return hash_function % self.size

# Creating a Bloom filter with 16 slots and 2 hash functions
filter_size = 16
num_hash_functions = 2
obj_bloom = BloomFilter(filter_size, num_hash_functions)

# Add the elements '1', '3', '5', ..., '19' to the Bloom filter
for i in range(1, 20, 2):
    obj_bloom.add(str(i))
    print(f"Bit at index {obj_bloom._hash(str(i), 0)} and {obj_bloom._hash(str(i), 1)} is set for {i}")

# Test the presence of elements '2', '4', '6', ..., '20' in the Bloom Filter
values_to_check = list(range(2, 20, 2))
for value in values_to_check:
    if obj_bloom.contains(str(value)):
        print(f"{value} may be in the set.")
    else:
        print(f"{value} is definitely not in the set.")
```

```
Bit at index 4 and 8 is set for 1
Bit at index 7 and 2 is set for 3
Bit at index 15 and 1 is set for 5
Bit at index 4 and 8 is set for 7
Bit at index 10 and 1 is set for 9
Bit at index 4 and 1 is set for 11
Bit at index 13 and 5 is set for 13
Bit at index 5 and 6 is set for 15
Bit at index 15 and 0 is set for 17
Bit at index 1 and 0 is set for 19
2 is definitely not in the set.
4 is definitely not in the set.
6 may be in the set.
8 is definitely not in the set.
10 may be in the set.
12 is definitely not in the set.
14 is definitely not in the set.
16 is definitely not in the set.
18 is definitely not in the set.
```

QUESTION 2:

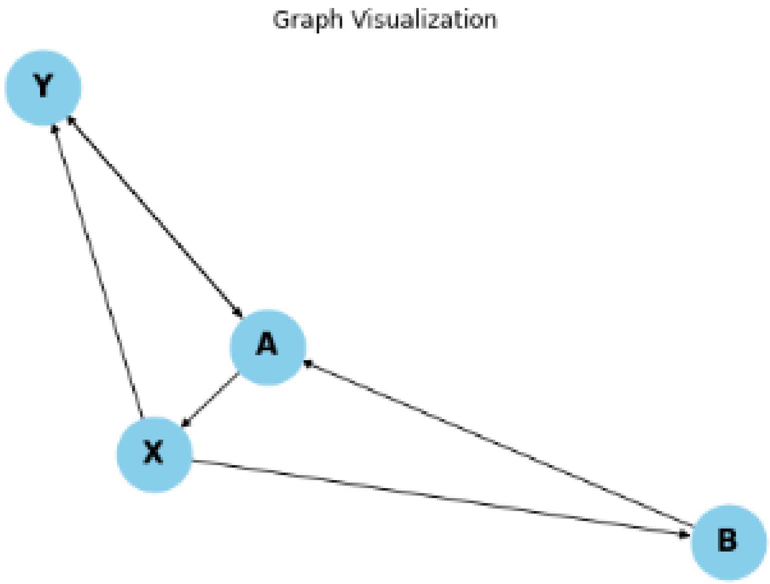
Link Matrix:

```
[[0.  1.  0.  1. ]
 [0.  0.  0.5 0. ]
 [0.5 0.  0.  0. ]
 [0.5 0.  0.5 0. ]]
```

Pages: `['A', 'B', 'X', 'Y']`

In [15]:

```python
import numpy as np
import networkx as nx
import matplotlib.pyplot as plt

def calculate_pagerank(connectivity_matrix, damping_factor=0.85, tolerance=1e-6):
    num_pages = connectivity_matrix.shape[1]
    pagerank_vector = np.ones((num_pages, 1)) / num_pages
    previous_pagerank_vector = np.ones((num_pages, 1)) * 100  # Initialize with a large difference
    pagerank_iterations = []  # Store PageRank vectors at each iteration
    while np.linalg.norm(pagerank_vector - previous_pagerank_vector, 2) > tolerance:
        previous_pagerank_vector = pagerank_vector.copy()
        pagerank_vector = damping_factor * np.dot(connectivity_matrix, pagerank_vector) + (1 - damping_factor) / num_pages
        pagerank_vector += ((1 - np.sum(pagerank_vector)) / num_pages)  # Adding missing probability mass to handle dead-end nod
        pagerank_vector = pagerank_vector / np.sum(pagerank_vector)  # Normalize the PageRank vector
        pagerank_iterations.append(pagerank_vector.flatten())
    return pagerank_iterations

# input matrix representing the connectivity of web pages
webpage_connectivity = np.array([[0, 1, 0, 1],
                                 [0, 0, 0.5, 0],
                                 [0.5, 0, 0, 0],
                                 [0.5, 0, 0.5, 0]])

# Running the PageRank algorithm until convergence on the input matrix
page_rank_iterations = calculate_pagerank(webpage_connectivity)

# Create a directed graph using NetworkX
graph = nx.DiGraph()

# Add edges from the matrix
edges = [("A", "Y"), ("A", "X"), ("B", "A"), ("X", "B"), ("X", "Y"), ("Y", "A")]
graph.add_edges_from(edges)

# Draw the graph
positions = nx.spring_layout(graph)
nx.draw(graph, positions, with_labels=True, node_size=1500, node_color="skyblue", font_size=15, font_weight="bold", arrows=True)
plt.title("Graph Visualization")
plt.show()

# Print all iterations until convergence
for i, page_rank in enumerate(page_rank_iterations):
    print("Iteration", i+1, "Page ranks:", page_rank)

# Print the final iteration in the respective page rank order
final_page_ranks = page_rank_iterations[-1]
page_ranks_with_labels = [(page, rank) for page, rank in zip(['A', 'B', 'X', 'Y'], final_page_ranks)]
page_ranks_with_labels.sort(key=lambda x: x[1], reverse=True)

print("Final Page Ranks:")
for page, rank in page_ranks_with_labels:
    print(f"{page}: {rank}")
```




```
Iteration 1 Page ranks: [0.4625  0.14375 0.14375 0.25   ]
Iteration 2 Page ranks: [0.3721875  0.09859375 0.2340625  0.29515625]
Iteration 3 Page ranks: [0.3721875  0.13697656 0.19567969 0.29515625]
Iteration 4 Page ranks: [0.40481289 0.12066387 0.19567969 0.27884355]
Iteration 5 Page ranks: [0.37708131 0.12066387 0.20954548 0.29270935]
Iteration 6 Page ranks: [0.38886723 0.12655683 0.19775956 0.28681638]
Iteration 7 Page ranks: [0.38886723 0.12154781 0.20276857 0.28681638]
Iteration 8 Page ranks: [0.38460957 0.12367664 0.20276857 0.28894522]
Iteration 9 Page ranks: [0.38822858 0.12367664 0.20095907 0.28713571]
Iteration 10 Page ranks: [0.3866905  0.1229076  0.20249715 0.28790475]
Iteration 11 Page ranks: [0.3866905  0.12356129 0.20184346 0.28790475]
Iteration 12 Page ranks: [0.38724613 0.12328347 0.20184346 0.28762693]
Iteration 13 Page ranks: [0.38677384 0.12328347 0.20207961 0.28786308]
Iteration 14 Page ranks: [0.38697457 0.12338383 0.20187888 0.28776272]
Iteration 15 Page ranks: [0.38697457 0.12329853 0.20196419 0.28776272]
Iteration 16 Page ranks: [0.38690206 0.12333478 0.20196419 0.28779897]
Iteration 17 Page ranks: [0.38696369 0.12333478 0.20193337 0.28776815]
Iteration 18 Page ranks: [0.3869375  0.12332168 0.20195957 0.28778125]
Iteration 19 Page ranks: [0.3869375  0.12333282 0.20194844 0.28778125]
Iteration 20 Page ranks: [0.38694696 0.12332809 0.20194844 0.28777652]
Iteration 21 Page ranks: [0.38693892 0.12332809 0.20195246 0.28778054]
Iteration 22 Page ranks: [0.38694233 0.12332979 0.20194904 0.28777883]
Iteration 23 Page ranks: [0.38694233 0.12332834 0.20195049 0.28777883]
Iteration 24 Page ranks: [0.3869411  0.12332896 0.20195049 0.28777945]
Iteration 25 Page ranks: [0.38694215 0.12332896 0.20194997 0.28777893]
Iteration 26 Page ranks: [0.3869417  0.12332874 0.20195041 0.28777915]
Final Page Ranks:
A: 0.3869417021334981
Y: 0.28777914893325096
X: 0.20195041300356312
B: 0.12332873592968788
```

QUESTION 3

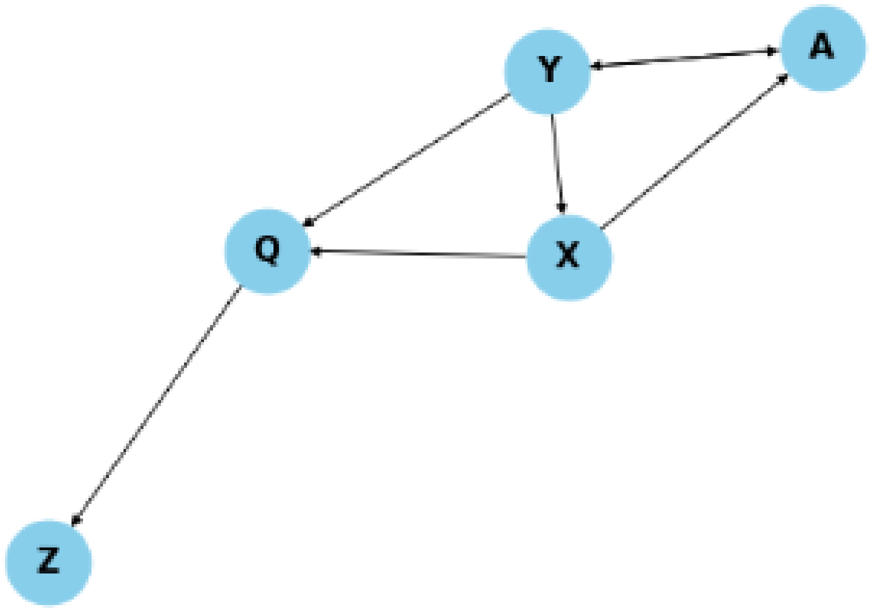
Link Matrix:

```
[[0.         0.         0.5        0.33333333 0.        ]
 [0.         0.         0.5        0.33333333 0.        ]
 [0.         0.         0.         0.33333333 0.        ]
 [1.         0.         0.         0.         0.        ]
 [0.         1.         0.         0.         0.        ]]
```

Pages: ['A', 'Q', 'X', 'Y', 'Z']

In [16]:

```
import numpy as np
import networkx as nx
import matplotlib.pyplot as plt

def calculate_pagerank(connectivity_matrix, damping_factor=0.85, tolerance=1e-6):
    num_pages = connectivity_matrix.shape[1]
    pagerank_vector = np.ones((num_pages, 1)) / num_pages
    previous_pagerank_vector = np.ones((num_pages, 1)) * 100  # Initialize with a large difference
    pagerank_iterations = []  # Store PageRank vectors at each iteration
    while np.linalg.norm(pagerank_vector - previous_pagerank_vector, 2) > tolerance:
        previous_pagerank_vector = pagerank_vector.copy()
        pagerank_vector = damping_factor * np.dot(connectivity_matrix, pagerank_vector) + (1 - damping_factor) / num_pages
        pagerank_vector += ((1 - np.sum(pagerank_vector)) / num_pages)  # Adding missing probability mass to handle dead-end nod
        pagerank_vector = pagerank_vector / np.sum(pagerank_vector)  # Normalize the PageRank vector
        pagerank_iterations.append(pagerank_vector.flatten())
    return pagerank_iterations

# input matrix representing the connectivity of web pages
webpage_connectivity = np.array([[0.0,0.0,0.5,0.33333333,0.0],
 [0.0,0.0,0.5,0.33333333,0.0],
 [0.0,0.0,0.0,0.33333333,0.0],
 [1.0,0.0,0.0,0.0,0.0],
 [0.0,1.0,0.0,0.0,0.0]])

# Running the PageRank algorithm until convergence on the input matrix
page_rank_iterations = calculate_pagerank(webpage_connectivity)

# Create a directed graph using NetworkX
graph = nx.DiGraph()

# Add edges from the matrix
edges = [

    ('A', 'Y'),
    ('Y', 'A'),
    ('Y', 'X'),
    ('Y', 'Q'),
    ('X', 'A'),
    ('X', 'Q'),
    ('Q', 'Z')

]
graph.add_edges_from(edges)

# Draw the graph
positions = nx.spring_layout(graph)
nx.draw(graph, positions, with_labels=True, node_size=1500, node_color="skyblue", font_size=15, font_weight="bold", arrows=True)
plt.title("Graph Visualization")
plt.show()

# Print all iterations until convergence
for i, page_rank in enumerate(page_rank_iterations):
    print("Iteration", i+1, "Page ranks:", page_rank)

# Print the final iteration in the respective page rank order
final_page_ranks = page_rank_iterations[-1]
page_ranks_with_labels = [(page, rank) for page, rank in zip(['A','Q','X','Y','Z'], final_page_ranks)]
page_ranks_with_labels.sort(key=lambda x: x[1], reverse=True)

print("Final Page Ranks:")
for page, rank in page_ranks_with_labels:
    print(f"{page}: {rank}")
```




```
Iteration 1 Page ranks: [0.20566667 0.20566667 0.12066667 0.234      0.234     ]
Iteration 2 Page ranks: [0.18736333 0.18736333 0.13608    0.24459667 0.24459667]
Iteration 3 Page ranks: [0.19871782 0.19871782 0.14088382 0.23084027 0.23084027]
Iteration 4 Page ranks: [0.19452321 0.19452321 0.13464759 0.23815299 0.23815299]
Iteration 5 Page ranks: [0.19518792 0.19518792 0.13796269 0.23583074 0.23583074]
Iteration 6 Page ranks: [0.19554408 0.19554408 0.13690993 0.23600095 0.23600095]
Iteration 7 Page ranks: [0.19517382 0.19517382 0.1369871  0.23633263 0.23633263]
Iteration 8 Page ranks: [0.19535698 0.19535698 0.13713746 0.2360743  0.2360743 ]
Iteration 9 Page ranks: [0.19530377 0.19530377 0.13702035 0.23618606 0.23618606]
Iteration 10 Page ranks: [0.19530466 0.19530466 0.13707101 0.23615983 0.23615983]
Iteration 11 Page ranks: [0.1953143  0.1953143  0.13705912 0.23615613 0.23615613]
Iteration 12 Page ranks: [0.19530757 0.19530757 0.13705745 0.2361637  0.2361637 ]
Iteration 13 Page ranks: [0.19531029 0.19531029 0.13706088 0.23615927 0.23615927]
Iteration 14 Page ranks: [0.19530974 0.19530974 0.13705887 0.23616082 0.23616082]
Iteration 15 Page ranks: [0.19530959 0.19530959 0.13705957 0.23616062 0.23616062]
Final Page Ranks:
Y: 0.23616062055337508
Z: 0.23616062055337508
A: 0.1953095925906417
Q: 0.1953095925906417
X: 0.13705957371196648
```

Y=Z followed by A=Q and then X

QUESTION 4

Suppose we recursively eliminate dead ends from the graph, solve the remaining graph, and estimate the PageRank for the dead-end pages as described in Section 5.1.4. Suppose the graph is a chain of dead ends, headed by a node with a self-loop, as suggested in Fig. 5.9. What would be the Page Rank assigned to each of the nodes?

The scenario outlined consists of a series of dead-end sites, where the initial node (Node 1) has a self-loop and each page links to the next. This is how each node's PageRank would be calculated:

Node 1: Self-Looping Root Node Node 1 keeps all of its significance and transfers its entire PageRank to itself because it has a self-loop. As a result, Node 1's PageRank stays at 1.

Dead-End Nodes That Follow (Node 2 to Node n): There is only one inbound link from each preceding node (Node 2 to Node n).

Hence, node 1, the root node with a self-loop, continues to have a PageRank of 1. Because each successive dead-end node (Node 2 to Node n) receives half of its PageRank from its predecessors, their PageRank is 1/2. As a result, each node in the chain of dead ends, which is led by a node with a self-loop, would have the following PageRank assigned to it:

Node 1: 1

Node 2 to Node n: 1/2

QUESTION 5:

a. Take a look at the file and report how many nodes and edges the web-Stanford.txt contains.

ix_1 | matrix_600x | matrix600 | matrix600 | matrix600 | matrix600 | matrix_600x | Spotify_Met | Spotify_User | Spotify_Num | web-Stanfor | web-

File Edit View

```
# Directed graph (each unordered pair of nodes is saved once): web-Stanford.txt
# Stanford web graph from 2002
# Nodes: 281903 Edges: 2312497
# FromNodeId	ToNodeId
1	6548
1	15409
6548	57031
15409	13102
2	17794
2	25202
2	53625
2	54582
2	64930
2	73764
2	84477
2	98628
2	100193
2	102355
2	105318
2	105730
2	115926
2	140864
2	163550
2	164599
2	175799
2	178642
2	181714
2	190453
2	204189
2	204604
```

Ln 1, Col 1 | 30,575,832 characters | 100% | Windows (CRLF)

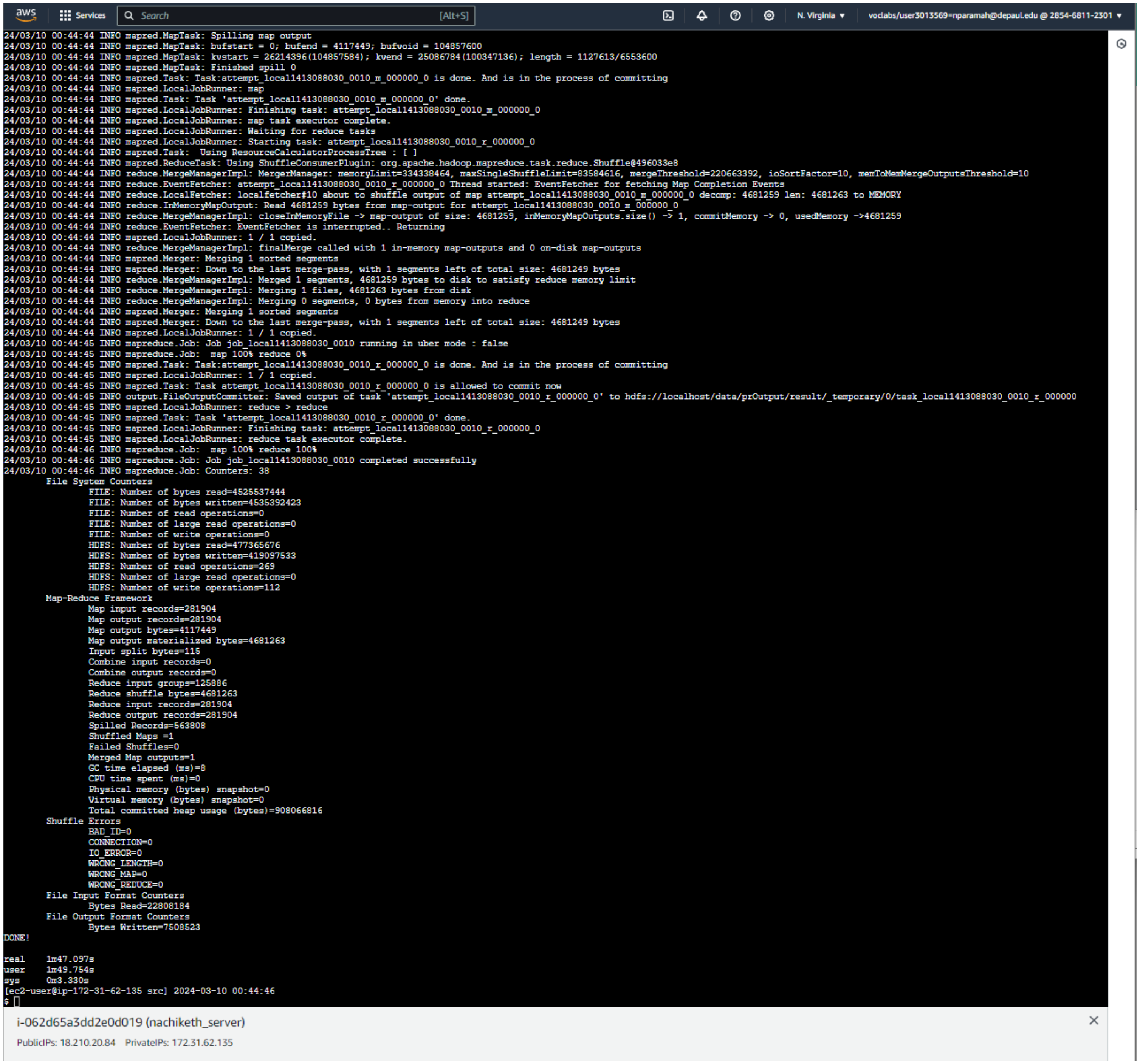
b. Report the runtime (took about 5 minutes to run when I tested it)

SINGLE NODE:

Real 1m47.097S

User 1m49.754s

Sys 0m3.330s



```
aws   Services   Search   [Alt+S]   N. Virginia   voclabs/user3013569=nparamah@depaul.edu @ 2854-6811-2301
24/03/10 00:44:44 INFO mapred.MapTask: Spilling map output
24/03/10 00:44:44 INFO mapred.MapTask: bufstart = 0; bufend = 4117449; bufvoid = 104857600
24/03/10 00:44:44 INFO mapred.MapTask: kvstart = 26214396(104857584); kvend = 25086784(100347136); length = 1127613/6553600
24/03/10 00:44:44 INFO mapred.MapTask: Finished spill 0
24/03/10 00:44:44 INFO mapred.Task: Task:attempt_local1413088030_0010_m_000000_0 is done. And is in the process of committing
24/03/10 00:44:44 INFO mapred.LocalJobRunner: map
24/03/10 00:44:44 INFO mapred.Task: Task 'attempt_local1413088030_0010_m_000000_0' done.
24/03/10 00:44:44 INFO mapred.LocalJobRunner: Finishing task: attempt_local1413088030_0010_m_000000_0
24/03/10 00:44:44 INFO mapred.LocalJobRunner: map task executor complete.
24/03/10 00:44:44 INFO mapred.LocalJobRunner: Waiting for reduce tasks
24/03/10 00:44:44 INFO mapred.LocalJobRunner: Starting task: attempt_local1413088030_0010_r_000000_0
24/03/10 00:44:44 INFO mapred.Task:  Using ResourceCalculatorProcessTree : [ ]
24/03/10 00:44:44 INFO mapred.ReduceTask: Using ShuffleConsumerPlugin: org.apache.hadoop.mapreduce.task.reduce.Shuffle@496033e8
24/03/10 00:44:44 INFO reduce.MergeManagerImpl: MergerManager: memoryLimit=334338464, maxSingleShuffleLimit=83584616, mergeThreshold=220663392, ioSortFactor=10, memToMemMergeOutputsThreshold=10
24/03/10 00:44:44 INFO reduce.EventFetcher: attempt_local1413088030_0010_r_000000_0 Thread started: EventFetcher for fetching Map Completion Events
24/03/10 00:44:44 INFO reduce.LocalFetcher: localfetcher#10 about to shuffle output of map attempt_local1413088030_0010_m_000000_0 decomp: 4681259 len: 4681263 to MEMORY
24/03/10 00:44:44 INFO reduce.InMemoryMapOutput: Read 4681259 bytes from map-output for attempt_local1413088030_0010_m_000000_0
24/03/10 00:44:44 INFO reduce.MergeManagerImpl: closeInMemoryFile -> map-output of size: 4681259, inMemoryMapOutputs.size() -> 1, commitMemory -> 0, usedMemory ->4681259
24/03/10 00:44:44 INFO reduce.EventFetcher: EventFetcher is interrupted.. Returning
24/03/10 00:44:44 INFO mapred.LocalJobRunner: 1 / 1 copied.
24/03/10 00:44:44 INFO reduce.MergeManagerImpl: finalMerge called with 1 in-memory map-outputs and 0 on-disk map-outputs
24/03/10 00:44:44 INFO mapred.Merger: Merging 1 sorted segments
24/03/10 00:44:44 INFO mapred.Merger: Down to the last merge-pass, with 1 segments left of total size: 4681249 bytes
24/03/10 00:44:44 INFO reduce.MergeManagerImpl: Merged 1 segments, 4681259 bytes to disk to satisfy reduce memory limit
24/03/10 00:44:44 INFO reduce.MergeManagerImpl: Merging 1 files, 4681263 bytes from disk
24/03/10 00:44:44 INFO reduce.MergeManagerImpl: Merging 0 segments, 0 bytes from memory into reduce
24/03/10 00:44:44 INFO mapred.Merger: Merging 1 sorted segments
24/03/10 00:44:44 INFO mapred.Merger: Down to the last merge-pass, with 1 segments left of total size: 4681249 bytes
24/03/10 00:44:44 INFO mapred.LocalJobRunner: 1 / 1 copied.
24/03/10 00:44:45 INFO mapreduce.Job: Job job_local1413088030_0010 running in uber mode : false
24/03/10 00:44:45 INFO mapreduce.Job:  map 100% reduce 0%
24/03/10 00:44:45 INFO mapred.Task: Task:attempt_local1413088030_0010_r_000000_0 is done. And is in the process of committing
24/03/10 00:44:45 INFO mapred.LocalJobRunner: 1 / 1 copied.
24/03/10 00:44:45 INFO mapred.Task: Task attempt_local1413088030_0010_r_000000_0 is allowed to commit now
24/03/10 00:44:45 INFO output.FileOutputCommitter: Saved output of task 'attempt_local1413088030_0010_r_000000_0' to hdfs://localhost/data/prOutput/result/_temporary/0/task_local1413088030_0010_r_000000
24/03/10 00:44:45 INFO mapred.LocalJobRunner: reduce > reduce
24/03/10 00:44:45 INFO mapred.Task: Task 'attempt_local1413088030_0010_r_000000_0' done.
24/03/10 00:44:45 INFO mapred.LocalJobRunner: Finishing task: attempt_local1413088030_0010_r_000000_0
24/03/10 00:44:45 INFO mapred.LocalJobRunner: reduce task executor complete.
24/03/10 00:44:46 INFO mapreduce.Job:  map 100% reduce 100%
24/03/10 00:44:46 INFO mapreduce.Job: Job job_local1413088030_0010 completed successfully
24/03/10 00:44:46 INFO mapreduce.Job: Counters: 38
        File System Counters
                FILE: Number of bytes read=4525537444
                FILE: Number of bytes written=4535392423
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=477365676
                HDFS: Number of bytes written=419097533
                HDFS: Number of read operations=269
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=112
        Map-Reduce Framework
                Map input records=281904
                Map output records=281904
                Map output bytes=4117449
                Map output materialized bytes=4681263
                Input split bytes=115
                Combine input records=0
                Combine output records=0
                Reduce input groups=125886
                Reduce shuffle bytes=4681263
                Reduce input records=281904
                Reduce output records=281904
                Spilled Records=563808
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=8
                CPU time spent (ms)=0
                Physical memory (bytes) snapshot=0
                Virtual memory (bytes) snapshot=0
                Total committed heap usage (bytes)=908066816
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=22808184
        File Output Format Counters
                Bytes Written=7508523
DONE!

real    1m47.097s
user    1m49.754s
sys     0m3.330s
[ec2-user@ip-172-31-62-135 src] 2024-03-10 00:44:46
$
i-062d65a3dd2e0d019 (nachiketh_server)
PublicIPs: 18.210.20.84   PrivateIPs: 172.31.62.135
```

aws Services Search [Alt+S] N. Virginia vocIabs/user3013569=nparamah@depaul.edu @ 2854-6811-230

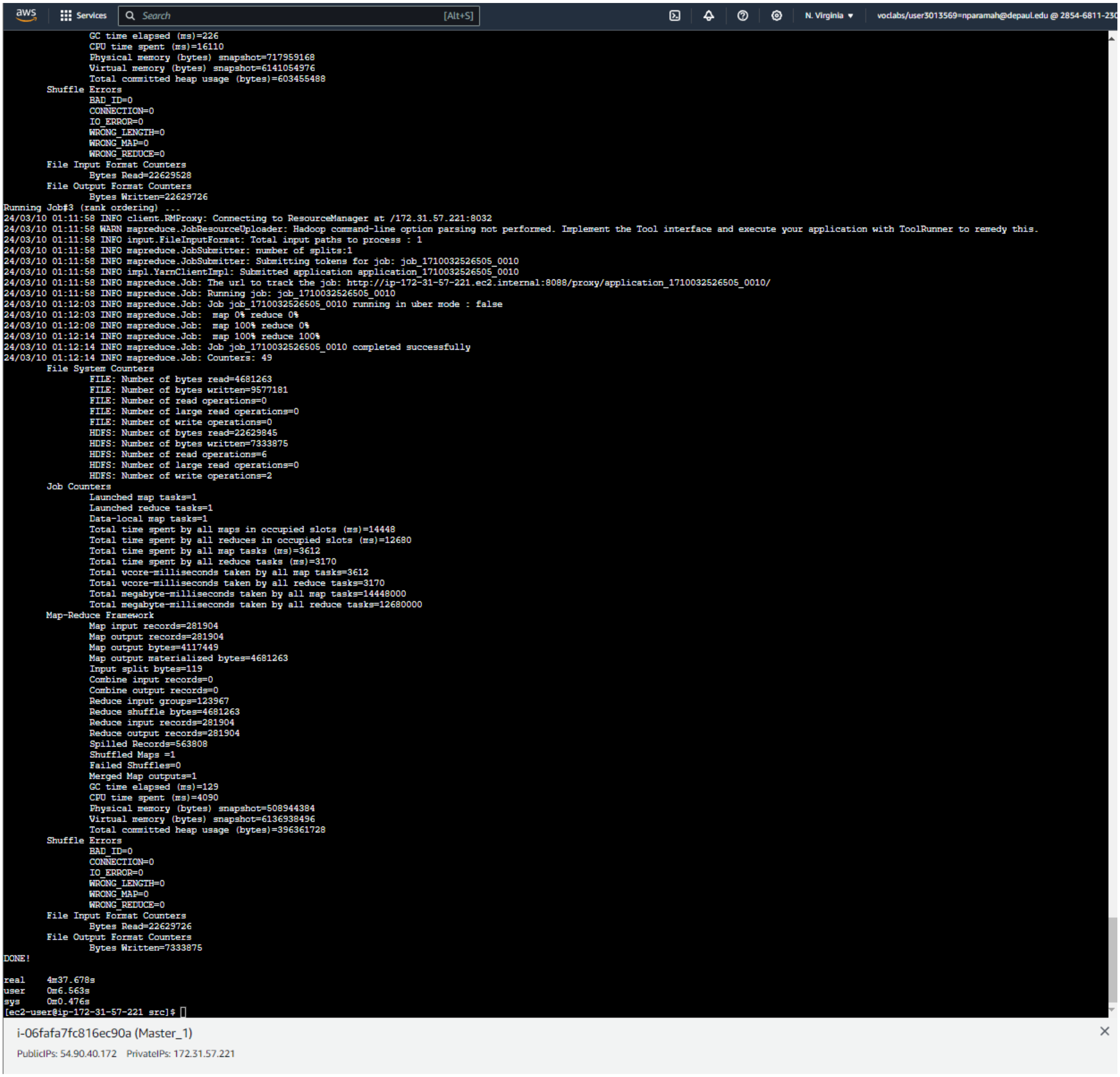
```
[ec2-user@ip-172-31-62-135 ~] 2024-03-10 00:55:31
$ hadoop fs -cat /data/prOutput/result/part-r-00000 | more
0.0     154952
0.0     36711
0.0     154931
0.0     3671
0.0     36689
0.0     36702
0.0     36693
0.0     99979
0.0     1792
0.0     24032
0.0     179164
0.0     24033
0.0     262390
0.0     179163
0.0     100020
0.0     100022
0.0     9995
0.0     99949
0.0     17915
0.0     244555
0.0     262402
0.0     262403
0.0     99924
0.0     218029
0.0     262405
0.0     100048
0.0     194867
0.0     194854
0.0     218032
0.0     99904
0.0     100058
0.0     100059
0.0     100061
0.0     179206
0.0     244561
0.0     194850
0.0     100069
0.0     100074
0.0     218039
0.0     262422
0.0     198464
0.0     99869
0.0     100084
0.0     244567
0.0     100092
0.0     262338
0.0     262431
0.0     217985
0.0     100094
0.0     99841
0.0     10010
0.0     21798
0.0     262443
0.0     218050
0.0     100107
0.0     100109
0.0     244575
0.0     99823
0.0     100114
0.0     262449
0.0     217974
0.0     99814
0.0     262457
0.0     100124
0.0     217970
0.0     179104
0.0     262333
0.0     998
0.0     194826
0.0     99794
0.0     218063
0.0     218066
0.0     244580
0.0     99785
0.0     100137
0.0     198474
0.0     100148
0.0     99770
0.0     179093
0.0     262485
0.0     100158
0.0     99763
0.0     262329
0.0     100165
0.0     100168
0.0     100171
0.0     218073
0.0     9975
0.0     179085
0.0     217950
```

i-062d65a3dd2e0d019 (nachiketh_server)

PublicIPs: 18.210.20.84 PrivateIPs: 172.31.62.135

MULTI NODE:

Real 4m37.678S
User 0m6.563s
Sys 0m0.476s



aws Services Search [Alt+S] N. Virginia vocdabs/user3013569=nparamah@depaul.edu @ 2854-6811-230

```
                GC time elapsed (ms)=226
                CPU time spent (ms)=16110
                Physical memory (bytes) snapshot=717959168
                Virtual memory (bytes) snapshot=6141054976
                Total committed heap usage (bytes)=603455488
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=22629528
        File Output Format Counters
                Bytes Written=22629726
Running Job#3 (rank ordering) ...
24/03/10 01:11:58 INFO client.RMProxy: Connecting to ResourceManager at /172.31.57.221:8032
24/03/10 01:11:58 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
24/03/10 01:11:58 INFO input.FileInputFormat: Total input paths to process : 1
24/03/10 01:11:58 INFO mapreduce.JobSubmitter: number of splits:1
24/03/10 01:11:58 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1710032526505_0010
24/03/10 01:11:58 INFO impl.YarnClientImpl: Submitted application application_1710032526505_0010
24/03/10 01:11:58 INFO mapreduce.Job: The url to track the job: http://ip-172-31-57-221.ec2.internal:8088/proxy/application_1710032526505_0010/
24/03/10 01:11:58 INFO mapreduce.Job: Running job: job_1710032526505_0010
24/03/10 01:12:03 INFO mapreduce.Job: Job job_1710032526505_0010 running in uber mode : false
24/03/10 01:12:03 INFO mapreduce.Job:  map 0% reduce 0%
24/03/10 01:12:08 INFO mapreduce.Job:  map 100% reduce 0%
24/03/10 01:12:14 INFO mapreduce.Job:  map 100% reduce 100%
24/03/10 01:12:14 INFO mapreduce.Job: Job job_1710032526505_0010 completed successfully
24/03/10 01:12:14 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=4681263
                FILE: Number of bytes written=9577181
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=22629845
                HDFS: Number of bytes written=7333875
                HDFS: Number of read operations=6
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Launched map tasks=1
                Launched reduce tasks=1
                Data-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=14448
                Total time spent by all reduces in occupied slots (ms)=12680
                Total time spent by all map tasks (ms)=3612
                Total time spent by all reduce tasks (ms)=3170
                Total vcore-milliseconds taken by all map tasks=3612
                Total vcore-milliseconds taken by all reduce tasks=3170
                Total megabyte-milliseconds taken by all map tasks=14448000
                Total megabyte-milliseconds taken by all reduce tasks=12680000
        Map-Reduce Framework
                Map input records=281904
                Map output records=281904
                Map output bytes=4117449
                Map output materialized bytes=4681263
                Input split bytes=119
                Combine input records=0
                Combine output records=0
                Reduce input groups=123967
                Reduce shuffle bytes=4681263
                Reduce input records=281904
                Reduce output records=281904
                Spilled Records=563808
                Shuffled Maps =1
                Failed Shuffles=0
                Merged Map outputs=1
                GC time elapsed (ms)=129
                CPU time spent (ms)=4090
                Physical memory (bytes) snapshot=508944384
                Virtual memory (bytes) snapshot=6136938496
                Total committed heap usage (bytes)=396361728
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=22629726
        File Output Format Counters
                Bytes Written=7333875
DONE!

real    4m37.678s
user    0m6.563s
sys     0m0.476s
[ec2-user@ip-172-31-57-221 src]$
```

i-06fafa7fc816ec90a (Master_1)

PublicIPs: 54.90.40.172 PrivateIPs: 172.31.57.221

c. Submit a screenshot of the first page of nodes, e.g., by running: