Page Rank Performance Analysis on Academic Cloud

Purshottam Vishwakarma
School of Informatics and Computing
Indiana University, Bloomington
pvishwak@indiana.edu

Bitan Saha
School of Informatics and Computing
Indiana University, Bloomington
bsaha@indiana.edu

Introduction:

The objective of running page rank algorithm on cloud environment is to obtain gain in the performance of the algorithm based on several parameters like number of cores, number of nodes, number of MPI processes, number of iterations, number of URLs, etc. In this report we present several experiments performed with varying parameter values to best reflect the gain in performance. The experiments are performed on **FutureGrid** and **Eucalyptus**. We like to mention that our sequential Page Rank Algorithm itself is much faster (by orders of magnitude) than other sequential Page Rank Algorithms. Hence, the relative performance gain is not too high (18 times). In the end we provide our conclusions with respect to both bare metal and eucalyptus which I find it very interesting.

Paramters and Data Sets:

The experiments were performed with the following parameters remaining constant.
Damping factor = 0.85
Number of Iterations = 10

Below Parameters were changed with different values during the experiment.
Number of Urls: 1K, 10K, 20K, 30K, 40K, 50K , 60K, 70K, 80K, 90K, 100K, 500K, 1M and 2M
Number of Nodes: 1 Node (8 Cores), 2 Nodes (16 Cores)
Number of MPI Processes: 2, 4, 6, 8, 10, 12, 14, 16
Number of VMs: m1.large(2 CPU cores) : 2, 4, 6, 8
c1.xlarge(8 CPU cores) : 1, 2

Note:

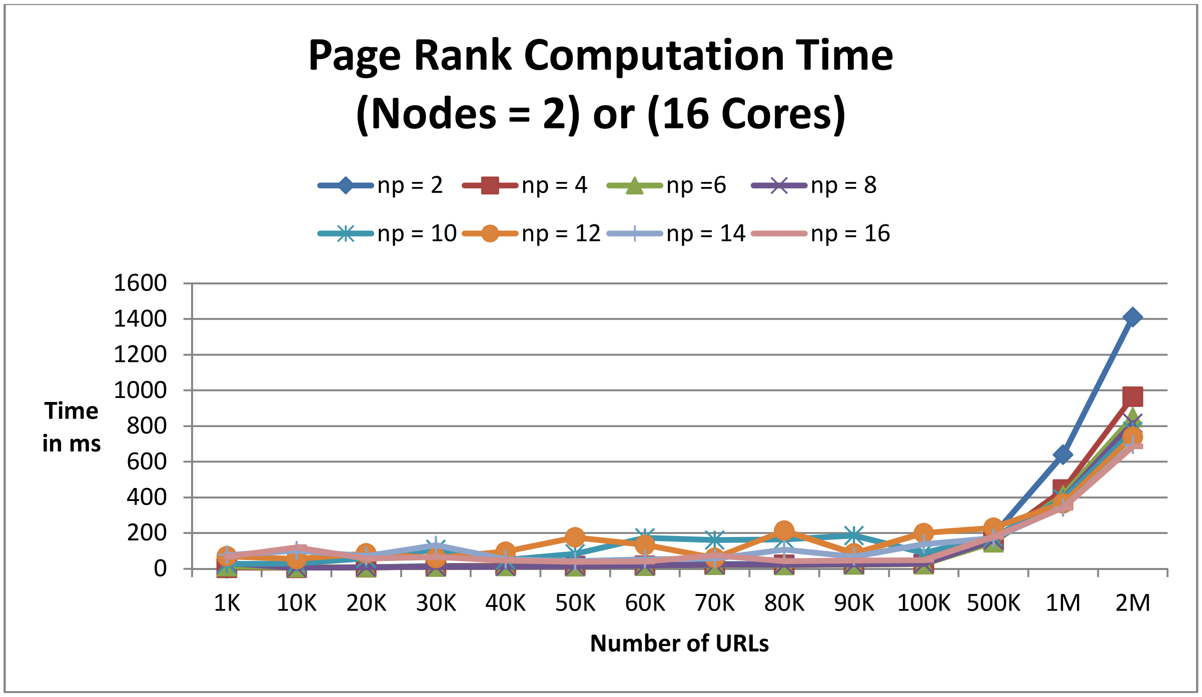
i. All the values were obtained as the average of 3 runs.
ii. The timings are purely computational timings i.e. time taken to calculate the page rank after 10 iterations. We have *not included the time for writing the final page rank values in the file* to simplify the analysis.

Results Bare Metal (*along with explanations*):

1. The first set of experiments was performed with 2 nodes (16 Cores) and all different number of URLs. Below table shows the page rank computation time in milliseconds (ms).

No. Of Urls	np = 2	np = 4	np =6	np = 8	np = 10	np = 12	np = 14	np = 16
1K	2	4	12	32	27	70	77	67
10K	5	4	7	5	30	54	102	120

20K	8	7	9	9	58	86	71	56
30K	15	12	10	10	107	60	132	65
40K	17	17	14	12	49	96	54	46
50K	16	14	13	14	83	175	42	38
60K	20	16	16	16	173	133	53	40
70K	26	22	21	22	160	60	57	74
80K	26	21	20	22	164	213	107	42
90K	28	24	23	24	186	85	68	46
100K	32	26	26	27	89	200	137	48
500K	188	158	148	160	176	229	171	179
1M	639	443	411	391	392	367	343	341
2M	1410	963	849	815	770	740	694	685

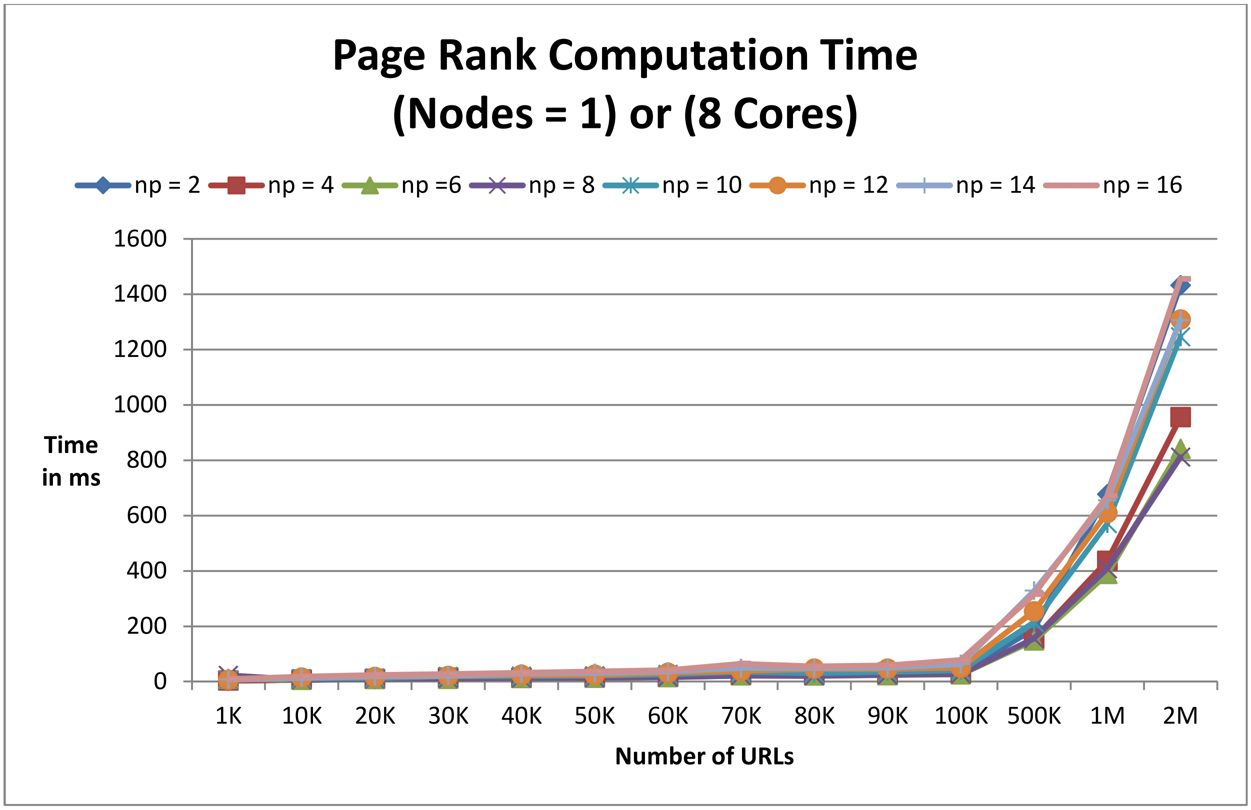


OBSERVATION: *Number of MPI Processes = 8 gives the best Performance Gain*

2. The second set of experiments was performed with **1 node (8 Cores)** and all different number of URLs. Below table shows the page rank computation time in milliseconds (ms).

No. Of Urls	np = 2	np = 4	np =6	np = 8	np = 10	np = 12	np = 14	np = 16
1K	2	3	10	22	6	6	7	9
10K	6	6	6	7	11	13	13	18
20K	8	9	9	9	12	17	19	24
30K	12	14	9	11	19	20	21	27
40K	18	13	14	13	19	24	28	31
50K	20	14	14	14	21	24	31	36

60K	20	16	17	16	26	31	36	41
70K	26	23	22	22	35	38	49	65
80K	26	22	22	21	30	46	49	55
90K	28	23	23	25	36	46	50	58
100K	32	28	26	27	41	50	63	78
500K	190	153	148	155	213	254	328	316
1M	676	436	389	408	571	610	654	671
2M	1432	955	841	811	1246	1308	1306	1455



OBSERVATION: *Number of MPI Processes = 8 gives the best Performance Gain*

- When comparing the with respect to number of MPI processes, it is evident that the performance for np =8 is much better performance with np = 16 (highlighted with yellow in the table). This is due to the fact that there are too many processes for which there is a huge communication overhead. With np = 16, the master process has to collect data from 15 different processes. Hence, in comparison to np = 8, it spends more time communicating which leads to performance degradation.
- When comparing single node and 2 node scenario, two nodes performance is marginally better than single node scenario. But as the number of nodes increases, the performance might decrease because there are more inter-machine communication and network latency impacts the performance. In case of single node, all the cores are in the same physical machine and hence there is no network communication overhead.

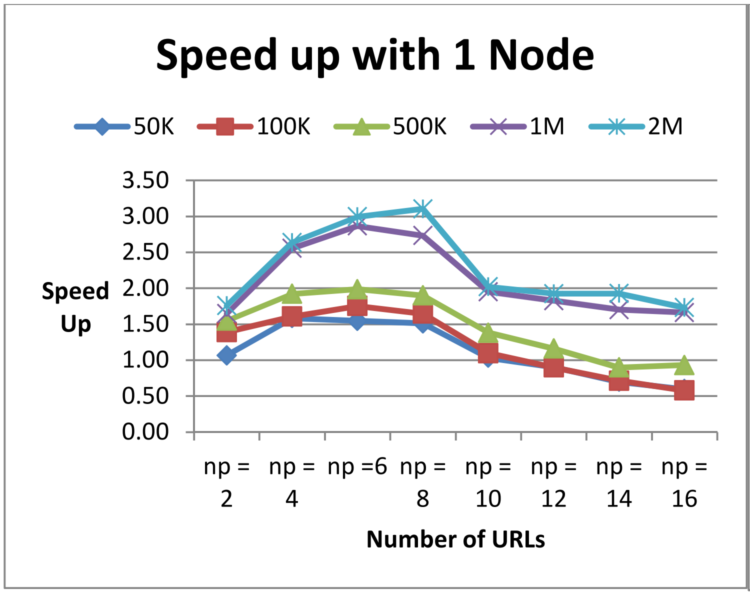
3. **Performance Gain (Speed up)**

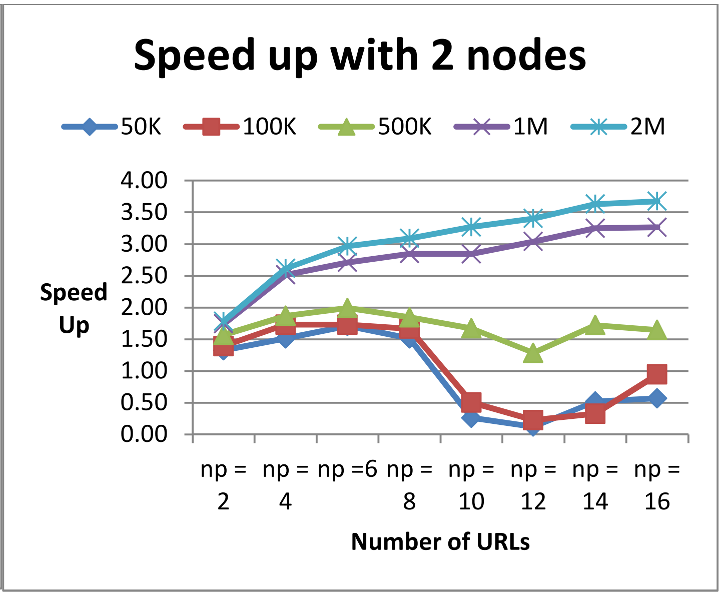
We measured the performance gain when number of MPI processes is 8 and 16. The Speed up is given by the formula

$$\text{Speedup} = \frac{T_1}{T_p}$$

- p is the total number of cores/processes
- T_1 is the execution time of the sequential algorithm
- T_p is the execution time of the parallel algorithm with p cores/processors

Best Speed up we received is ***3.11 times (1 node, 2M URLs and np = 8) and 3.67 times (2 nodes,2M URLs and np =16)*** when number of MPI processes = 6.





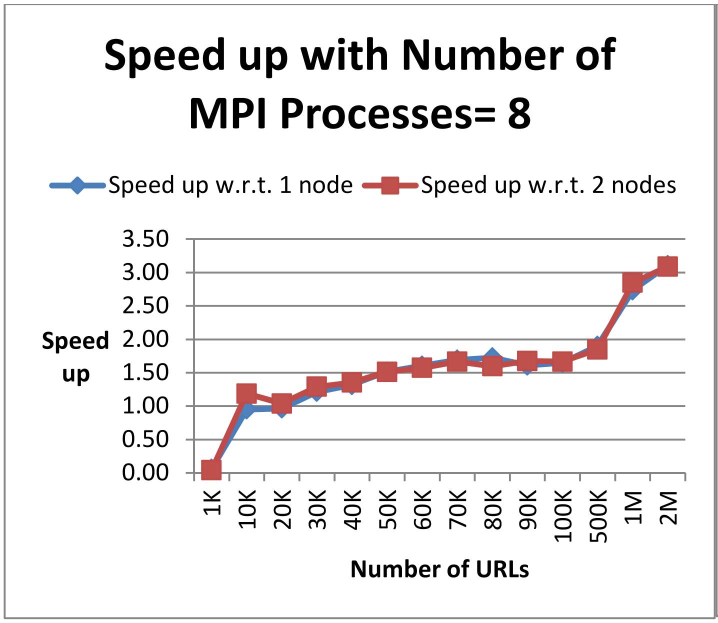
OBSERVATION: *Performance degrades with (Number of MPI Processes) np > 8 due to increased communication overhead between 2 machines (each 8 cores) and increased number of processes (exception is 2 nodes with 1M and 2 M URLs).*

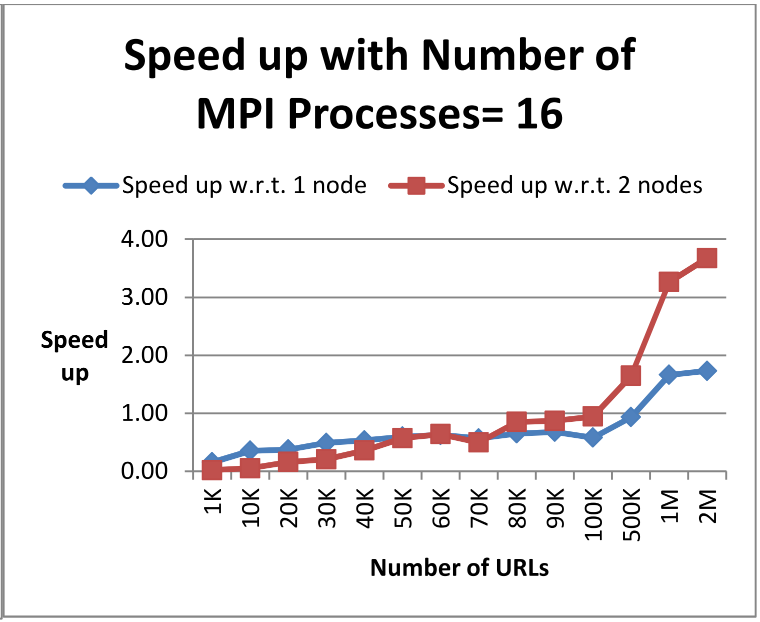
In the above graphs the speed up is maximum when Number of URLs = 50K after which the speed up decreases. This is due to the fact that as the data size increases it takes more time to transfer the data across machines (2 nodes). In case of single node the speed up almost remains flat after 50K but marginally less than the speed up obtained with 2 nodes because the computation task is spread across 2 nodes in parallel which makes it marginally faster.

When speed up is compared with respect to number of nodes, for higher number of URLs, number of nodes = 2 gives better performance. This is because the computation is distributed among more number of nodes which leads to faster calculations of Page Ranks. When the number of URLs is low, it doesn't make sense to distribute the computations among larger number of processes because most of the time is lost in communication overhead which leads to low performance. We can see that the speed up graph for 1M and 2M URLs is almost increasing.

With 1M and 2M URLs and 1 Node: The performance decreases as number of processes go beyond 8. This is due to the fact that the single machine has only 8 cores. As the number of processes goes beyond the number of cores, the additional processes keep waiting for the CPU core to get executed. With np=8, each process is executing on each core, so no process waits for the CPU and the performance is the best.

With 1M and 2M URLs and 2 Node: In this case the data set is huge and hence it becomes necessary to divide the work between different processes. Now we have 16 cores available. So if we create any number of MPI Processes less than or equal to 16, each process will get their own exclusive core to execute leading to higher performance. We should not forget the communication overhead involved between 2 nodes. When the data set is less, the communication overhead weighs more than the computation cost. Hence the performance is less for smaller data set i.e. Number of URLs < 50K. It is evident in the two graphs below:





OBSERVATIONS:

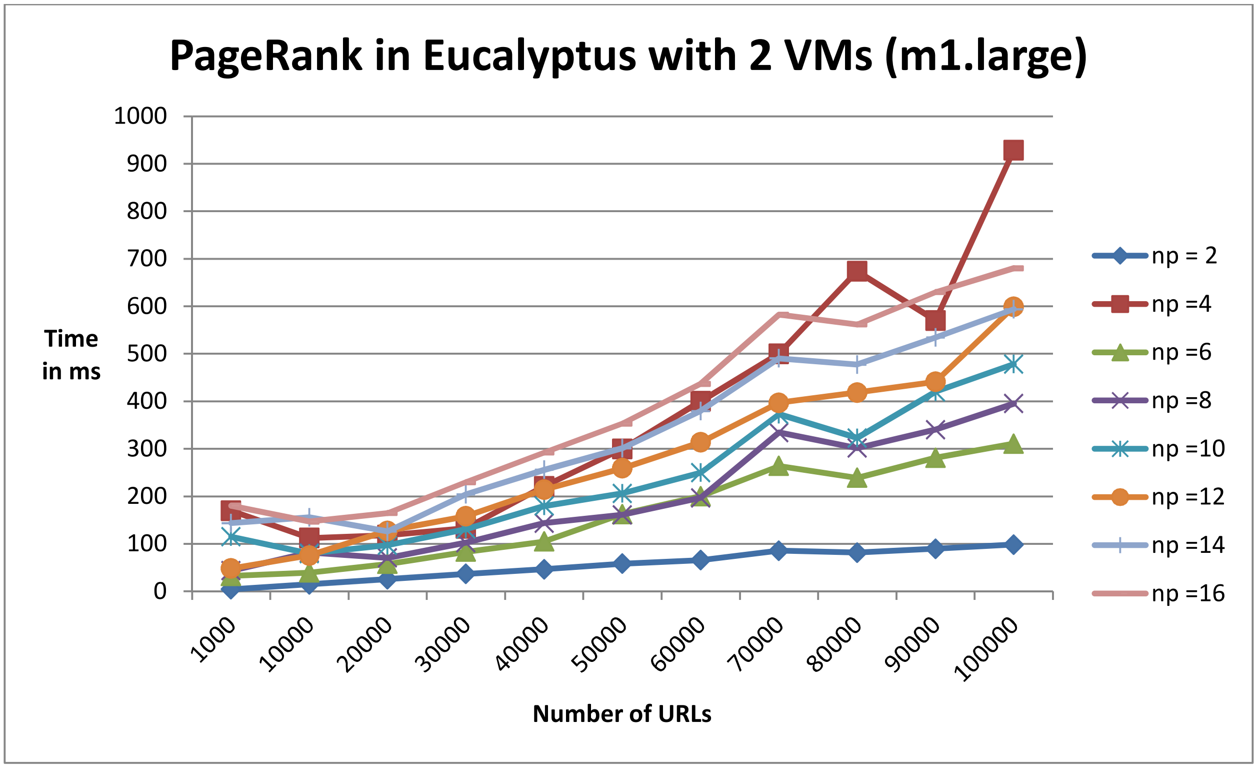
i. The Performance gain is high with *2 nodes as compared to 1 node*.

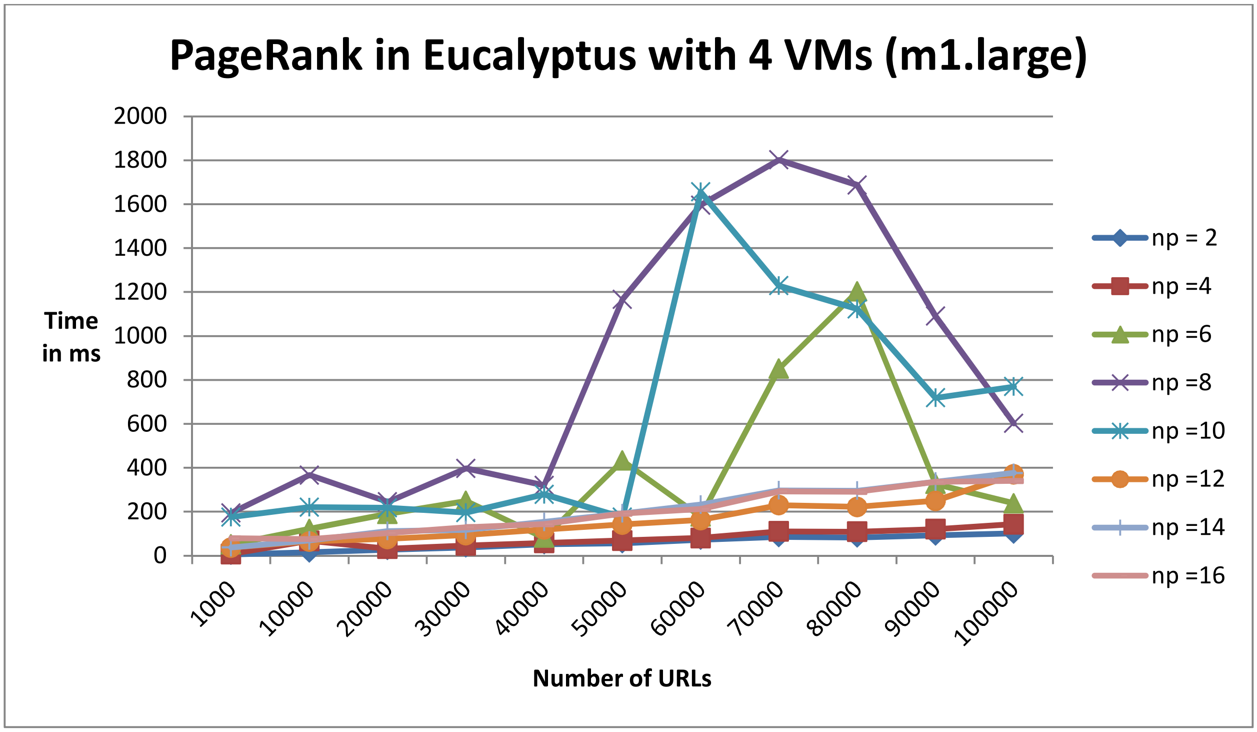
ii. If plotted graph for a single value of Number of URLs and varying number of MPI Processes, we conclude that **best performance gain is obtained when Number of MPI Processes = 8**. The Performance decreases as number of MPI Processes goes beyond 8 because of communication overheads between large numbers of processes.

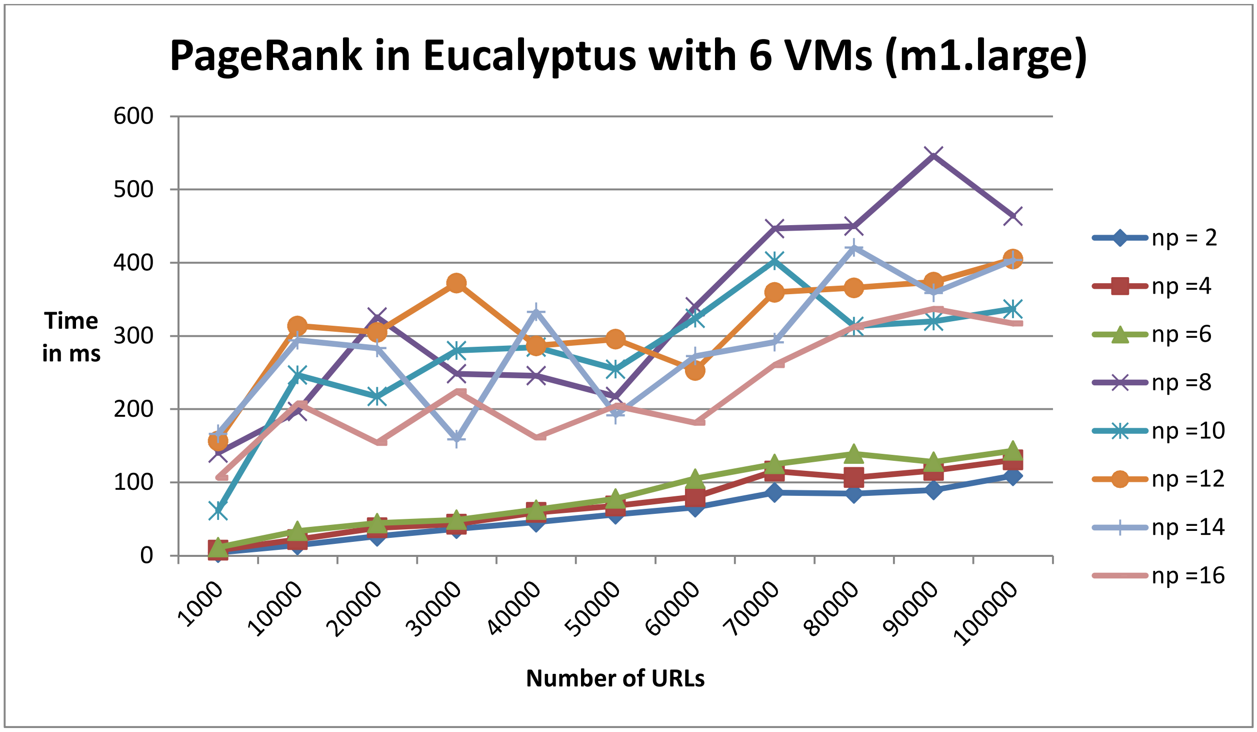
iii. As the data size increase the increase in number of processes gives a better performance. This is because the computation task for huge data is divided among several processes which execute in parallel leading to faster completion of the pagerank computation.

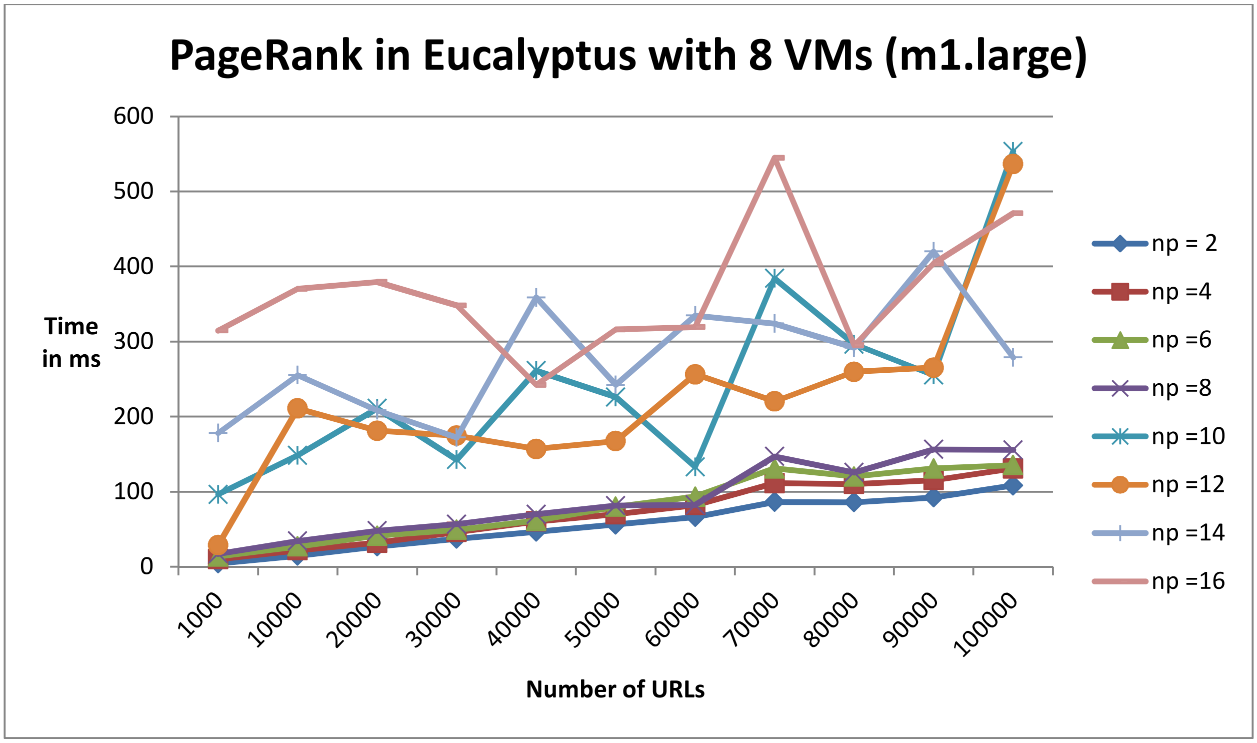
Eucalyptus Results:

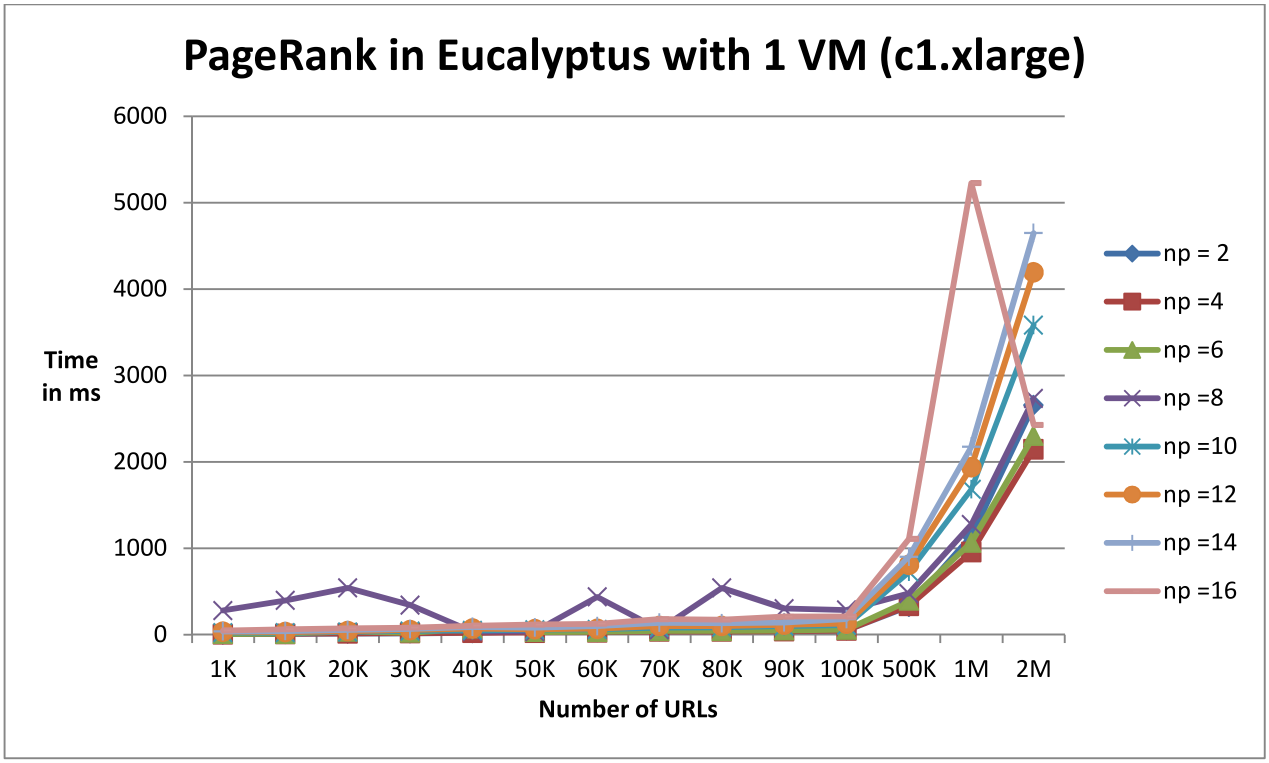
1. We ran MPI pagerank program with all the combinations of different number of VMs and different number of MPI Processes. The Graphs of all of them are shown below. The table of values can be found in the attached excel sheet (Eucalyptus Analysis.xlsx).

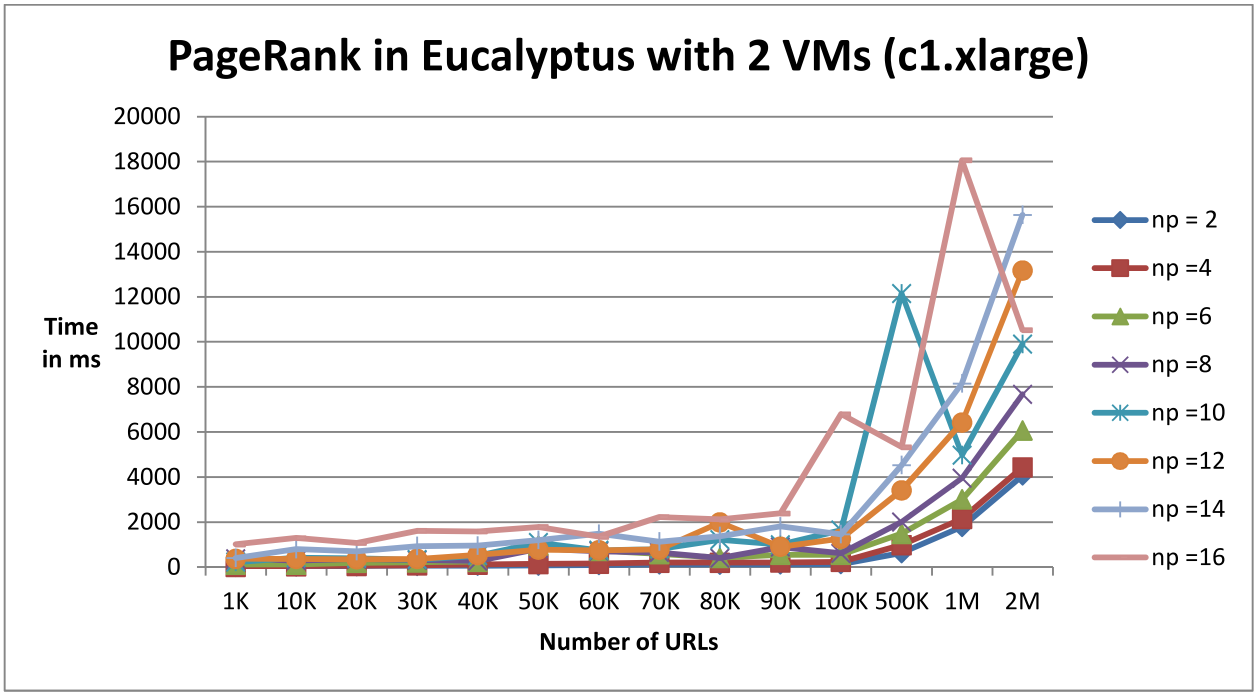

PageRank in Eucalyptus with 2 VMs (m1.large)
Time in ms
1000
900
800
700
600
500
400
300
200
100
0
1000
10000
20000
30000
40000
50000
60000
70000
80000
90000
100000
Number of URLs
np = 2
np =4
np =6
np =8
np =10
np =12
np =14
np =16


PageRank in Eucalyptus with 4 VMs (m1.large)
Time in ms
2000
1800
1600
1400
1200
1000
800
600
400
200
0
1000
10000
20000
30000
40000
50000
60000
70000
80000
90000
100000
Number of URLs
np = 2
np =4
np =6
np =8
np =10
np =12
np =14
np =16


PageRank in Eucalyptus with 6 VMs (m1.large)
Time in ms
0
100
200
300
400
500
600
1000
10000
20000
30000
40000
50000
60000
70000
80000
90000
100000
Number of URLs
np = 2
np =4
np =6
np =8
np =10
np =12
np =14
np =16


PageRank in Eucalyptus with 8 VMs (m1.large)
Time in ms
0
100
200
300
400
500
600
1000
10000
20000
30000
40000
50000
60000
70000
80000
90000
100000
Number of URLs
np = 2
np =4
np =6
np =8
np =10
np =12
np =14
np =16





Speed Up & More Clear Analysis with different URLs (*Interesting Pattern*)

If we chart out the Performance graph for different URLs and VMs we can an interesting pattern and result which are highlighted in green in the tables below.

Number of URLs = 100K

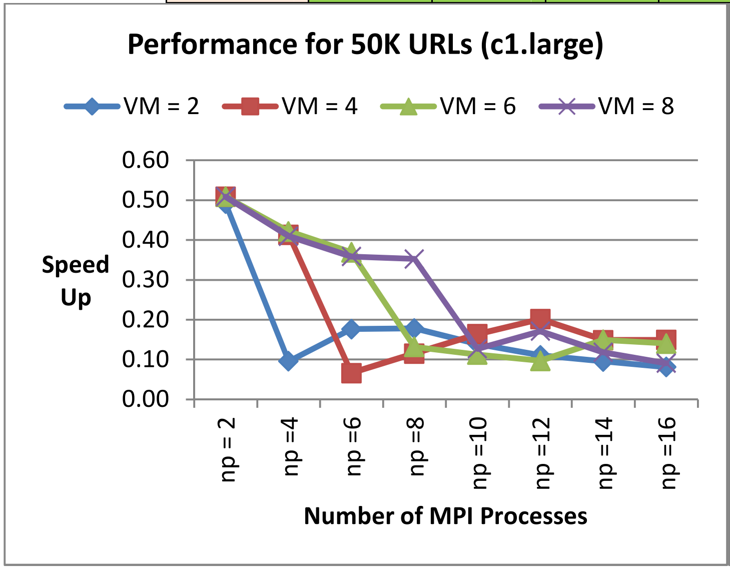
	np = 2	np =4	np =6	np =8	np =10	np =12	np =14	np =16
VM = 2	0.63	0.42	0.20	0.16	0.13	0.10	0.10	0.09
VM = 4	0.62	0.44	0.26	0.10	0.08	0.17	0.17	0.18
VM = 6	0.57	0.48	0.43	0.13	0.18	0.15	0.15	0.20
VM = 8	0.58	0.48	0.46	0.40	0.11	0.12	0.22	0.13

Number of URLs = 50K

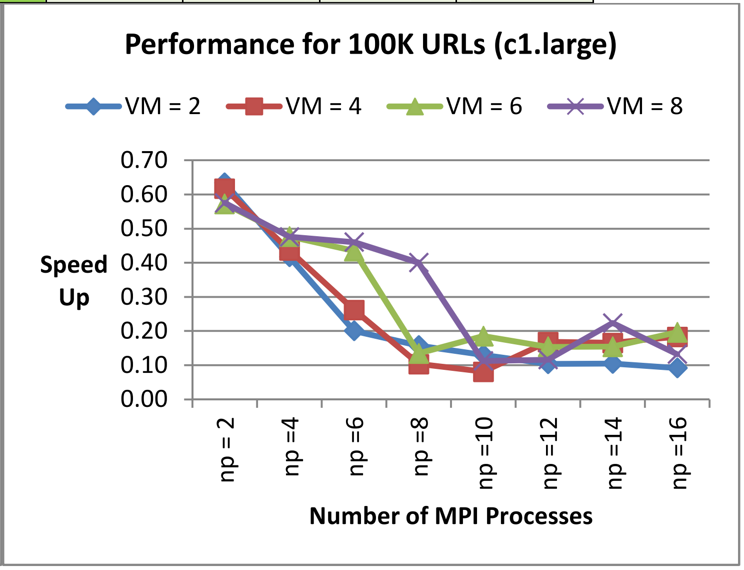
	np = 2	np =4	np =6	np =8	np =10	np =12	np =14	np =16
VM = 2	0.49	0.10	0.18	0.18	0.14	0.11	0.10	0.08
VM = 4	0.51	0.41	0.07	0.12	0.16	0.20	0.15	0.15
VM = 6	0.51	0.42	0.37	0.13	0.11	0.10	0.15	0.14
VM = 8	0.51	0.41	0.36	0.35	0.13	0.17	0.12	0.09

Number of URLs = 500K

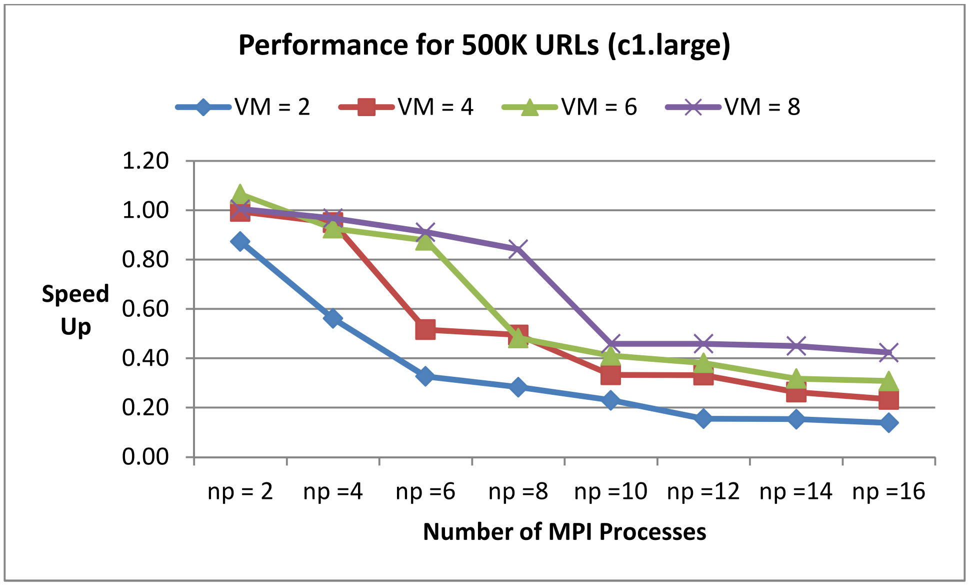
	np = 2	np =4	np =6	np =8	np =10	np =12	np =14	np =16
VM = 2	0.87	0.56	0.33	0.28	0.23	0.16	0.15	0.14
VM = 4	1.00	0.95	0.52	0.50	0.33	0.33	0.26	0.23
VM = 6	1.07	0.93	0.88	0.48	0.41	0.38	0.32	0.31
VM = 8	1.01	0.97	0.91	0.84	0.46	0.46	0.45	0.42



Performance for 50K URLs (c1.large)



Performance for 100K URLs (c1.large)



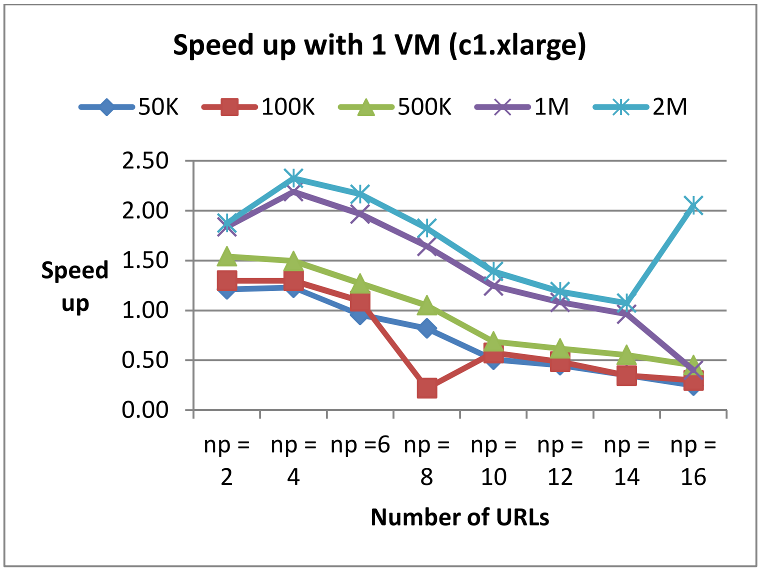
OBSERVATION: Performance is ***best when the number of MPI Processes are less than or equal to the Number of Virtual Machines***

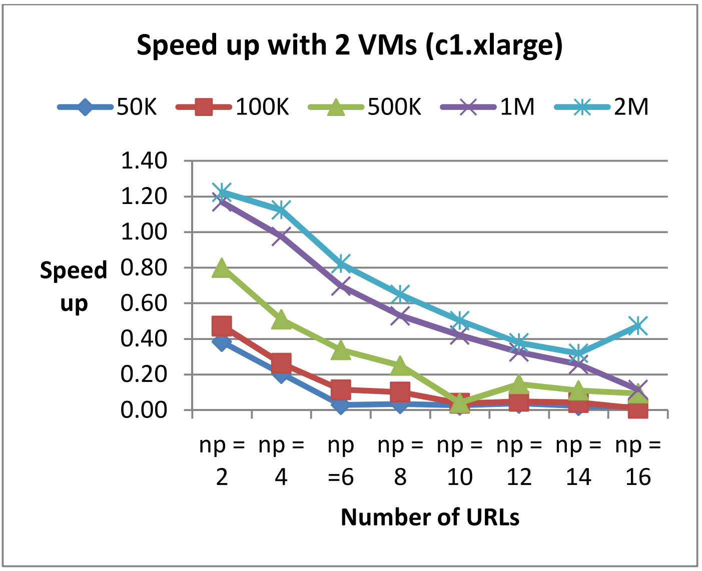
Observations and Explanation:

- The Performance is ***best when the number of MPI Processes are less than or equal to the Number of Virtual Machines***.
- The reason is that when the number of MPI processes becomes greater than Number of Virtual Machines, the communication overhead between Virtual Machines is large leading to decrease in performance.
- The MPI library assigns each process to each VM i.e. one to one, leading to better performance if the ratio of MPI Process and Virtual machines remains 1 or less.
- ***When the Number of Processes per VM is more than 1***, it increases the communication overhead and hence leads to decreased performance.

Performance Analysis with c1.xlarge

- When comparing the performances with 1 VM and 2 VMs, both on c1.xlarge, the speed up with 1 VM is higher than that of 2 VM. This is because the in case of 2 VMs there is a huge communication overhead between the VMs which is not the case with single VM. In single VM, all the processes remain in the same VM and hence there is no network communication overhead involved.
- Also, the performance decreases with the increase in number of processes. This is because as the number of processes increase, the ratio of no. of cores to number of processes decreases which makes the other processes waiting for the CPU. Also with VM there are additional processes in the system waiting for the CPU. This also proves why the sequential program performance decreases in VM by 20% - 50% as compared to Bare Metal.





Conclusion:

- **Bare-Metal**: The Performance increases till a certain point (np = 8) from number of MPI processes = 2. The performance is best when number of MPI processes = 8. After that the performance again starts degrading due to increased communication overhead which is because of increased number of processes.
- **Eucalyptus**: When the Number of Processes per VM is more than 1, it increases the communication overhead and hence leads to decreased performance. As long as the number of MPI Processes are less than or equal to the Number of Virtual Machines we get the optimal performance.

Feedback:

- Getting hands on experience with Cloud environment itself is exhilarating. We believe this project has given us both much needed insight into using the cloud resources for data intensive applications like page rank. We are thankful to our instructor Prof. Judy Qui and our Associate Instructors Stephen and Ikhyun Park for arranging all the resources and helping our way out till the completion of the project.