<u>Project1</u>- Hadoop Map-Reduce **<u>Unity ID</u> : akagrawa**

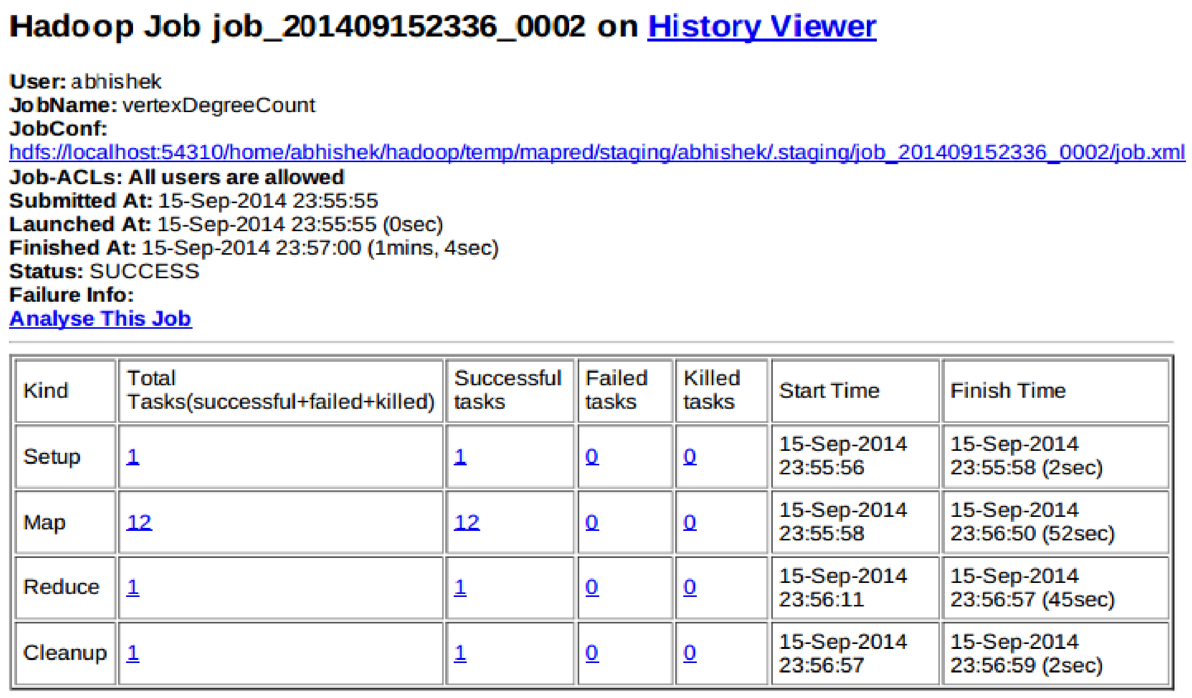
<u>Problem</u> <u>2</u>: The goal of this problem is <u>to</u> <u>make</u> <u>sure</u> <u>you</u> <u>know</u> <u>how</u> <u>to</u> <u>write</u> <u>the</u> <u>map/reduce-type</u> <u>distributed</u> <u>graph</u> <u>mining</u> <u>program</u>. You are asked to adopt the Java Hadoop program for word counts across a collection of documents provided with the Hadoop tutorial (Part 3) to the program that computes the degree of each vertex in the graph. The file format for each graph is the same as in Problem #1.

Code: Please refer README.md for the code description and commands for the execution.

Approach: Implemented a java hadoop map-reduce program to count the degree of the vertices of the given graph. With the class VertexDegree.java configured a map-reduce job which call VertexDegreeMapper.java for mapper action and VertexDegreeReducer.java for reducer action. In VertexDegree.java the key-value pair is accessed and an intermediate key-value pair is produced. This intermediate key value pair is sorted and combined using VertexDegreeReducer.java functions and from there it is written in the output folder. Single mapper and single reducer are configured to perform the task. **Not ignoring the first line of the input file** here since it will only corrupt few vertices.

Input Files: The graph data-set given for amazon. dblp and youtube is used for the performance evaluation. All the graph sizes (small, medium, large, original) are included for the processing and performance.

Job Analysis : Following is the portion of the job analysis output of the hadoop framework at port 50030. For more details please find hadoop-jobXXX-history.pdf file inside files folder.



Hadoop Job job_201409152336_0002 on History Viewer

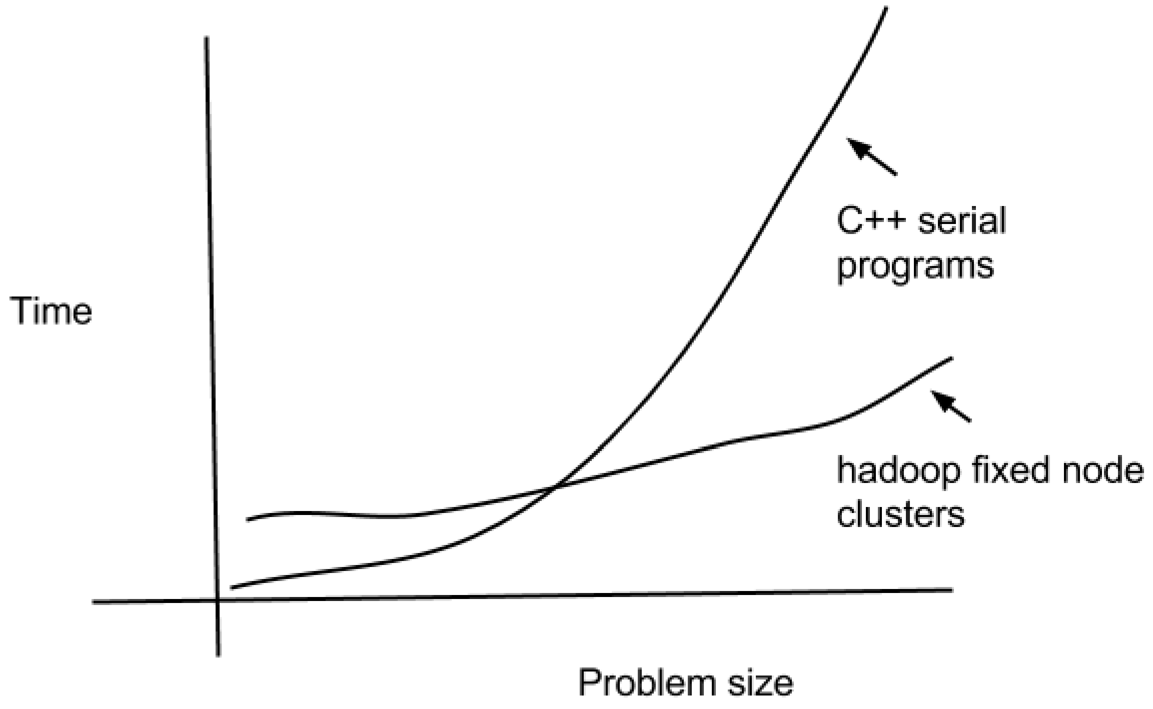
User: abhishek
JobName: vertexDegreeCount
JobConf:
hdfs://localhost:54310/home/abhishek/hadoop/temp/mapred/staging/abhishek/.staging/job_201409152336_0002/job.xml
Job-ACLs: All users are allowed
Submitted At: 15-Sep-2014 23:55:55
Launched At: 15-Sep-2014 23:55:55 (0sec)
Finished At: 15-Sep-2014 23:57:00 (1mins, 4sec)
Status: SUCCESS
Failure Info:
Analyse This Job

Kind	Total Tasks(successful+failed+killed)	Successful tasks	Failed tasks	Killed tasks	Start Time	Finish Time
Setup	1	1	0	0	15-Sep-2014 23:55:56	15-Sep-2014 23:55:58 (2sec)
Map	12	12	0	0	15-Sep-2014 23:55:58	15-Sep-2014 23:56:50 (52sec)
Reduce	1	1	0	0	15-Sep-2014 23:56:11	15-Sep-2014 23:56:57 (45sec)
Cleanup	1	1	0	0	15-Sep-2014 23:56:57	15-Sep-2014 23:56:59 (2sec)

(1) Compare the timing your Hadoop program takes using a single mapper and a single reducer versus the C++/C program in Problem #1. Do you see the differences in performance? What factors may contribute to such differences?

Computation time for C++ operation = 58.65 seconds
Hadoop performance for above operation = 1mins, 4sec

Conclusion and Inference : For above problem the performance of the C++ program and hadoop java program is comparable. But such results are also biased on many factors such as system configuration, system load at the time of execution, free memory etc.
Few points are noteworthy before such comparative analysis:
1- Hadoop framework implements map-reducible problems in an embarrassingly parallel structure. Hence multi node clusters results better as compare to single map-reduce node. Hadoop performance drastically improves adding more nodes for the same problem size.
2- C++ programs runs comparatively faster than that of java programs if they are run in a serial fashion.
3- C++ program losses its performance with serial execution and in-efficient file reading whereas hadoop program is restricted to a single node which is not meant for hadoop framework ideally.
4- Hadoop map-reduce job performs massive operations on job scheduling, synchronization and logging which is not a overhead for serial C++ program. Hence much of hadoop program time for the above job is spend on such operations making it comparable with C++.
In my view, eventually for the same problem, C++ will take more time than that of multi-node hadoop clusters. The below plot is sketched based on my understanding, I have not performed any experimental trials for the empirical proof.


C++ serial programs
Time
hadoop fixed node clusters
Problem size