Independent Study : Project 2 – Futuregrid - Summer 2013

Objective

Implement Mergesort using the SAGA-BigJob API in a distributed fashion on FutureGrid for varying data set sizes and analyze the results

Design

The Pilot API is used to reserve the resources on the machine. In the pilot description, we request for 8 cores/processes for execution. The data to be sorted is randomly generated and appended to an array. The number of subjobs is varied between (8,128). Depending on the number of subjobs for that particular iteration, the data is divided into smaller chunks of data and submitted through that subjobs to the particular core. The chunks of data are sent to the subjob file as compute unit arguments. The sierra cluster of Futuregrid is utilized for this experiment along with the teragrid redis server. Each of the smaller chunks of data are sorted during the subjob execution and stored in a file. Each subjob uses 1 core for execution. Thus, each of the files contains data that is locally sorted. The files created by the subjobs are transferred back to the pilot manager. After the completion of each of these subjobs, the pilot manager accesses these files to perform the final merging step for each of the blocks of data.

Results & Observation – Repex

The number of jobs was varied for different data sizes and the time to completion, pilot job setup time, execution time and the final merging time were recorded.

Time to completion = time for pilot job setup + time for subjob executions + final merge time

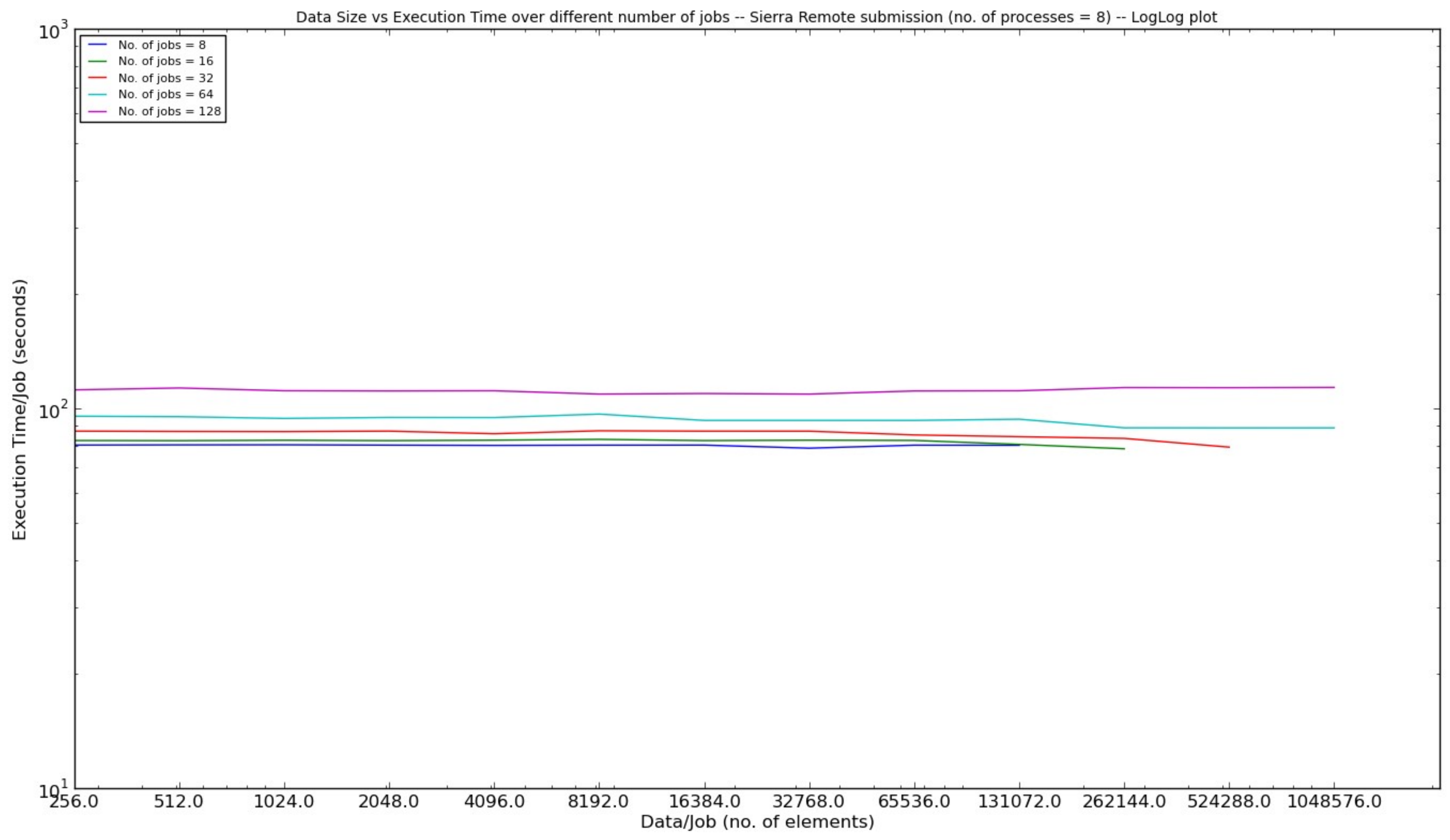
The observation from graph 1 (data size vs execution time) is that the total time to execution for all the subjobs is a factor of the number of jobs and does not have much dependence on the size of the data. For a given curve, the time for execution remains almost the same for small as well as big sizes of data sets (no. of elements). So depending on the resource availability and time constraints, the number of jobs can be decided and the data processing can be done.

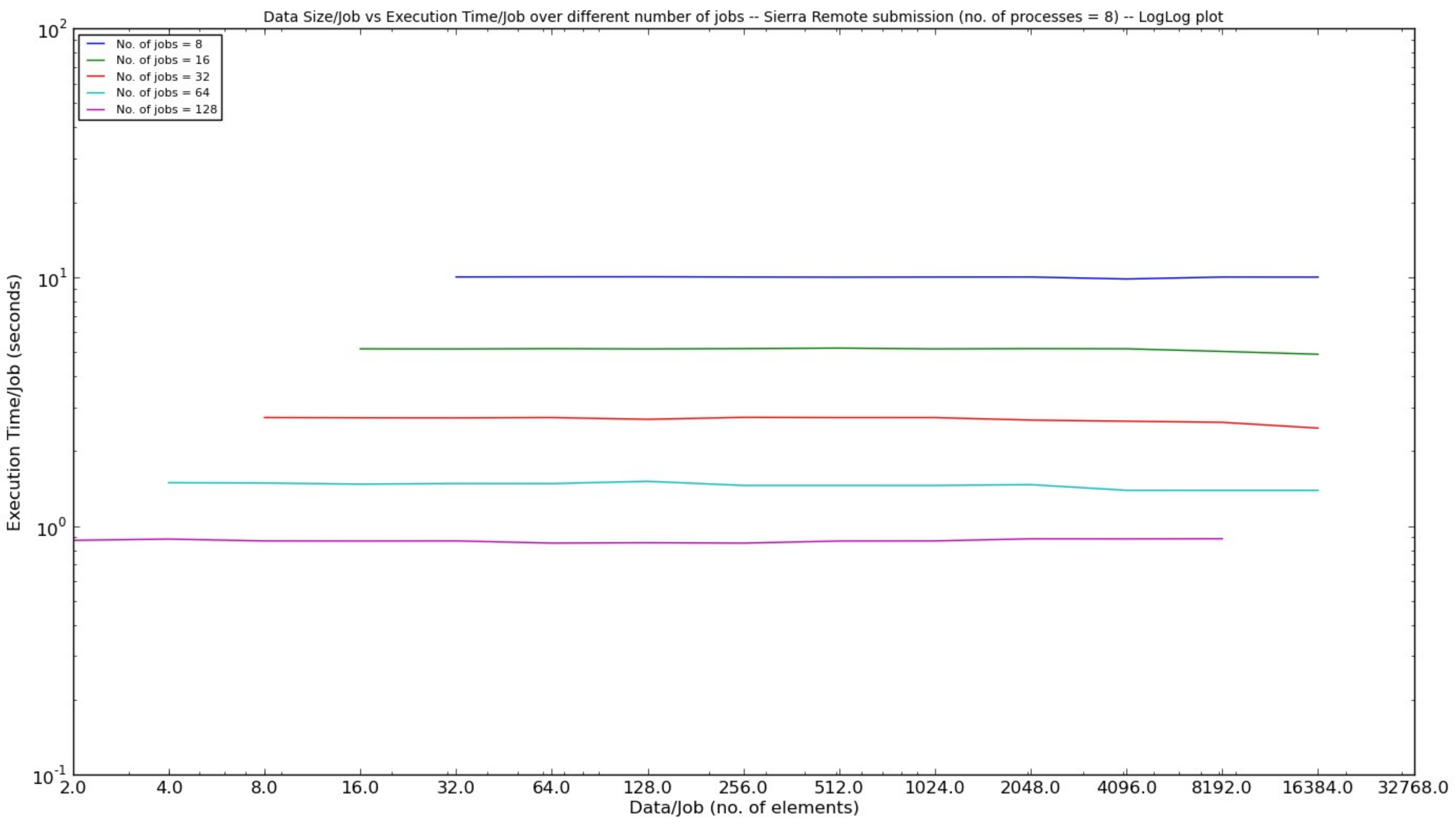
The observations for graph 2 (data size per job vs execution time per job) were calculated by dividing the total execution time and data size per reading by the number of jobs used for that reading. As expected, for a given chunk size of data, the execution time decreases as we increase the number of jobs.

It was also observed that the total time to completion (not shown in the graph) was dominated by the execution time for low values of number of jobs and dominated by the execution time + final merge time for high values of number of jobs. The pilot job setup time remained the same throughout all the set of experiments.

The conclusion from this set of experiments is that the total time to completion is a factor of the subjob execution time and final merge time which are both seem to be a function of the number of jobs utilized.

The data chunk size was limited to 16384 as that seems to be the limit of the size of the arguments of the compute unit description.


Data Size vs Execution Time over different number of jobs -- Sierra Remote submission (no. of processes = 8) -- LogLog plot
No. of jobs = 8
No. of jobs = 16
No. of jobs = 32
No. of jobs = 64
No. of jobs = 128
Execution Time/Job (seconds)
10^3
10^2
10^1
256.0
512.0
1024.0
2048.0
4096.0
8192.0
16384.0
32768.0
65536.0
131072.0
262144.0
524288.0
1048576.0
Data/Job (no. of elements)



Data Size/Job vs Execution Time/Job over different number of jobs -- Sierra Remote submission (no. of processes = 8) -- LogLog plot
No. of jobs = 8
No. of jobs = 16
No. of jobs = 32
No. of jobs = 64
No. of jobs = 128
Execution Time/Job (seconds)
10^{2}
10^{1}
10^{0}
10^{-1}
2.0
4.0
8.0
16.0
32.0
64.0
128.0
256.0
512.0
1024.0
2048.0
4096.0
8192.0
16384.0
32768.0
Data/Job (no. of elements)