**COMP90024 Assignement1 Analysis**

To run the program with efficient parallelizing and distributing the program tasks across multiple nodes and its CPU process, we used MPI (message passing interface) class in mpi4py[S1] python module in the code as given below . The code was run using slurm as workload and scheduler with various CPU and memory allocation in SPARTAN HPC environment.

comm = MPI.COMM\_WORLD # creates a communicator to all process in the MPI job

rank = comm.Get\_rank() #rank of current process in the communicator

size = comm.Get\_size() #total number of processes in the communicator

The sbtach command was used to submit the batch script to slurm as shown below

#!/bin/bash

#SBATCH --account="COMP90024" #Slurm account to be charged for resource usage.

#SBATCH --job-name="TESTCASE2" # Job name

#SBATCH --nodes=1 # Number of nodes required

#SBATCH --cpus-per-task=8 # Number of CPU cores per process (tasks) per node

#SBATCH --time=1:00:00 # Maximum runtime

#SBATCH --mail-user=sthanadanfra@student.unimelb.edu.au

#SBATCH --mail-type=ALL

module purge # unload all loaded modules to free memory.

module load foss/2022a Python/3.10.4 SciPy-bundle/2022.05 #modules (FOSS toolchain, Python 3.10.4, and SciPy-bundle) needed

srun -n 8 python shsa\_comp90024\_as1\_v1.86.py #Executing the script using process needed for parallel tasks launched

my-job-stats -a -n -s # monitor the job status

The above script was used to test all the scenarios in the assignment objective. However, it was noticed that the code was using ~ 164.48 MB/CPU to process the given 120GB twitter json file. Hence the memory percentage utilization was observed as 4.11% as ~3.91GB was allocated to each CPU by default by slurm in SPARTAN.

To test the efficiency of the memory usage (>80% ), The slurm script was additionally run with explicit parameter for memory allocation as given below (with allocation of 200MB per cpu).

#SBATCH --mem-per-cpu=200M # Memory allocation per cpu core

S1.https://mpi4py.readthedocs.io/en/stable/mpi4py.html

SLURM Job analysis

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| No Of Nodes | No of CPUs/Node | Job Time  (code run)  in secs | Core-walltime | CPU Efficiency | Total Memory Allocation across Core(s) | Total Memory Utilized  (Estimated Max) | Reported Memory Efficiency |
| Job Run 1 : with default memory allocation per CPU (3.91GB) | | | | | | | |
| 1 | 1 | 1854.812 s | 00:30:59 | 99.46% | 3.91 GB | 164.48 MB | ~4.11% |
| 1 | 8 | 672.977 s | 01:30:16 | 97.80% | 31.25 GB  (3.91 GB/core) | 1.31 GB | ~4.20% |
| 2 | 4 | 678.497 s | 01:31:04 | 99.01% | 31.25 GB  (3.91 GB/core) | 1.31 GB | ~4.19% |
| Job Run 2 : with custom memory allocation (200MB) per CPU (using #SBATCH --mem-per-cpu=200M) | | | | | | | |
| 1 | 1 | 2069.848 s | 00:34:33 | 99.42% | 200.00 MB | 164.48 MB | 82.24% |
| 1 | 8 | 591.488s | 01:19:28 | 99.01% | 1.56GB  (200.00 MB/core) | 1.56 GB | 85.53% |
| 2 | 4 | 937.432 s | 02:05:36 | 98.96% | 1.56GB  (200.00 MB/core) | 1.56 GB | 84.08% |