Project 2: PageRank Implementation Using MPI and Map/Reduce

Aim:
The aim of Project 2 is to implement the PageRank algorithm using the MPI (Message Passing Interface) framework via the mpi4py library on the SeaWulf high-performance cluster. The goal is to calculate and identify the top 10 webpages with the highest PageRank values from the dataset located in /gpfs/projects/AMS598/projects2025_data/project2_data/

The implementation uses the taxation method with a damping factor β = 0.9, simulating realistic web navigation behavior by redistributing part of the rank uniformly across all nodes. The computation follows a Map/Reduce structure and is executed in four iterations, during which intermediate results are saved in a dedicated personal directory under /gpfs/projects/AMS598/class2025/Deshpande_Gunjan/.

The program is configured to run with five parallel processes, but is designed to flexibly handle any arbitrary number of MPI processes. This ensures scalability and adaptability for various computational environments while maintaining accurate PageRank computation and output consistency.

Code Logic:

1. **Script file**
   The code begins by loading essential modules (mpi4py, defaultdict, json, and os). The `load_graph_data()` function reads multiple input files representing directed web links between pages. Each process contributes to building a complete adjacency list, mapping each node (webpage) to its outgoing links. The root process aggregates and broadcasts this adjacency structure to all nodes via MPI communication. Intermediate information files are stored in the directory /gpfs/projects/AMS598/class2025/Deshpande_Gunjan.

2. Parameter Setup
   Key parameters include a damping factor (β = 0.9) and four Map/Reduce iterations, as required. Each node's initial PageRank is initialized as 1/N1/*N*, assuming equal probability distribution, where N*N* is the total number of nodes.

3. Map/Reduce Execution
   Each process maps a subset of nodes and distributes their rank contributions proportionally across outgoing links. After every iteration, the ranks are normalized, intermediate values are saved, and the updated PageRank table is broadcast cluster-wide. At the end of the fourth iteration, the top ten nodes with the highest ranks are identified and saved in both .txt and json formats.

4. Output and Result Generation
   Each iteration produces a textual summary of rank convergence. The final result

includes `pagerank_results.json` and `pagerank_final.txt` files containing the ten webpages with the highest scores, demonstrating which pages have the strongest inbound network influence.

SLURM Script Logic:

The file `project2.slurm` automates the distributed run of the PageRank MPI program. It defines essential SLURM job parameters, resource allocation, and module setup for execution. It loads required software modules: `anaconda/2 for the Python environment mvapich2/gcc/64/2.2rc1 for MPI support` and `installs mpi4py`

After setup, the script runs the main command

```
mpirun -np 5 python pagerank_project2.py
```

```
mpirun -np 5 python
/gpfs/projects/AMS598/class2025/gdeshpande/PProject2fol/pagerank_proje
ct2.py
```

This initializes five parallel MPI processes where each process executes one partition of the dataset. SLURM manages process scheduling and task communication.

Execution Command:

```
# Navigate to working directory

cd /gpfs/home/gdeshpande/PProject2fol

# Submit the job

sbatch project2.slurm

# View job output

cat
/gpfs/projects/AMS598/class2025/Deshpande_Gunjan/pagerank_top10_results.tx
t

or

cat project2_pagerank_JOBID.out

# View JSON format results

cat /gpfs/projects/AMS598/class2025/Deshpande_Gunjan/pagerank_results.json
```

Result:

Earlier, I was working on improving my Python script and Slurm file, but unfortunately, the terminal was not working, so I could not fix the errors in the iteration files also the job get cancelled after 30 mins and output file wont be created, I wanted to fix this issue but could not fix it as terminal was not working, I actively communicated with the support team as well and created ticket for the same. However, I was able to get the following results in my iteration files and output files:

ID	PageRank
1002618	91.96336171288576
5625	73.53788331803494
1003440	69.21855751969518
2008709	61.61873786808535
1004077	59.04901042191172
2004766	58.93966248056534
1007650	56.588633628983125
1003509	56.315263205404996
1003817	56.09655766224834
8396	53.47216039127785

When the terminal was working I made changes in the parallel reducer logic: `comm.allgather(local_contribs)`, ensuring full parallelism during the reduce phase so all ranks process their assigned contributions independently. The dataset restrictions were removed from `load_graph_data()`—the slicing of paths[:3] and the `max_lines_per_file` = 500000 optimization eliminated.

I need to do few changes in the PageRank implementation: which leverages true MPI Map/Reduce parallelism in both mapper and reducer stages, and processes the entire link graph for correct top-10 rankings. I tried my best to run the Python and Slurm files to the best of my knowledge and get the following output, my MPI initialization is failing due to OpenFabrics and srun errors but was not able to fix it as nodes were not available. However, there are still some errors in the iteration text files, and since the terminal was not working and nodes were drained and reserved for other tasks, I could not resolve them.

ID	PageRank
2817335	3.138694872e-05
9423027	3.074288557e-05
3465571	3.062723544e-05
6813130	2.956257441e-05
8065384	2.933583003e-05
5132918	2.929537228e-05
1460753	2.871062484e-05
4159429	2.575197964e-05
7495117	2.262181180e-05
878533	2.133152760e-05