Report

2013EE10462 (J. Shikhar Murty)
2013CS50278 (Anmol Mahajan)

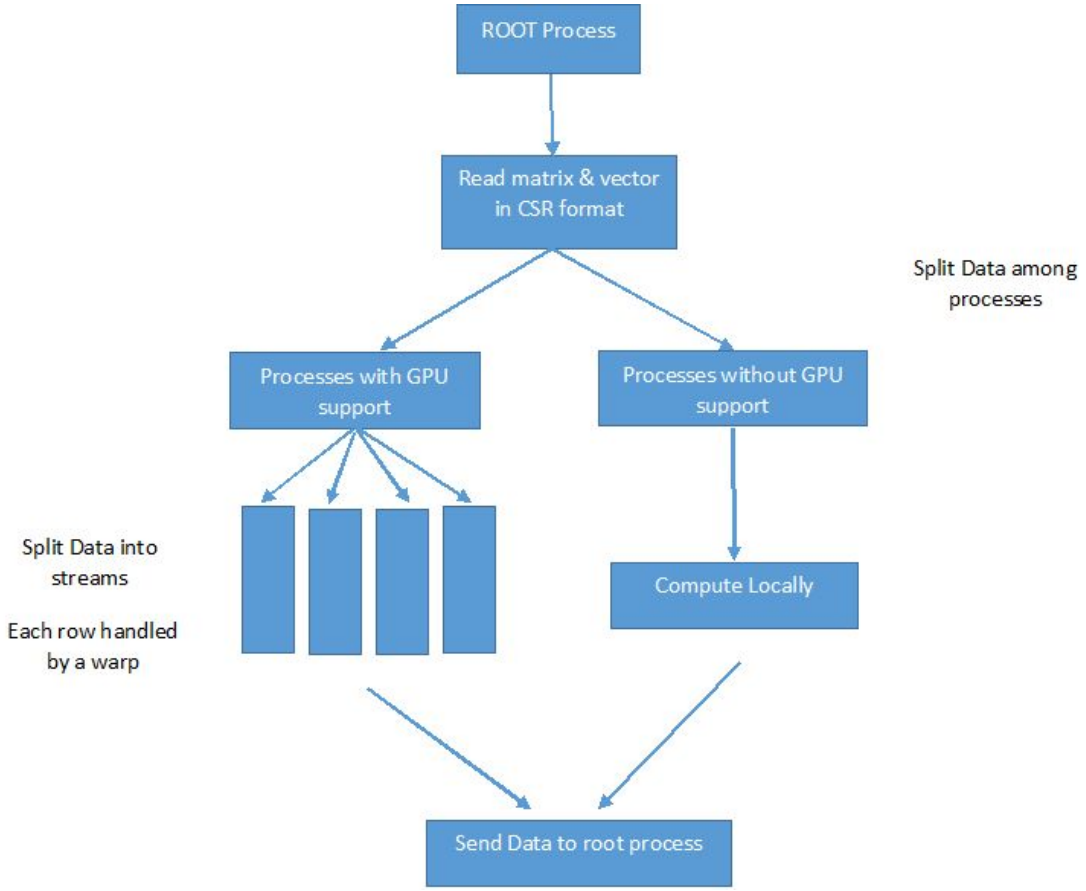
In this project, we implemented an efficient sparse matrix-vector multiplication algorithm using MPI and CUDA.

Details:-

1. MPI was used to distribute computation among the nodes.
2. The nodes used may or may not have GPU compute capability.
3. This was implemented by splitting the processes into 2 communicators- one where processes are scheduled on nodes with GPU compute capability and others scheduled on nodes without GPU compute capability.
4. Processes that can use GPU were given greater workload.
5. The matrix was represented in CSR format.
6. To reduce the data sent to processes, we represent the vector in CSR format. SO only the elements corresponding to non-zero entries of the matrix were sent instead of the entire vector.

LOAD BALANCING STRATEGIES:-

- More workload was given to the processes with GPU compute capability compared to processes without this capability.
- Equal number of rows were given to each block of threads.
- Each row was handled by a warp.



PARALLELIZATION :-

- To extract more parallelization from the GPU, we divide the workload given to every GPU into multiple streams. These streams execute asynchronously, and hence we can use them to overlap GPU I/O with compute. We set the number of streams to 4 for each device.
- We use MPI to distribute the workload among multiple processes

DESIGN DECISIONS :-

- Kernel optimizations: The kernel is designed such that every warp computes a row. This is done to ensure that we can use shared memory without having to deal with race conditions since warps execute in locksteps.
- Variable work load for GPU based on ratio of cpus to gpus