COL380: Introduction to Parallel & Distributed Programming

Lab 4

Problem Statement

Perform a matrix-vector multiplication for this lab. Assume that the matrices are sparse and square. You can mix & use MPI, OpenMP and CUDA to handle the problem. Note that this lab can be performed in **groups of at most 2.**

Sparse matrices are matrices in which most elements are zero. To save space and running time it is critical to exploit this feature. There are many efficient storage structure options for sparse matrices, which can be explored.

Let A be a n × n matrix and B be a n x 1 vector.

The product C = AB is defined as follows: $c_{ij} = \sum_{k=1}^{n} a_{ik}b_{kj}$, for $1 \leq i, j \leq n$.

Note: The number of operations can be reduced by avoiding the computation of products $a_{ik}b_{kj}$ for which $a_{ik} = 0$ or $b_{kj} = 0$.

Compiling the code

1. One way is to put all the code (MPI, OpenMP and CUDA) in a single file

   Compile using nvcc, which internally uses gcc/g++ to compile your C/C++ code, and link to your OpenMP and MPI library:

   ```
   nvcc -I/usr/mpi/gcc/openmpi-1.4.6/include -L/usr/mpi/gcc/openmpi-
   1.4.6/lib64 – lmpi mpicuda.cu –o mpicuda
   ```

   (Modify for the version of MPI that you are using)

2. The other way is to have MPI, OpenMP and CUDA code separate in two files

   ```
   module load openmpi cuda #(optional) load modules on your node
   mpicc -c mpi.c -o mpi.o
   nvcc -c cuda.cu -o cuda.o
   ```

```
mpicc mpi.o cuda.o -lcudart -L/apps/CUDA/cuda-5.0/lib64/ -o mpicuda
```

(Modify for the version you are using)

Statement of work

The assignment can be done in groups of at most 2.

1. Implement a solution to Sparse Matrix Multiplication problem using MPI+CUDA.
    a. Input: signed integer (ie *int* not long integers or int64) matrices A and B in a text file, in the following format

    Sample_Input.txt

    ```
    Name: Sample_Input
    Dimension: 5
    A
    0  1  0  0  5
    2  0  0  0  3
    3  3  0  0  0
    0  0  0  0  0
    0  0  0  4  0
    B
    0
    4
    0
    0
    0
    ```

    b. Output: Multiplied Matrix, in the following format

    Output_Sample_input.txt

    ```
    Dimension: 5,1
    4
    0
    12
    0
    0
    ```

2. Use of both MPI and CUDA is compulsory. Skipping any one will lead to a heavy penalty. Use of Openmp, however, is optional.
3. Along with implementation, prepare a lab report (.pdf)
    a. Provide a well formed and elaborate parallel algorithm and a block diagram of your solution.

b. Additionally, clearly explain your design decisions, parallelization strategy, load-balancing strategy. Make sure that explanations are bulleted and are not long paragraphs.

4. Zip the implementation and report together and name the zip with your entry nos.

Testing and Running on HPC

No GPU support on your personal system. *Fret Not!*

Use High Performance Computing at IITD (http://supercomputing.iitd.ac.in/).

***Word of caution**:* Be judicious and responsible when using HPC resources.

You must **not** use at any point more than 2 CPU nodes (each node has 24 cores) and 1 GPU node (containing 2x Nvidia K40 GPU cards, each card having 2880 *CUDA cores).*

To use HPC:

1. Request for an account on HPC
    a. Use your LDAP credentials
    b. Enter

        Faculty supervisor (uid) :

        Requested Expiry date of access: 1 May 2017
2. ssh into HPC using your LDAP credentials

    ```
    ssh <entry_no>@hpc.iitd.ac.in
    ```

3. Copy your files to your HPC client node sing scp

    ```
    scp ./<file> <entry_no>@hpc.iitd.ac.in:
    ```

4. Write PBS script to submit a job. Follow http://supercomputing.iitd.ac.in/?pbs
    a. Keep a wall time of 30 min.
5. Submit using qsub

    ```
    qsub <script> -q low
    ```

6. Check status of your submitted job using qstat

    ```
    qstat <job_id>
    ```

Tips and Tricks

- Explore an efficient way of managing sparse matrices.
- You can put both MPI and CUDA code in a single file. The downside of this approach is that it might end up being a plate of spaghetti, if you have some seriously long program.
- You can explore cuda-aware openmpi.
- Remember, sending to and receiving from GPU is a costly affair.
- Think, how to make best use of MPI and CUDA. Exploit their strengths don't let anyone of them become a bottleneck.

Grading Policy

Grading of the assignment would be done based on a competitive scoreboard.

Your submission would go directly to our server and you will be able to see where you stand relative to your class. Scoring on this assignment is going to be relative.

1. Correct execution for at least $n=2^{10}$ values: 35 Marks
2. Scalability and efficiency 0-50 Marks
3. Clarity in report and discussion of your solutions: 15 Marks