CSCE 435 Group project

1. Group members:

1. Connor Nicholls
2. Noah Thompson
3. Cyril John
4. Tarun Arumugam

---

2. Primary mode of communication

Discord/GroupMe

2. *due 10/25* Project topic

We have chosen the example topic, which is: "Choose 3+ parallel sorting algorithms, implement in MPI and CUDA. Examine and compare performance in detail (computation time, communication time, how much data is sent) on a variety of inputs: sorted, random, reverse, sorted with 1% perturbed, etc. Strong scaling, weak scaling, GPU performance."

2. *due 10/25* Brief project description (what algorithms will you be comparing and on what architectures)

- Enumeration Sort (MPI + CUDA)
- Enumeration Sort (MPI on each core)
- Odd-Even Transposition Sort (MPI + CUDA)
- Odd-Even Transposition Sort (MPI on each core)
- Parallel Merge Sort (MPI + CUDA)
- Parallel Merge Sort (MPI on each core)

All credit for pseudocode goes to https://www.tutorialspoint.com/parallel_algorithm/parallel_algorithm_sorting
Enumeration Sort Pseudocode: procedure ENUM_SORTING (n)

begin for each process P1,j do C[j] := 0;

for each process Pi, j do

```
if (A[i] < A[j]) or A[i] = A[j] and i < j) then
   C[j] := 1;
else
   C[j] := 0;
```

for each process P1, j do A[C[j]] := A[j];

end ENUM_SORTING

Odd-Even Transposition Sort

procedure ODD-EVEN_PAR (n)

begin id := process's label

for i := 1 to n do begin

```
if i is odd and id is odd then
   compare-exchange_min(id + 1);
else
   compare-exchange_max(id - 1);

if i is even and id is even then
   compare-exchange_min(id + 1);
else
   compare-exchange_max(id - 1);
```

end for

end ODD-EVEN_PAR

Parallel Merge Sort

procedureparallelmergesort(id, n, data, newdata)

begin data = sequentialmergesort(data)

```
for dim = 1 to n
   data = parallelmerge(id, dim, data)
endfor
```

newdata = data end