**PS8 Questions**

**To-Do 5 (ring.cpp):**

****

**Norm**

\* Question 1: What is your code for `mpi\_norm`? (Cut and paste the code here.)



(I also made a modification where I used Scatter to distribute the vector in the main function):

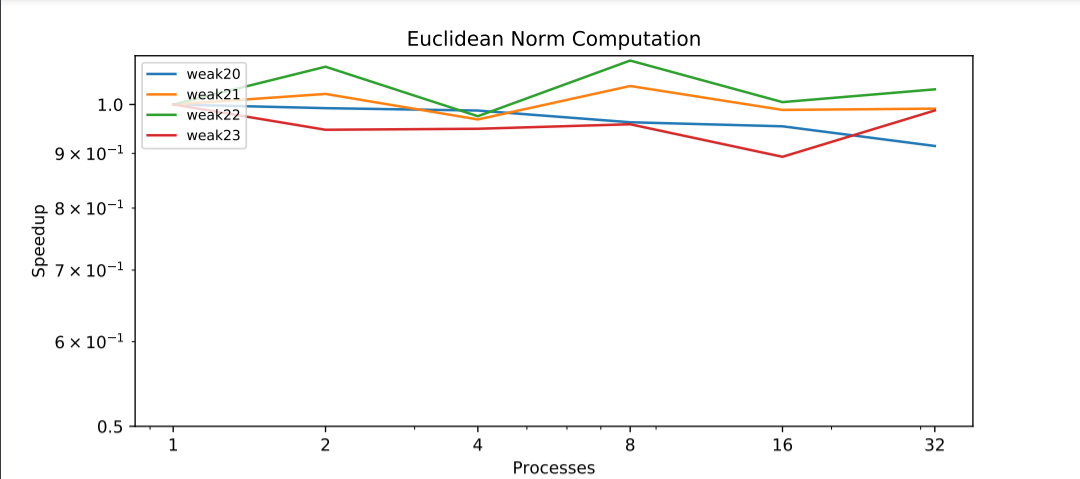


**strong.pdf for mpi\_norm.exe:**

**Chart, line chart

Description automatically generated**

**weak.pdf for mpi\_norm.exe:**

****

\* Question 2: Per our discussions in lectures past about weak vs strong scaling, do the plots look like what you would expect? Describe any (significant) differences (if any).

Answer: For strong.pdf, the plot looks as expected. We see that as more processes added, the speedup we get is strongly scaled with the number of processes we have which is expected for strong scaling. For weak.pdf, the plot also looks as expected. We see that as more processes added, for all the problem sizes, the speedup is consistent around 1 which is expected for a weak scaling.

\* Question 3: For strong scaling, at what problem size (and what number of nodes) does parallelization stop being useful? Explain.

Answer: From the plot we see that for all the problem sizes, the speedup scales almost consistently to the number of process added. This implies that adding more processes can be still useful in terms of performance speedup for all the problem sizes we tested by looking at the plot.

**Grid**

\* Question 4: What is your code for halo exchange in `jacobi`? (Cut and paste the code here.) If you used a different scheme for extra credit in `mult`, show that as well.



Q4 cont. (For **extra credit**, I have used a different approach for mult() in mpiStencil.hpp):



\* Question 5: What is your code for `mpi\_dot`? (Cut and paste the code here.)



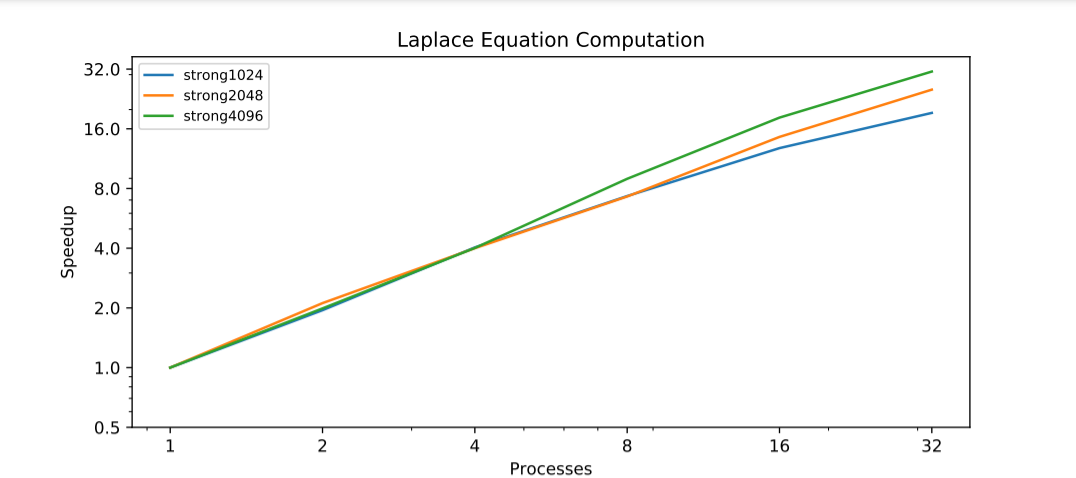
\* Question 6: What changes did you make for `ir` in ` mpiMath.hpp `? Copy and paste relevant code lines that contain your edits to your report. Provide comments in the code near your edits to explain your approach.



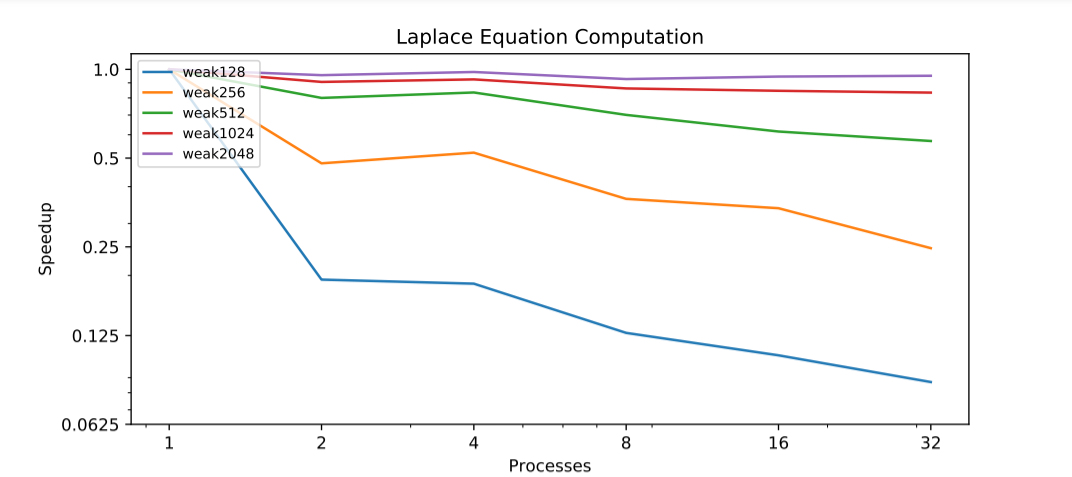
\* Question 7: (583 only) What changes did you make for `cg` in `mpiMath.hpp`? Copy and paste relevant code lines that contain your edits to your report. Provide comments in the code near your edits to explain your approach.

Answer: This is for Amath 583 only, I am from Amath 483.

**strong.pdf for grid.exe:**

****

**weak.pdf for grid.exe:**

****

\* Question 8: Per our discussions in lectures past about weak vs strong scaling, do the plots look like what you would expect? Describe any (significant) differences (if any).

Answer: For strong.pdf, the plot looks as expected. For strong scaling, we see that as more processes added, the speedup we get is strongly scaled with the number of processes we have which is expected for strong scaling. For weak.pdf, the plot also looks as expected. For weak scaling, we see that as the problem size increases, with more processes added, the speedup is more and more close and consistent to be 1 which is expected for weak scaling. For smaller problem sizes such as N = 128, as more processes added, we see the speedup dropped. Because the problem size is so small that the time took to add a new process surpasses the benefits of speedup brought by distributing the problem to pieces. This is reasonable and as expected.

\* Question 9: For strong scaling, at what problem size (and what number of nodes) does parallelization stop being useful? Explain.

Answer: For problem size N = 1024, we see a turning point when the process number equals to 10, after that we see a drop in the speedup as more processes added. This implies that for N = 1024, after 10 processes, adding additional process stop being useful.

For problem size N = 2048, we see a turning point when the process number equals to 16, after that we see a drop in the speedup as more processes added. This implies that for N = 2048, after 16 processes, adding additional process stop being useful.

For problem size N = 4096, there is not an obvious turning point from the plot. We see that the speedup increased by adding more processes is generally consistent from 1 process to 32 processes. So, from what is provided by our plot, adding more than 32 processes for problem size N = 4096 can still be useful.