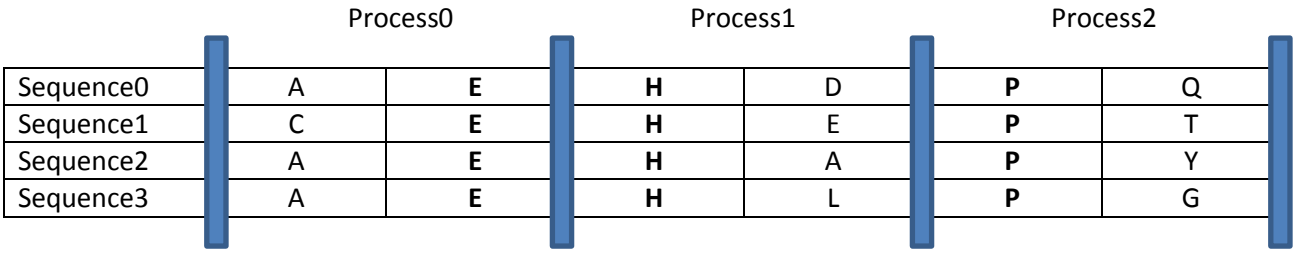
Hyung Yul Choi

Parallelizing the Search for Primers Given a Multiple Sequence Alignment

For this final project, I am building a program that given an input of multiple sequence alignment (MSA), it will be able to find regions of the sequences that are called primers. A Multiple Sequence Alignment is a sequence alignment of three or more biological sequences. From that MSA, it will determine appropriate primers that can be used in the polymerase chain reaction (PCR) method of gene amplification. The polymerase chain reaction is a technology in molecular biology used to amplify a single or a few copies of a piece of DNA which then generates thousands to millions of copies of a particular DNA sequence. This is useful for DNA cloning for sequencing, DNA-based phylogeny, or functional analysis of genes. In order to do a PCR, primers must be found among the MSA. Primers are sub-sequences that are used as templates for gene amplification

It is possible for there to be multiple "candidates" for primers. To be a good candidate, the primer should have nucleotides that are consistent in a region of a sequence for all of the organisms that are being compared. In other words, given a MSA, a candidate primer is a sequence of a given length determined by the user that consists of nucleotides that are the same for all the sequences in the MSA in a given region. Searching for areas of commonality between sequences can take a very long time due to their lengths. In addition, it is possible to have a sequence alignment of hundreds of biological sequences. Therefore, scalability of the problem is an issue.

The obvious part of the problem that can be parallelized is the searching for areas of commonality. I plan to do this by dividing up the MSA evenly between each process. The root will create a new sub-array containing the appropriate number of sequences and will MPI_Scatter each of the sub-arrays to its appropriate process number. Each process will then index through its sub-array and does a simple equality check of whether the nucleotide in a particular index is the same for all of the sequences. If an index of commonality is found, it adds that index into another local array. Once each process has completed its job, the root will MPI_Gather all of the indexes into a global array that will contain all of the indexes of the sequence where the nucleotides in all sequences are equal. MPI_Reduce is used for collecting the total sum of the number of indexes found by each process.



With this information, the root can then simply find a potential primer by searching for indexes that are contiguous for a certain length.

Data:

- Tests were done on Jupiter
- All tests were done with 4 sequence MSA but with varying lengths of sequence

	4 Sequences	100000 Nucleotides			
	1	2	4	8	12
Time	0.047275	0.023777	0.016955	0.014137	0.009642
Speedup		1.98826597	2.78826305	3.34406168	4.90302842
Efficiency		0.99413299	0.92942102	0.83601542	0.61287855

	4 Sequences	200000 Nucleotides			
	1	2	4	8	12
Time	0.10717243	0.05390246	0.03843699	0.03204858	0.02185841
Speedup		1.98826597	1.40235919	1.19933508	1.46618959
Efficiency		0.99413299	0.3505898	0.14991688	0.12218247

	4 Sequences	400000 Nucleotides			
	1	2	4	8	12
Time	0.30544141	0.15362201	0.10954541	0.09133845	0.06229648
Speedup		1.98826597	1.40235919	1.19933508	1.46618959
Efficiency		0.99413299	0.3505898	0.14991688	0.12218247

	4 Sequences	800000 Nucleotides			
	1	2	4	8	12
Time	0.87050802	0.43782272	0.31220441	0.26031458	0.17754497
Speedup		1.98826597	1.40235919	1.19933508	1.46618959
Efficiency		0.99413299	0.3505898	0.14991688	0.12218247

Self-grade

Scope: 7/10 – the program was a simplified version of the actual primer finding program. The parallel part wasn't really impressive in any way

Completion: 9/10

Implementation: 8/10 – I was trying to use MPI_Gatherv but it just wouldn't happen for me. I changed that part to a bunch of MPI_Send/Recv.

Assessment: Simply ran the program on Jupiter using different # processes and different sizes of the problem.