COMP90025 Parallel and Multicore Computing

Project 2B - Multiple Sequence Alignment

Aaron Harwood and Lachlan Andrew

School of Computing and Information Systems
The University of Melbourne

2020 Semester II

Problem Summary

- Project 2B will follow on from Project 2A, using the same problem, with one main difference: your code will be given k sequences on the input and will be required to compute the sequence alignment between all $k(k-1)/2$ pairs. This increases the amount of required calculations considerably as k increases without increasing memory requirements.
- This project will require you to use MPI on Spartan and you may use OpenMP combined with MPI if you wish, i.e. using OpenMP within each MPI task.
- We intend to test your programs on the `snowy` partition, and use that for assessment purposes (i.e. timing and ranking), and we intend to limit your Spartan job to use at most 12 nodes, with 16 cores and 32GB per node.
- This may turn out to be infeasible from a job queue time point of view and we may have to change it. In which case we may simply reduce the number of nodes to something feasible, but try and keep the cores per node the same.

Additional submission options

- We will provide additional `handin` parameters that allow you to set the number of cpus (cores) per task (i.e. MPI task) that you want allocated, and as well the number of tasks per node that you want allocated. This gives you some freedom to consider different approaches. For example, 16 cores per MPI task and 1 task per node (i.e. hybrid with OpenMP), or 1 core per MPI task and 16 tasks per node, or some variation that fits the constraints.
- We may or may not have some other options that you can't change, e.g. restricting all tasks on a node to be on a single socket (the sockets on snowy are 16 core processors), for performance measurement reasons. We will try to make sure to announce any changes to what we intend to do as soon as we can.

Code Summary

- Similar to Project 2A we have supplied you with skeleton code that does multiple sequence alignment, with some MPI driver code at the top that you are not permitted to change.
- The skeleton code serves as the sequential code as well, for the purpose of correctness in output.
- The problem is read from standard input, only the rank 0 task in MPI will read it however. That task is then required to send appropriate data to other tasks in order to make use of more than 1 node. You can't change this aspect of how the driver code works (even though it may be that standard input is duplicated and sent to all tasks by the system, we are forcing you to consider the case when all problem data is available only at the rank 0 task).

Example input

The problem instance now has an additional parameter; the third line is the number of sequences, k, that follow. Each sequence can be arbitrarily long (up to about 100,000 symbols max length).

```
3
2
3
AGGGCT
AGGCA
AAAGGGCT
```

Output

- In Project 2B we will not output all of the sequence alignments as there are $k(k-1)/2$ such pairs and they are largely unimportant to us, other than for determining correctness when compared to the sequential code. But we do want to know whether your code can generate the correct sequences and testing the minimum penalty found is insufficient for this. Therefore your code will compute a combination of SHA512 hashes (see the skeleton code for details) of the sequences and output the final hash only. You will also output a line after that with all of the minimum penalties for the $k(k-1)/2$ individual sequence alignment problems. See the supplied code to see exactly how this must be done.
- You are provided with a `sha512.hh` file that you must use to do the SHA512 hashes and you are not permitted to modify that file. See the supplied code for examples how to use it. There may be room for performance improvements around the combination of the hash results, which is accessible to you, and there may be some parallelism that can help here.

Output

- Since the actual aligned sequence is used to compute the hash, the aligned sequence is no longer computed in the driver, but rather you must compute it and there may be room for improvements there.
- The skeleton code uses much of the same computations as Project 2A and you may reuse as much of your Project 2A code as you wish, in particular the computation of the alignment for a single pair of sequences.
- The example below shows what the output may look like for the trivial problem on the previous slide, when run sequentially (the hash string did not wrap on the slide...):

```
$ cat seq.dat | mpirun seqalign-mpi-skeleton
Time: 132 us
602d0f604e8fb908195d53e681094f7d063c4168a33a18f32b4ca3d29f27073
5 4 9
```

Assessment

- Project 2B is worth **10%** of your total assessment and is half of the assessment for Project 2. It is individual work.
- There is a written submission (2/10) and a code submission (8/10) required.
- Assessment of the program (8/10) is based on correctness and performance. Incorrect programs (i.e. that give incorrect outputs or that fail to compile/run) will attract few if any marks. The top 5 fastest running programs, when given a mystery work load (you will not be told the work load in advance), will be given a bonus mark; i.e. the maximum mark for this project part is 11/10.
- At most five people will get the bonus mark. If a sixth person is tied with anyone in the top five, then anyone tied with that sixth person will not receive a bonus mark.

Submission

- Submission of Project 2B is due on **September 27th, 23:59**.
- Optimal code often requires specific compiler options. As a comment in the last line of your C++ code, put the exact command line used to compile your code. Do not put anything else on that line
- Your code submission must be submitted on Spartan following the submission instructions for Project 2B that will be announced soon. They will be similar instructions to Project 2A. You may test your code on Spartan as much as you wish.
- Both your written submission and code will need to be submitted on Canvas as well.