Running gene sequences in parallel using I-TASSER

Introduction

I-TASSER (Iterative Threading ASSEmbly Refinement) is an advanced computer algorithm for protein structure and function predictions, produced by Zhang Labs at the University of Michigan [1]. It consists of a number of binary executables that are run in order by a Perl script. Users provide the gene sequence they wish to run in an input file called seq.fasta, an example of which is:

```
>3mabA
LANLSELPNIGKVLEQDLIKAGIKTPVELKDVGSKEAFLRIWENDSSVCMSELYALEGAVQGIRWHGLDEAKKIELKKFHQSLEG
```

Note that the sequence is labelled (“3mabA” in this case) by the comment on the line starting with “>” before the gene sequence itself is specified on subsequent lines.

Running an individual sequence in parallel

At various stages in an I-TASSER calculation there are a number of independent subtasks that can be executed in parallel. The Perl script provided with I-TASSER included commands for the PBS batch queue system to submit separate batch jobs for each independent task. The Perl script in the original batch job must continue running, using the facilities of the PBS batch queue system to monitor the progress of the other tasks, so that it can determine when it is appropriate to continue to the next stage in the overall calculation. This model has two disadvantages for the user:

1. Each batch job submitted by the user submits a number of other batch scripts. While the number is not very large (~8 other jobs), it makes it more difficult to track progress of the jobs and to interrupt or cancel the job during execution if required.
2. If the system is busy, the generated batch jobs for the independent subtasks may not start, or may not start sufficiently early to complete, before the original batch job exceeds its runtime limit. Thus the job will be left in a partially completed state (Note: I-TASSER is reasonably robust at re-starting work on a given gene sequence which mitigates this problem to some extent).

For installation at HPC Wales, the I-TASSER script had to be adapted for use with the LSF batch queue system rather than PBS. Improvements to the methodology of running the independent subtasks were made at the same time, taking advantage of the ***blaunch*** command within LSF to allow the subtasks to be dispatched to run on different cores within the system, all handled within a single batch job. The user's batch job reserves a number of jobs slots (usually this means the same number of cores) on one or more nodes of the system. When running, the modified I-TASSER script determines the list of nodes available to the job from the environment variable ***LSB_HOSTS***. Where previously separate batch scripts were generated for subtasks and submitted via a PBS ***qsub*** (or LSF ***bsub***) command, ***blaunch*** is used instead to run each subtask on one of the cores of the nodes reserved by the batch script. Subtasks are distributed in a round-robin fashion across all of the nodes to aid load balancing. It is still necessary for the I-TASSER perl script to track what subtasks are running at each of the two parallel stages, to determine when the script can move on to the next stage. Previously this was done using PBS ***qstat***, now the ***lsgrun*** command is used to run ***ps*** on all the nodes reserved by the batch script to monitor the execution of the subtasks.

Running multiple sequence in parallel

Key users of I-TASSER on HPC Wales wanted to be able to efficiently run multiple gene sequences, however I-TASSER can only run a single sequence in each input file. A new Perl script was written to process a single fasta file containing all of the required sequences, generate separate seq.fasta files for each sequence in subdirectories named using the sequence labels, and then to run in parallel the normal I-TASSER script for each sequence. For example, if the user's input file contained:

```
>sequence1
ACTCCCCGTGCGCGCCCGGCCCGTAGCGTCCTCGTCGCCGCCCCTCGTCTCGCAGCCGCAGCCCGCGTGG
ACGCTCTCGCCTGAGCGCCGCGGACTAGCCCGGGTGGCC
>sequence2
CAGTCCGGCAGCGCCGGGGTTAAGCGGCCCAAGTAAACGTAGCGCAGCGATCGGCGCCGGAGATTCGCGA
ACCCGACACTCCGCGCCGCCCGCCGGCCAGGACCCGCGGCGCGATCGCGGCGCCGCGCTACAGCCAGCCT
CACTGGCGCGCGGGCGAGCGCACGGGCGCTC
>sequence3
CACGACAGGCCCGCTGAGGCTTGTGCCAGACCTTGGAAACCTCAGGTATATACCTTTCCAGACGCGGGAT
CTCCCCTCCCC
>sequence4
CAGCAGACATCTGAATGAAGAAGAGGGTGCCAGCGGGTATGAGGAGTGCATTATCGTTAATGGGAACTTC
AGTGACCAGTCCTCAGACACGAAGGATGCTCCCTCACCCCCAGTCTTGGAGGCAATCTGCACAGAGCCAG
TCTGCACACC
```

then the four subdirectories created would be called *sequence1, sequence2, sequence3* and *sequence4*. No aggregation or post processing of the results for all the sequences is performed. Each sequences is run independently and the user is required to do any collective analysis of the results required as a separate task.

The same principles for running the sequences in parallel apply as described above: ***blaunch*** and ***lsgrun*** are used to dispatch and monitor individual I-TASSER scripts for each sequence, all within a single batch job. The number of sequences running at any one time should not exceed the number of job slots reserved in the batch script, and the user can specify to under-populate each node, i.e. to use fewer jobs slots per node than there are processor cores per node if the processing of each sequence is likely to require large amounts of memory.

The time taken to do the computation for each sequence can vary significantly from sequence to sequence. Two features aid load balancing

1. Sequences are distributed in a round-robin fashion across all the nodes allocated to the job.
2. Where the total number of sequences is larger than the number of job slots, new sequences are started dynamically as each sequence completes.

Running many sequences within a single batch script makes it more likely that one or more sequences will not complete before the time limit for that batch job is reached. However, the restart functionality within I-TASSER can be used to continue computation on sequences that have not completed (without redoing work that has finished successfully) simply by resubmitting the batch job again if required.

References

[1] http://zhanglab.ccmb.med.umich.edu/I-TASSER/

[2] Similar facilities may exist within PBS and other batch queue software; however this was not investigated as part of this work.