Reproducibility Application for the Student Cluster Competition at SC16

Introduction

Replication and reproducibility of experimental computer science results in peer-reviewed papers is gaining relevance in the HPC community. SC, the leading conference in the field, wants to promote and support replication and reproducibility through a new initiative that aims to integrate aspects of past technical papers into the Student Cluster Competition (SCC). The SCC is excited to announce "A parallel connectivity algorithm for de Bruijn graphs in metagenomic applications" as the winning paper for the inaugural reproducibility initiative. This paper and accompanying application, ParConnect, will be reproduced in the SCC at SC16.

The students will be undertaking replication, which is a first step towards reproducibility. "Replication is a pre-requisite for the success of any project that intends to build on the results of a published paper. [...] Without the assurance that comes from replication, there is a high risk that you will make false assumptions about the results in a published paper, leading to incorrect comparisons or failure of your own experiments that attempt to build on those published results." says Mike Heroux, Senior Scientist at Sandia National Laboratories and author of the Transactions on Mathematical Software Journal's Replicated Computational Results Initiative.

In the rest of this document, "the paper" will be used to refer to the article, "A parallel connectivity algorithm for de Bruijn graphs in metagenomic applications." Below is a description of: (i) how to build and run ParConnect, (ii) how to create smaller datasets from the originals in the paper, and (iii) other competition details.

Getting ParConnect

Go to https://github.com/cjain7/parconnect_SCC16 and use git to do a recursive clone of the source code and external modules.

```
$ git clone --recursive <URL>
```

Building and Running the Code

ParConnect requires a C++11 compliant compiler, MPI for parallelization, and cmake to build the software. An important cmake option to set is BENCHMARK_ENABLE_CONN. Setting this to "ON" will output timing information you will need for the competition.

In the paper, there are 3 algorithm variants: "Naive", "AP", and "AP_LB". Switching between these variants requires changing one line of the source code and recompiling. In line 55 of `src/coloring/labelProp.hpp`, set

`opt_level::{loadbalanced / stable_partition_removed / naive}`.

- `loadbalanced` = AP_LB
- `stable_partition_removed` = AP
- `naive` = Naive

This is the only change that should be made to the source code! The goal is to replicate the results in the paper with the same source code used to produce the results in the paper.

To run Parconnect for the fastq file format, use the following command.

```
./benchmark_parconnect --input dbg --file <filename>
```

If running with BENCHMARK_ENABLE_CONN set to on, you can compute the timings for communication and computation, and the tuple load distribution as done in the paper. The tuple load distribution figures are clearly labeled in the program output. To calculate the communication time, add the max timings of each line (3rd timing column) for `fix_partition`, `alltoall` and `get_splitters`. The computation time is calculated as the total time minus the communication time.

Getting Test DataSets

You can get the smallest dataset used in the paper, `649.4.815.fastq`, at https://iu.box.com/shared/static/6zhvbiv1lqwareu1kniyxkvo6qelfvp7.fastq which is over 6 GB in size. This dataset will almost certainly be too large for small scale runs of ParConnect given memory constraints. Since the datasets are line based text files, it is simple to create smaller datasets from a larger dataset. Every four lines corresponds to one data point. Below is an example of how to create a dataset with 1,000,000 data points on a UNIX-like system.

```
$ head –n4000000 649.4.815.fastq > small.fastq
```

Profiling

You will need to use a profiler during the competition with the capability to profile MPI communication times. Each team will be given a license to use the Allinea debugging and profiling tools. You may use this software package or any other for profiling during the competition.

Competition Details

For the competition, new datasets will be supplied to each team. These datasets will require up to but no more than 2GB of memory per MPI rank running on 128 MPI ranks. You will be expected to be able to recreate the graphs and tables presented in the paper for the datasets given during the competition. The final output that is graded will be a report you create in PDF format. An outline for the report will be supplied at the competition and will only require text and images.

Change Log

October 5, 2016

- In the second to last sentence of the "Building and Running the Code" section, `fix_partition` was added to `alltoall` and `get_splitters` for calculating the communication time.
- Added two sentences to the end of the section "Competition Details" stating a report in PDF format will be the graded output for this challenge.