CSCI-4974/6971: Assignment 4 (100 pts)

Social Graph Analysis

Due Date: Thursday 10 Nov. 2016, 16:00

For this assignment, we're going to be used the distributed graph code we developed in class to actually run in a distributed environment. We'll be using the CCI Amos system and examining various partitioning/load balancing strategies on various graphs. This assignment will cover the following:

1. Connecting to and transferring files to the CCI systems
2. Submitting and running jobs on these systems
3. Examining how different partitioning strategies effect execution performance on distributed graph processing

For background material and reference, use the lecture slides and our hands-on code along with the CCI user wiki:

- `https://secure.cci.rpi.edu/wiki/index.php/CCI_User_Wiki`
- `https://secure.cci.rpi.edu/wiki/index.php?title=Blue_Gene/Q`

Submit the code file to `slotag@rpi.edu` by the due day and time listed above. Put your responses to the short answer questions in comments in the code file.

1 Connecting and Setting up Environment

You should have received an email from CCI indicating that your user name has been created for a CCI project (this class). Create a new password, if necessary, for the user name and use it to test your ability to connect to one of the landing pads as listed in the wiki.

- `ssh username@lp01.ccni.rpi.edu`
- `ssh amos`

Additionally, use it to copy the test file tar from the course website to your home directory and expand it there. This will contain the code file, graph file, and partition files.

- `rsync --progress hw04-dist.tar.gz username@lp01.ccni.rpi.edu:./barn/hw04/`
- `tar xvf hw04-dist.tar.gz`

2 Running Code

Use the supplied `cpp` code file. For this assignment, we won't actually be writing any new code. Instead, we're just going to be creating a job script to process an R-MAT graph and examine scaling behavior on PageRank and breadth-first search. Create and run a job script that outputs results for all combinations of the test cases:

1. Executes using 2, 16, and 64 MPI tasks
2. Utilizes vertex block, vertex balanced, and vertex+edge balance partitioning strategies

For example, one test case might be executed as:

- `srun -o test.vb.2 --partition debug --time 10 --ntasks 2 ./a.out rmat_22.ebin rmat_22.vb.2`

Check the wiki for information on how to use modules, compile (make sure you use `-std=c++11`), create/run/submit jobs. To run with vertex block partitioning, don't supply an argument for partfile. For vertex balanced and vertex+edge balanced partitions use the supplied partition files of "`rmat_22.vb.#`" and "`rmat_22.vbeb.#`" for vertex balanced and vertex+edge balanced partitions, respectively. For simplicity, you can just submit all runs as a single job. Or you can run each separately, whichever you find easiest.

3 Response Questions

In a single file, submit your job commands/scripts, the output of your job runs, as well as responses to the following questions/prompts:

1. How does each algorithm scale (computation, communication, and idle times) for each partitioning strategy with increasing task count? Just give a general overview.
2. Explain this scaling behavior in the context of the R-MAT graph structure (high irregularity) and your knowledge of the two algorithms.
3. Based on the above observations, if we wanted to scale to a larger irregular graph running label propagation with a high number of tasks, what considerations should we make for effective load balancing?
4. We talked about but haven't explicitly implemented an edge block partitioning, where each task has a (near) equivalent number of edges but a variable number of vertices. How would you expect this partitioning strategy to perform relative to the other strategies for these tests?

This homework will be evaluated on a successful creation and execution of the jobs (60 pts) and answers to response questions (10 pts each).