CSE 687 Object Oriented Design Project

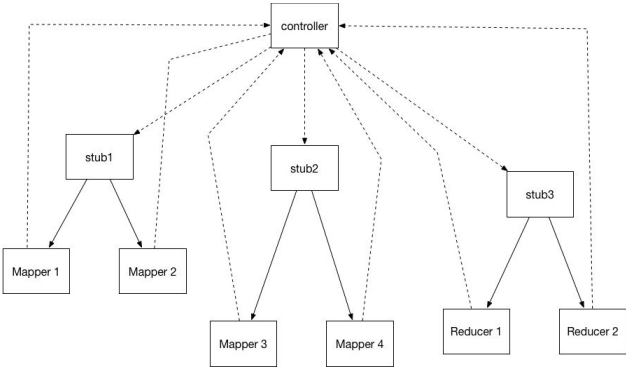
MapReduce

Phase #4

# Background

In Stage 4, you will be extending your Project 3 solution:

1. Create a stub process that sits and listens for messages (using sockets). The stub will get a message from the controller to tell it to create child processes (either map or reduce). The stub programs will be assumed to be always running (i.e. just start these manually before you run your controller).
2. Map and Reduce processors will communicate status updates to the controller (not the stub) via a heartbeat message sent every k seconds (you may choose a reasonable k value).
3. The controller will tell the Reduce processes when to begin working.
4. The same stub process may create both Map and Reduce processes.
5. You must specify your message format.
6. We can assume the correct input files are already locally available (no need to transfer them).
7. Functionality does not need to transfer files over the network from the map processes to the reduce processes.
8. The controller program will send message to a predefined list of stubs. This can be maintained in configuration.



# Methodology Requirements

1. You must work in a team of 2. If there is an odd number of students, there will be a single team of 3. No one will be allowed to work independently.
2. You must use C++ in conjunction with MS Visual Studio and GitHub (student account).
3. You may use anything in the std library and additionally may use anything in the BOOST (https://www.boost.org/) library.
4. All changes must go through a code review by your partner(s).
5. All partners must submit a reasonably equal number of check-ins for each project, as evidenced by git submissions.
6. All partners must treat each other with respect.

# Technical Requirements

1. Code must have unit tests above 95% line coverage. You may use, Boost unit testing, Google unit tests, or your own unit testing.
2. Code formatting & comments should follow Google style guide: <https://google.github.io/styleguide/cppguide.html>
3. Errors, warnings, information, fatal errors, etc., will be logged. Boost Logging can be used (<https://www.boost.org/doc/libs/1_63_0/libs/log/doc/html/index.html>).

# Rubric

1. should not crash in error scenarios: 10%

Errors should be appropriately handled and made visible.

1. Code behavioral correctness: 50%
2. Code Unit testing: 10%

Code must have code coverage above 95% by line count.

1. Code Commenting & Style: 10%

Code must meet style & comment guidelines.

1. Code Organization and Structure: 10%

Code must adhere to SOLID principles.

1. Procedural Correctness: 10%

Code must have gone through code reviews. Code reviews should be sufficiently thorough. Members that do not give their team quality code-reviews will be penalized.

Additionally, each member of the team must contribute a reasonably equal amount to their project. Each member will submit a brief report detailing what percentage of work was completed by each member. I encourage teams to keep detailed meeting notes.

# Bonus

You may only attempt the bonus if the code is behaviorally correct, unit tested, correctly formatted, and reasonably structured.

10%: Transfer the Map processes output over the network via socket transmission of the file contents from the Map processes to the Reduce processes.