High Performance Optimizations and Dynamic Load Balancing for Computational Aerodynamics Solvers

Blessy Boby TVE15CS020
Gmon K TVE15CS024
Shilpa S TVE15CS055
Robin R LTVE15CS068

April 24, 2019

1 Initial Requirements

1. Use the Ubuntu operating system and your PC must have at least 4 cores.
2. You must install the OPEN MPI latest version using either of the following two ways.
   - `sudo` apt-get install openmpi-bin openmpi-common openssh-client openssh-server libopenmpi1.3 libopenmpi-dbg libopenmpi-dev
   - Download the open-mpi tar file from http://www.open-mpi.org/software/ompi
   - Decompress the downloaded file (should be called something like openmpi-x.x.x.tar.xxx, changing x.x.x for the version downloaded):

     `tar` -xvf openmpi-*
   - Go into the new folder created from the decompress.

     `cd` openmpi-*
   - It is necessary to add on the prefix the installation directory we want to use for OpenMPI. The normal thing to do would be to select the next directory "/home/user/.openmpi".

     `./configure` –prefix=”/home/$USER/.openmpi”
   - Install the MPI
     `make`
     `sudo` make install

- All that is left to do is to include the path to our path environment the path "installation_directory/bin" and to the library environment variable "installation_directory/lib/". If your system uses bash you'll have to use the command export.

  ```
  export PATH="$PATH:/home/USER/.openmpi/bin"
  export LD_LIBRARY_PATH="$LD_LIBRARY_PATH:/home/USER/.openmpi/lib/"
  ```

3. Next you must have metis software package installed on your system.

   - Download the metis tar file from http://www.gilith.com/software/metis/download.html
   - Extract it with the command

     ```
     tar xvzf metis.tar.gz
     ```

   - Change to directory metis

     ```
     cd metis
     ```

   - Building METIS requires CMake 2.8, found at http://www.cmake.org/, as well as GNU make. Assumming CMake and GNU make are installed, two commands should suffice to build metis:

     ```
     make
     make config
     ```

   - Install the metis

     ```
     make install
     ```

   - Change into the directory as

     ```
     cd /build/Linux-x86_64/programs/
     ```

   - In this folder you can see a file 'gpmetis' which is executable and you can use this program to partition the graph using the following command

     ```
     ./gpmetis filename.graph count
     ```

4. You must have a gcc compiler and g++ compiler in your system you can install it using

   ```
   sudo apt-get install gcc
   sudo apt-get install g++
   ```

2 Running the system

1. Clone into the following repository

   ```
   git clone https://github.com/robinl3680/Final-Year-Project.git
   ```

2. Change into the directory

   ```
   cd Final-Year-Project/Progress/MeshToGraph
   ```

3. Compile and Execute the Mesh to Graph conversion algorithm as follows.

   ```
   g++ MeshToGraph.cpp -o MtoG
   ./MtoG input_file_name.su2 output_file_name.graph output_faces.txt
   ```

4. Go to the directory 'MPI'

   ```
   cd ../MPI/
   ```

5. In order to execute the different communication algorithms individually do the following.

   - Centralized communication approach

     ```
     mpicxx Centralized.cpp -o Centralized
     mpirun -np cores_count+1 Centralized input.graph.part.count input.graph
     ```

   - Peer to Peer communication approach

     ```
     mpicxx PeerToPeer.cpp -o Peer
     mpirun -np cores_count Peer input.graph.part.count input.graph
     ```

   - Greedy communication approach

     ```
     mpicxx Greedy.cpp -o Greedy
     mpirun -np cores_count Greedy input.graph.part.count input.graph
     ```

   - Another way to run the three in once use following shell script and you can edit the input files mentioned there.

     ```
     sh testbash.sh
     ```

3 Running within a Cluster

1. Connect the different systems using a wired or wireless LAN.
2. Make sure that all system contain same version of the 'MPI'. To check that.

   ```
   mpirun –version
   ```

3. Using the ping command check the physical connectivity between systems.

   ```
   ping ip_address
   ```

4. Make sure that each system has the required programs to be executed are in the same path.
5. Enable password less login between the systems to avoid the authentication issues.

   ```
   ssh-keygen -t rsa
   ssh-copy-id remote_username@server_ip_address
   ```

6. Check the $PATH variable to ensure they are same.

   ```
   echo $PATH
   ```

7. If they are not same make it same by copying one of them and replacing it with others using

   ```
   export $PATH=new_path
   ```

8. Create a host file that contain the ip addresses of the connected systems.

   ```
   nano host_file
   ip_address1
   ip_address2
   ...
   ```

9. Now execute the different communication algorithms by specifying the host file. An example is.

   ```
   mpicxx Greedy.cpp -o Greedy
   mpirun -np cores_count –hostfile host_file_name Greedy input.graph input.graph.part.count
   ```