README

To demonstrate the code, we use the problem – 1 (heat conduction in annular disc) with much coarser grid (36796 grid points as opposed to 2356220 grid points in the main report) and reduced number of iterations (10000) for quick demonstration of the code capability and to facilitate submission of code along with necessary input files in Submitty.

In general, the prerequisite for running the analysis are three ASCII files describing the grid–

1) xadj

2) adjncy

3) nodal_data

These files for problem – 1 (as mentioned in the report) are provided in the folder containing the code. The file descriptions are in the report and will not be elaborated here. These files will, however, not be required in this case as described below.

To convert the ASCII grid files to BINARY files with partitioned grid, a grid partitioning code was developed – *grid_partition.c*. This code utilizes the METIS code to perform the grid partitioning.

The dependency for this code is the **METIS** package (**METIS** is a software package for graph partitioning that implements various multilevel algorithms). It needs to be loaded before compilation. Thus, the compilation procedure is as follows –

module load metis

gcc -Wall -o grid_partition grid_partition.c -lmetis

grid_partition code needs 5 inputs – xadj, adjncy, nodal_data, number of partitions, and endianness of the output files (0 for Little Endian and 1 for Big Endian). The output files are – *g_parts, g_xadj, g_adjncy, g_nodal_data, g_global_vert_ids*. The output files are in BINARY format with endianness specified in the input.

For example, suppose we need to partition a domain into 8 partitions. The run command would be –

(for Big Endian output format)

./grid_partition xadj adjncy nodal_data 8 1

(for Little Endian output format)

./grid_partition xadj adjncy nodal_data 8 0

PLEASE NOTE -

For convenience, the output files - *g_parts, g_xadj, g_adjncy, g_nodal_data, g_global_vert_ids* are already created for 8 partitions of problem 1 domain. These files are present in the folder.

for little endian format the names are modified with suffix '_l' (manually) as –

g_parts_l, g_xadj_l, g_adjncy_l, g_nodal_data _l, g_global_vert_ids_l

for big endian format the names are modified with suffix '_b' (manually) as –

MAIN PROCEDURE

The grid solver code consists of 'c' files with prefix 'gs'

The main file is ***gs_maincode.c***. There are 5 header files – *gs_constants.h, gs_pdtypes.h, gs_sdtypes.h, gs_keyFuncs.h,* and *gs_utils.h*. The files should be present in the same folder. Compile the code as following –

mpicc -O3 -Wall -o gsolve gs_maincode.c -lpthread

Note: the code is indifferent to endianness of the local machines which means it is the user's responsibility to ensure that machine endianness matches the endianness of the input binary data.

Note *gs_constants.h* contains all the constants defined in the code. To run the code in BGQ set the 'BGQ' macro in the *gs_constants.h* to 1. Currently, it is defined as 0.

please modify the following line in *gs_constants.h* to run on Blue Gene Q

'#define BGQ 0 // BGQ = 1 when running on BG/Q'

To run the code –

mpirun -np 8 ./gsolve g_parts g_xadj g_adjncy g_global_vert_ids g_nodal_data res

where, 8 is the number of processors (should be same as the number of partitions created previously) and 'res' is the output binary file name.

For the present case, to run the code on a little endian machine do following –

mpirun -np 8 ./gsolve g_parts_l g_xadj_l g_adjncy_l g_global_vert_ids_l g_nodal_data_l res

for Big endian machine

mpirun -np 8 ./gsolve g_parts_b g_xadj_b g_adjncy_b g_global_vert_ids_b g_nodal_data_b res

---

To view the final output as a figure, run the following MATLAB scripts – *view_res_l.m* or *view_res_b.m* in the same folder with the results file 'res' and the input 'g_*' files.

- ⇨ for little endian output format (e.g. output from mastiff) –

matlab -nodesktop -nosplash -r "view_res_l; exit;"

- ⇨ for big endian output format (e.g. output from BGQ) –

matlab -nodesktop -nosplash -r "view_res_b; exit;"

The end result would be a '***result.png***' file showing the output in graphical format.

Note that the MATLAB module must be loaded before running the *view_res_l.m* or *view_res_b.m* scripts.

THUS, FOR MASTIFF –

mpicc -O3 -Wall -o gsolve gs_maincode.c -lpthread

mpirun -np 8 ./gsolve g_parts_l g_xadj_l g_adjncy_l g_global_vert_ids_l g_nodal_data_l res

matlab -nodesktop -nosplash -r "view_res_l; exit;"

FOR BGQ – (set BGQ to 1 in *gs_constants.h*)

mpicc -O3 -Wall -o gsolve gs_maincode.c -lpthread

mpirun -np 8 ./gsolve g_parts_b g_xadj_b g_adjncy_b g_global_vert_ids_b g_nodal_data_b res

matlab -nodesktop -nosplash -r "view_res_b; exit;"