CS 404: Lab 6: Parallelizing RAD1D

Getting Started

1. Log in to a collaboratory machine.

2. Open a web browser and go to the course website[1]. Download RAD1Dmp.zip and save it in your home directory (h:\Users\userid). You may also copy it from my home directory: H:\Users\ajp.

3. Double click on the file to extract. This will create a directory called RAD1Dmp with three subdirectories: source contains the source for RAD1Dmp, scripts contains sample .bat scripts, and results is an empty directory where your results will go. The goal of this lab is to evaluate the performance of RAD1Dmp as we run it on more and more processors.

Building RAD1Dmp

1. First, we need to build the executable, and we'll do it with VizStudio. Launch VizStudio and create a new "Win32 Console Application" in the RAD1Dmp directory. I named mine "rad1mp," and I'll assume you did as well. Add the contents of RAD1D\source to the project. Be sure to add the files for the microsecond timers (hrpc.h and hrpc.c) which are located in the timers directory.

2. In addition to the timers, I've added two new files: commlib.c and commlib.h. These files contain the main parallel routine Update, which shares data between the processors. Update uses the same communication algorithm we developed on Monday with the cookies. There is also a modified version of GetInitialCond that gets the section of initial data needed by processor P. The parallel-specific sections of code are turned on and off by setting the PARALLEL flag in commlib.h (#define PARALLEL turns the PARALLEL code on, #undef PARALLEL turns it off). Look through main. See if you can figure out what's happening in the parallel sections. We will be making both parallel and serial runs, so turn PARALLEL off and build the executable (rad1mp.exe–it will

[1] www.cs.cornell.edu/Course/cs404/2002sp

be in rad1mp\Debug directory). Change the name of rad1mp.exe to rad1ser.exe.

3. Now, let's build the parallel version. First, turn PARALLEL back on. Under the "Build" menu, select "Rebuild All" to force VizStudio to recompile everything. What happens? We have two problems: VizStudio doesn't know where to find the header file for the MPI library, and it doesn't know where to find the .lib file. We need to change these in the "Settings" panel. Select "Settings" from the "Project Settings" menu to bring up the panel. Click on the "C/C++" tab. From the "Category" menu, select "Preprocessor" to bring up the options for the C-preprocessor. Add C:\Program Files\MPIPro\include to the "Additional Include Directories" field; this is the location of mpi.h. Now, click on the "Link" tab and select the "Input" category. Add MPIPro.lib to the "Object/Library Modules" line. This tells VizStudio to link to the MPIPro library (analogous to -lMPIPro). Then type C:\Program Files\MPIPro\lib in path field. This adds the MPIPro*bs* directory to the list of directories VizStudio will search (analogous to the -L flag). Click OK and build the executable.

Running Interactively

1. We will now run both the serial and parallel versions of RAD1D on Velocity. Velocity nodes are requested using the Cluster CoNTroller batch System (CCS) by submitting a .bat file. A .bat file is simply a series of DOS commands. Commands for CCS are specified as comments (lines starting with REM) followed by "CCS." All .bat files must specify five things: the account (userid), the type (batch or interactive), the number of nodes, the job requirements (allows you to choose the type of nodes you want), and the number of minutes you're requesting. Open interactive.bat (VizStudio is a good choice, you could also use WordPad or NotePad). This is a simple .bat file that requests a single node from the "development" pool (2-way Pentium machines set aside for short running jobs). The job is set to "interactive." The only real difference between interactive and batch is that in batch-mode, CCS will execute the .bat file on the first node allocated for the project. Edit your copy of interactive.bat, replacing my userid with yours. Then, open a command prompt and cd into the scripts directory. To send the .bat file to

CCS, type "ccsubmit interactive.bat."

2. Your job is now in the job queue. Jobs go through four states: waiting (CCS is trying to find nodes that meet your needs), starting (CCS is gathering the nodes you requested and setting them up for your exclusive access), running (you can run your commands), and clearing (you've released your nodes and your time has expired–CCS is logging you out). To see the queue, type "ccq." Find your job (look for your ID on the left). Your job's state is indicated by a single letter (W,S,R,or C). What state is your job in? Keep checking ccq until your job begins running. When your job is running, the last column of ccq is the master node for your job.

3. You now have exclusive access to one development node; however, since you asked for an interactive job, nothing is happening on the node. You need to log on to that node and do your work. To do this, type "telnet ctcdevXX" where XX is replaced by the number of the node indicated in ccq. You are now on your node, any commands you enter will be run on that machine.

4. First, we need to get the data and executables on to the local drive for this machine (reading and writing from a local drive is much faster than using the shared H: drive). Enter the following commands (don't type the italicized notes), substituting your userid for mine:

   > t: *Changes drives to the local t: drive*
   > mkdir ajp *Creates a directory named after you*
   > cd ajp *You are now in the directory*
   > copy h:\Users\ajp\RAD1Dmp\rad1mp\Debug*.exe . *Copies the executables*
   > copy h:\Users\ajp\RAD1Dmp\xlarge.cmnd . *Copies the command file*
   > copy h:\Users\ajp\RAD1Dmp\Cxl.txt . *Copies the initial condition file*

5. You're now ready to run. To run in serial, type "rad1ser.exe xlarge.cmnd." This will output the usual Cfinal.txt and a file called Time_1.txt containing the amount of time spent setting up the program and the av-

erage time for each iteration. To check the time values, type "more Time_1.txt." Write down the second number.

6. To run in parallel, you must use the mpirun command. This command takes a program, and starts several copies of it, allowing them to refer to each other through MPI calls. However, before we can use mpirun, we need to tell it which machines to run on (it doesn't communicate with CCS, so it doesn't know!). CTC has created a program that figures out which nodes CCS gave you and creates a file called "machines" that mpirun needs. Thus, to run RAD1D on two processors, type:

   > call machinemaker *Creates machines*
   > mpirun -np 2 rad1mp.exe xlarge.cmnd

   For a few seconds, it will appear that nothing is happening. Then, all of a sudden, you will see all of the output from RAD1D. Type "dir" to list the contents of your directory. In addition to the inputs, executable, and the outputs from the serial version, you should see three files from the parallel version: Cfinal.txt.0 (the concentration at the first 50,000 grid points), Cfinal.txt.1 (the concentration at the second 50,000 grid points), and Time_2.txt. Look at the new time file, is the parallel version faster? Write down these times.

7. Now, you need to move the output back to your results directory and delete your files from t:

   > copy *.* h:\Users\ajp\RAD1Dmp\results *Copies everything to h:*
   > del *.* *Deletes everything*

8. Lastly, you need to let CCS know that you're done, so you don't get charged for minutes you're not using. Type "ccrelease" to release your node, and then "exit" to quit telnet.

Running in Batch

1. Running interactively is necessary when you are developing your application and is a good way to learn the system. Once you've got your

program working, you want to automate it as much as possible: you want to run in batch mode. A .bat for a batch job must set up the same CCS parameters as an interactive job (type should now be set to batch), and you need the DOS commands to set-up and run your program. These DOS commands will look exactly like the commands you entered interactively. I've provided a sample batch script called MPsample.bat. Open it up and take a look. The beginning of the file sets the CCS parameters. The next section sets some DOS variables. These function like a macro in a Makefile or a variable in a program– they allow you to change how the program runs with minimal typing (and hopefully, fewer errors). The value of USR and NODES should match the CCS account and nodes variables. The variable PROCS is the number of processes to start (some multiple of NODES), EXEC is the name of the executable, INPUT is the name of the input file, and SETSCR and CLNSCR are the names of two additional scripts: one to copy the data onto the T drive, and the other to copy the results back to H. You've been assigned a number of nodes and processors to run. Change the appropriate variables (don't forget to change the account, and make sure you are requesting development nodes).

2. You will also need to make some changes in the set and clean scripts. Make sure that these are set to point towards your files (change the USR variable), and the variables that point to the directories and files.

3. Now, submit your job to CCS by typing "ccsubmit MPsample.bat." Check the queue periodically to follow its progress. When the job starts, a message will appear on your screen (the result of the "net send" commands in the .bat file). You will also get messages when rad1mp starts and when it is finished. When it completes, check your results directory to see if your answers are there. Look at the timing file (Time_N_P.txt where N is the number of nodes you used and P is the number of processes), and write down your answer. Give the times from your three runs to me.

4. If time permits, repeat your run using the large.cmnd file (and Cl.txt). You will have to change the name of the input file in both MPsample.bat the MPsamp_set.bat and the name fo the data file in MPsamp_set.bat. Check the time and give it to me.