THREAD LEVEL PARALLELISM

5/06/2009

TLP Project (DUE: Fri. 5/15/2009, 5PM)

Now that you understand the basics of pthreads and MPI, you have been assigned the following task:

Management needs a recommendation regarding the next purchase of computers. *Should management purchase fewer more expensive quad-core systems or more single core systems that will be clustered?*

Thread Level Parallelism

Thanks to the TA, Marty Nicholes, for the prior project handout that I leveraged.

BACKGROUND

Suppose your company wants to purchase some new computers. There are two types of computers that you can choose: multiple single core systems that will be clustered and fewer expensive quad-core systems. Suppose your budget is limited. Please write a report to a technical manager who needs to make purchase decisions with your suggestions about the best type of computers to purchase.

INTRODUCTION

In the TLP project, you must parallelize a program that generates the Mandelbrot set. The code does lend itself to parallelizing. The code must be able to execute in parallel on both the pthreads architecture and the MPI architecture. Then you need to run the parallelized code on various pthreads and MPI configurations in order to understand how performance scales, when you add threads or cluster members. Based on this observation and your understanding, your report will recommend either higher performance quad-core machines, or lower performance single-core machines that will be clustered.

TOOLCHAIN

- pthreads howto (http://www.yolinux.com/TUTORIALS/LinuxTutorialPosixThreads.html)
- MPI Documentation (http://heather.cs.ucdavis.edu/~matloff/MPI/NotesLAM.NM.html)
- General LAM MPI documentation (http://www.lam-mpi.org/tutorials/one-step/lam.php)

SAMPLE CODE

(Download source: EEC 171 001 SQ 2009 Resources / Project / TLPCode)

- mandelbrot4.c – Sample code that computes the Mandelbrot set and outputs a tga file.
- mandelbrot5.c – Sample code that does a dot product between two vectors using mpi
- Mandelbrot_parms – parameters for the Mandelbrot runs.
- Data move in mpi – Sample code showing movement of data between master and slaves. (http://www.lam-mpi.org/tutorials/one-step/ezstart.php)
- vecsum.c – Sample code showing MPI scatter and gather in action.
- dotprod.c.pthreads – Sample code that does a dot product between two vectors using pthreads.
- dotprod.c.mpi – Sample code that does a dot product between two vectors using mpi.

OTHER REF

- Gather Web page – Web page illustrating MPI gather (http://mpi.deino.net/mpi_functions/MPI_Gather.html)
- Mandelbrot pictures – These pictures will match your output (http://www.maths.tcd.ie/~nryan/mandelbrot/seahorsezoom.html).

BEFORE YOU START

1. LAM setup

a. First logon to one of the ECE unix systems.

b. 3 initial steps before you using MPI

```
setup lam
vi .cshrc (add "setenv LAMRSH 'ssh -Y'")
open a new terminal window
```

Note: Make sure that the lam setup is correct (`cat .software` and make sure `lam #lam` is listed next to `@standard`).

2. View the processor speed information

```
cat /proc/cpuinfo
```

3. Start up MPI

Run the `lamboot` command to start up MPI. Compile MPI version of the target file, e.g., dotprod.c, using the following command.

```
mpicc -I $LAMHOME/include -L $LAMHOME/lib -g -o dotprod dotprod.c
```

4. Compile Mandelbrot samples

The sample file mandelbrot4.c requires no special handling for compiling. However, mandelbrot5.c does require the math library, so you compile it like this:

```
cc –lm –o mandelbrot5 mandelbrot5.c
```

5. Parallelize Mandelbrot

The goal here is to review the code and determine a way to parallelize the algorithm. I suggest you go for parallelizing along the rows, as the algorithm is already set up that way. In your report you will briefly describe how you did this step, both for pthreads and MPI.

Verify that the tga file produced in parallel matches the file you get running the provided code.

THREADS PERFORMANCE

You need to determine how to characterize the performance of the algorithm as you spread the work among threads. Please debug your code on ECE systems before checking performance on tetra.

1. First logon to one of the ECE unix systems. These systems are single core. Record the processor speed information for later use in your report and recommendation.
2. Run the pthreads version of mandelbrot to use various numbers of threads, recording performance for each run and the output the best performance of your system (program runtime in msecs). Note as the balance may be different among the systems with different number of threads, you probably need to test a wide range of possibilities until to find the best results.
3. Now logon to tetra.cs.ucdavis.edu and perform steps 1-2. Tetra is a quad core system.

USING MPI

You need to determine how to characterize the performance of the algorithm as you spread the work among cluster members using MPI. Please debug your code on 2 ECE systems at most before using more nodes.

1. First logon to one of the ECE unix systems. These systems are single core. Record the processor speed information for later use in your report and recommendation.

   Create the lamboot file to point LAM at other ECE systems for cluster use (Available machine names can be found in **171 001 SQ 2009 Resources/ Project /ECEhostname.txt**.).
2. Run the MPI version of mandelbrot to use various numbers of cluster members, recording performance (program runtime in msecs).

NOTE: Here is the procedure to access command line arguments with MPI.

1. Write your MPI capable code to process command line arguments after the call to MPI_Init().

```
MPI_Init (&argc, &argv);
/* do not access command-line arguments until after MPI_Init()
 *       is called */
numthrds = atoi(argv[2]);  /* first arg should be number 2 */
```

2. Run the MPI version of the code, passing in the argument (3 in this case):

```
mpirun -c 3 mandelbrot -- 3
```

SUBMISSION

Use the SmartSite to turn in: **1)** the source code for the pthreads version of Mandelbrot, 2) the MPI version of Mandelbrot, and **3)** the PDF file for your recommendation (see requirements below).

REPORT GUIDELINES:

- Your report should target a technical manager who needs to make design decisions. You can assume that your manager understands parallel architecture (threads, clusters.) but doesn't know the code you're evaluating, or your methodology. Your report should help the manager choose the best type of computers to purchase and support his decision with technical data.
- As part of the support for your recommendation, your report should cover the requirements listed above for threads and MPI, as well as discuss how MPI compares with threads in terms of performance and cost/performance.
  - Make sure your report discusses the following points on threads:

        - How many threads can profitably be put on a threads capable machine? How does performance per thread scale with more threads? How does overall performance scale with more threads?

    - Make sure your report discusses the following points on MPI:

        - How does performance per MPI node scale with more nodes? How does overall performance scale with more nodes; at what point does it not make sense to add more nodes?

- The report should be no more than 3 pages in PDF format with 1 inch margins and no smaller than 10 point type.
- You are encouraged to include graphs as mentioned above to better illustrate how your experimental results and support your conclusions. However, don't feel like you have to submit every single graph. You should definitely use a few graphs to underscore important points that you are making.
- Organize your report in a way that is easy to follow.
- Do not just write it as a sequence of responses to each section! Put the problem in context, have a conclusion, and so on ... everything you've learned about writing an essay in English class applies here.
- Remember to include a short introduction about the background of the project and how the testing set up.
- Be sure you have thoroughly read the assignment and include all information it asks for.

DUE DATE: Friday 5/15 at 5PM.

Note: tetra may be shut down for a couple of days for research purposes (before May 11). Please check the processor activity with *`top`* command to see whether there are CPU-intensive tasks on the system if you run your program before May 11. Please help to avoid the overlap of your experiments with exclusive access for research usage. Thank you.