

# THREAD LEVEL PARALLELISM

4/29/2009

TLP Warmup (DUE: Wed. 5/6/2009, 5PM)

Start to play with programs utilizing pthreads and Message Passing Interface (MPI).

# 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. Your technical manager needs to make a purchase decision with your suggestions about the best type of computers to purchase.

## INTRODUCTION

The TLP project will be done using two similar techniques. First, using pthreads, a program will be split up into threads, which share memory on a single system. Second, using the MPI protocol, the same program will be split up across a cluster of systems. This presents some interesting trade-offs, but the focus of the project will be on trading off a fast quad-core system, versus a cluster of cheaper single core systems.

## 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](#))

- 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

Note : dot product between vector  $\mathbf{a}$  and  $\mathbf{b}$  is:

$$\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^n a_i b_i = a_1 b_1 + a_2 b_2 + \dots + a_n b_n, \text{ where } n \text{ is the length of the vectors.}$$

## USING PTHREADS

The goal here is to become familiar with the pthreads. Some steps will be provided, and some steps you will need to determine by looking at the docs. Commands to be entered will be shown in the *courier italic font*.

NOTE: when you are asked to note any items in the following steps, record values into a file that you will turn in with the assignment.

1. First logon to one of the ECE unix systems. These systems are single core. Record the processor speed information provided by the following command (record result in notes file):
-

```
cat /proc/cpuinfo
```

2. Modify the pthreads version of dotprod to use 8 threads (by modifying the max threads define), and compile by typing:

```
cc -lpthread -o dotprod dotprod.c
```

3. Run the program, and save the output of the run.
4. Modify the pthreads version of dotprod by increasing the vector length to 10,000,000. Compile and run it, saving the output.
5. Now logon to tetra.cs.ucdavis.edu and perform steps 1-4. Tetra is a quad core system.

## USING MPI

The goal here is to become familiar with the MPI. Some steps will be provided, and some steps you will need to determine by looking at the docs. Commands to be entered will be shown in the *courier italic font*.

NOTE: when you are asked to note any items in the following steps, record values into a file that you will turn in with the assignment.

1. First logon to one of the ECE unix systems.
2. Create a lamboot config file listing the hostname of the current system, as well as 3 other ece systems. The configuration file tells the lamboot command which systems are part of the cluster. For example, my configuration file, called lam\_boot\_schema looks like this:

```
indigo.ece.ucdavis.edu
mamba.ece.ucdavis.edu
redbelly.ece.ucdavis.edu
viper.ece.ucdavis.edu
```

Note: some other ece system names can be found in **171 001 SQ 2009 Resources / Project/ECEhostname.txt**

3. Run the lamboot command to startup up MPI. Lamboot starts up the LAM MPI daemon on all machines you have specified in the configuration file. For example:

```
lamboot -v lam_boot_schema
```

4. Run the lamnodes command and save the output in the log file. The lamnodes command simply shows the nodes configured into the MPI cluster.
5. Compile MPI version of the dotprod.c file, using the following command. mpicc is a wrapper for the C compiler that includes all the necessary command line switches for the underlying compiler to find the LAM include files, the relevant LAM libraries, etc..

```
mpicc -g -o dotprod dotprod.c
```

NOTE: remember to copy the executable file dotprod into a directory that is in your executable PATH. (for example ~/bin)

6. Run dotprod across 4 nodes, using the following command, logging the output to hand in. If the cluster is working correctly, you should see partial sums from each of the nodes in the cluster, and then the final sum. The -c option specifies how many members of the cluster to run on.

```
mpirun -c 4 dotprod --
```

NOTE: It is a good idea to run the *lamclean -v* command between runs. Also run the *lamhalt* command when you are finished.

---

7. Modify the MPI version of dotprod.c to use a vector length of 10,000,000, compile and rerun. Log the output to hand in.

## SUBMISSION

Pretty easy for the warmup. Use the SmartSite to turn in: **1)** the log files from the pthreads and MPI runs; **2)** the file containing the notes you recorded in response to the question, i.e., pthreads step 1; **3)** a text file named "README" that describes each of the log and note file. For the real project, you will have to turn in a FILEINFO file that provides a quick description of each file that is part of your submission.

**DUE DATE: Wednesday 5/6 at 5PM.**

---