# THREAD LEVEL PARALLELISM

4/27/17

## TLP Warmup (DUE: Tue. 5/2/2017, 10AM)

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

# Thread Level Parallelism

## INTRODUCTION

The TLP warmup 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, and the focus of the following TLP 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 (https://www.open-mpi.org/doc/current/)
- MPI Hello World (http://mpitutorial.com/tutorials/mpi-hello-world/)

## SAMPLE CODE

(Download source from Smart Site: Resources / Projects / TLP / Code)

- dotprod_pthreads.c – Sample code that does a dot product between two vectors using pthreads
- dotprod_mpi.c – Sample code that does a dot product between two vectors using MPI

  Note: dot product between vector ***a*** and ***b*** is:

$$\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^{n} a_i b_i = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$$

  where *n* is the length of the vectors.

## USING PTHREADS

The goal here is to become familiar with 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 or record 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 Linux systems. Record the processor speed information provided by the following command, and record the number of cores:

***cat /proc/cpuinfo***

2. After downloading the sample code from SmartSite, compile dotprod_pthreads.c by typing:

   ***gcc -lpthread -o dotprod dotprod_pthreads.c***
3. Run the program by typing *dotprod*, and record the runtime from the program output.
4. Now modify dotprod_pthreads.c by changing the maximum number of threads to 8 (see the MAXTHRDS define). Compile and run it again, recording the runtime. How much did the runtime decrease from increasing the number of threads? Is this what you expected from the increased parallelism and your observations in Step 1?

## USING MPI

The goal here is to become familiar with MPI. Some steps will be provided, and some steps you will need to determine by looking at the documentation. 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. Log in to one of the ECE Linux systems.
2. If you are unfamiliar with Linux shell take this tutorial. It'll be time well spent:
   http://linuxcommand.org/lc3_learning_the_shell.php
3. Add the Open MPI directory to your path:

   **setenv PATH ${PATH}\:/afs/ece/users/jowens/eec171/openmpi/openmpi-latest/bin**

   You may want to add this line to your .cshrc file
4. Set the MPIRUN variable

   **setenv MPIRUN /afs/ece/users/jowens/eec171/openmpi/openmpi-latest/bin/mpirun**

   Again, you may want to add this to your .cshrc
5. Create a config file listing the **hostname of the current system**, as well as 3 other ECE systems (if you do not know the hostname of the current system, type *echo $HOST* at the command prompt). The configuration file tells Open MPI which systems are part of the cluster. For example, my configuration file, called MPI_HOSTS looks like:

   **indigo.ece.ucdavis.edu**

   **mamba.ece.ucdavis.edu**

   **redbelly.ece.ucdavis.edu**

   **viper.ece.ucdavis.edu**

   Note: More ECE system names can be found on Canvas as **ECE_hostnames.txt**

6. Run dotprod across 4 nodes using the following command. 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 –n option specifies how many proceses to run. The -npernode option specifies how many processes to run per machine. The ***--mca btl_tcp_if_include eno1*** option specifies which network interface to use. You may run into problems without it. After the run has finished, record the runtime into your log file.

   ```
   mpirun -npernode 2 -np 6 -hostfile MPI_HOSTS --mca btl_tcp_if_include eno1 dotprod_mpi
   ```

   Note: the dotprod executable must be within your home directory so it is visible from all machines in the cluster; do not use a working directory in /tmp.
7. Modify dotprod_mpi.c to use a vector length of 10,000,000, compile and rerun. Record the runtime in your log file. How much did the runtime increase from increasing the workload? Can you explain why this is the case?

## SUBMISSION

Pretty easy for the warmup. Use Canvas to submit the log file with your observations and answers from the steps in both Pthreads and MPI. This should be a page long at most.

**DUE DATE: Wednesday 5/2 at 10AM.**