Prof. Jingke Li (FAB 120-06, li@cs.pdx.edu); Class: MW 2:00-3:50pm @ FAB 40-07; Office Hr: MW 1-2pm & by appt.

Lab 1: Getting Started

For this and all future labs, you will be working on the CS linux systems. You can either use a lab machine, or remotely log on to one (connect to `linuxlab.cs.pdx.edu`).

Download and unzip the file `lab1.zip` from D2L. You'll see a `lab1` directory with a `mpihosts` file and multiple versions of the sum program:

```
sum.c sum-pthd.c sum-omp.c sum-mpi.c sum1.chpl sum2.chpl
```

Checking Your Environment

- Check to make sure you have the latest version of `gcc`:

  ```
  linux> gcc --version
  gcc (Ubuntu 4.8.4-2ubuntu1~14.04.1) 4.8.4
  ```

- Check to make sure you have access to the MPI compiler, `mpicc`:

  ```
  linux> which mpicc
  /usr/bin/mpicc
  ```

- Use `addpkg` to add the Chapel compiler to your environment:

  ```
  linux> addpkg
  ...
  ```

  Select `chapel-1.12`; exit, logout, then re-login.

Compiling the Programs

- To compile Pthreads programs, use `gcc` with `-pthread` flag:

  ```
  linux> gcc -pthread -g -o sum-pthd sum-pthd.c
  ```

- To compile OpenMP programs, use `gcc` with `"-fopenmp"` flag:

  ```
  linux> gcc -fopenmp -g -o sum-omp sum-omp.c
  ```

- To compile MPI programs, use `mpicc`:

  ```
  linux> mpicc -g -o sum-mpi sum-mpi.c
  ```

- To compile Chapel programs, use `chpl`:

  ```
  linux> chpl -g -o sum1 sum1.chpl
  ```

Running Pthreads and OpenMP Programs

These programs are run just like regular C programs:

```
linux> ./sum-pthd
linux> ./sum-omp
```

Note that for these two examples, the number of threads is hardwired in the program. In the future, we'll see how to make that adjustable.

Exercise Add a `printf` statement in the `worker()` routine in `sum-pthd.c` to show the id and the work range of each individual thread.

You may want to place this statement under a control flag:

```
#ifdef DEBUG
  printf(...);
#endif
```

This way, the same program can be compiled to two different versions:

```
linux> gcc -pthread -g -o sum-pthd sum-pthd.c                 # regular version
linux> gcc -DDEBUG -pthread -g -o sum-pthd-debug sum-pthd.c  # debug version
```

It is not as easy to do the same in the OpenMP program. We'll defer it to the future.

Running MPI Programs

Before running MPI programs, you need to setup a host file. Copy `mpihosts` to your home directory, and set the following env variable:

```
setenv OMPI_MCA_orte_default_hostfile ~/mpihosts
```

You should include this line in your shell startup file (*e.g.* `.cshrc`) to avoid typing it in every time.

An MPI program is run with `mpirun`, with a flag `-n <#copies>` indicating the number of copies you'd like to execute:

```
linux> mpirun -n 4 ./sum-mpi    // running 4 copies of the program
```

Note that for this program, the number of program copies is specified externally at the time of execution.

Exercise Add a `printf` statement in `sum-mpi.c` to show the rank and size of the current copy of the program.

Running Chapel Programs

For running Chapel programs, you need to set the following env variables in your shell startup file:

```
setenv CHPL_HOME /pkgs/chapel/chapel-1.12.0
setenv CHPL_HOST_PLATFORM linux64
setenv CHPL_COMM gasnet
setenv GASNET_SPAWNFN S
setenv GASNET_SSH_SERVERS "bevatron boson ..."  # list of host names
setenv SSH_CMD ssh
```

A Chapel program is run with a flag `-nl <#locales>` indicating the number of locales you'd like to use:

```
linux> ./sum1 -nl 1        // running the program over 1 locale
```

Exercise In both programs, the problem domain size `N` is a configurable constant. Try to change it at the time of execution:

```
linux> ./sum1 --N=10 -nl 1
```