In the following, $HOMEDIR$ refers to the directory:

/afs/cs.cmu.edu/academic/class/15740-f09/public

and $ASSTDIR$ refers to the subdirectory $HOMEDIR$/$asst/asst2$.

1 About the Machines

We will be using two parallel machines that support OpenMP this semester: $cobalt$ at the National Center for Supercomputing Applications (NCSA), and $pople$ at the Pittsburgh Supercomputing Center (PSC).

$cobalt$ is a collection of SGI Altix machines, which are built using Intel Itanium 2 processors. There is a smaller cluster ($co$-$login1$) for interactive use: you will be logging onto these machines to compile and test your code. Once your code is ready for a timing measurement, you will submit it to a batch queue to run on the 512-processor SGI Altix machine ($co$-$compute1$). There is a wealth of information about $cobalt$ and how to use it at the following web site:

http://www.ncsa.uiuc.edu/UserInfo/Resources/Hardware/SGIAltix/

$pople$ is a SGI Altix 4700 NUMA shared-memory system. It has 192 blades, where each blade has 2 Itanium2 dual-core processors (768 cores total). The 4 cores per blade each share 8 GB of memory, though the local memory per processor is visible to all other processors via the interconnect. Details about $pople$ can be found here:

http://www.psc.edu/machines/sgi/altix/pople.php

2 Logging In

You should use $ssh$ to connect to the front-end machines at both NCSA and PSC. In particular, type the following to log onto the interactive $cobalt$ machine ($co$-$login1$) at NCSA:

> ssh yourUserName@cobalt.ncsa.uiuc.edu

Type the following to log into $pople$:

> ssh youUserName@pople.psc.edu

(Alternatively, it is possible to use your TeraGrid Portal login to ssh to these machines. This method is described at http://www.teragrid.org/userinfo/getting_started.php?level=new_account, under the heading “Single Sign-on (SSO) with a TeraGrid Certificate Authority (CA) - Default”. You are welcome to try this method, but the course staff will not be providing support if there are problems.)
3 Compiling your programs

Because the OpenMP standard is supported on both cobalt and pople, you should not need to modify your source code to run on either platform. Since these machines were designed by different vendors and have different processors, compilers, etc., you do need to compile your code with different compilers and different compiler flags. You can find several trivial OpenMP C code examples in ASSTDIR/examples, along with a Makefile for each machine: i.e. ASSTDIR/examples/Makefile.pople and ASSTDIR/examples/Makefile.cobalt (there is also a ASSTDIR/examples/Makefile.warhol).

3.1 Compiling on cobalt and pople

To compile OpenMP programs on cobalt, use icc and specify -openmp to the compiler. For example, to compile a program hello.c with optimization, type:

> icc -openmp -O hello.c -o hello

4 Running your programs

For both cobalt and pople, you can run your program on the frontend nodes (i.e. the ones that you login to) by simply typing the program name at the command line directly (i.e. the way that you would normally run a program on a UNIX-based system.) This is a very useful way to debug your program, and to get some rough performance numbers.

To get a proper performance measurement on either machine, however, you need to submit your program to the batch queueing systems. When your program reaches the head of the batch queue, it will run with the machine (or your portion of the machine) to itself.

4.1 Submitting batch jobs on cobalt

To learn how to submit batch jobs on cobalt, please read the following web page:

http://www.ncsa.uiuc.edu/UserInfo/Resources/Hardware/SGIAltix/Doc/Jobs.html

4.2 Submitting batch jobs on pople

pople uses the Portable Batch Scheduler (PBS) system to control access to compute processors for batch jobs. Therefore you need to compose different job scripts for pople. Please see the following web page for details:

http://www.psc.edu/machines/sgi/altix/pople.php#running

Note that pople requires that jobs request CPUs in a multiple of 4; this may mean that if you are testing 1 or 2 thread jobs, you will have to request 4 CPUs but only use 2 threads.

In particular, search for the sample script that follows “A sample job script for an OpenMP program is”. You can also learn more about PBS by typing:

> man pbs