EECS 587 Discussion

10/30

OpenMP useful resources

A well explained tutorial, PLEASE READ: http://bisqwit.iki.fi/story/howto/openmp/

Official documentation, provides reference to some other important functions:
http://www.openmp.org/mp-documents/spec30.pdf

HelloWorld.cpp

```
#include <iostream>
#include <omp.h>

using namespace std;

int main(int argc, char *argv[])
{
  int th_id, nthreads;
  #pragma omp parallel private(th_id) shared(nthreads)
  {
    th_id = omp_get_thread_num();
    #pragma omp critical
    {
      cout << "Hello World from thread " << th_id << '\n';
    }
    #pragma omp barrier

    #pragma omp master
    {
      nthreads = omp_get_num_threads();
      cout << "There are " << nthreads << " threads" << '\n';
    }
  }
  return 0;
}
```

Login

We use greenfield, there are two ways to login.

1. use XSEDE account first connect to SSO:
`ssh <your_xsede_user_name>@login.xsede.org`

then connect to Greenfield

`gsissh greenfield.psc.xsede.org`

1.2 use PSC account
find PSC user name: 1) login XSEDE user portal, 2) go to [My XSEDE] > [Accounts], and then you can find the user name for your greenfield account. (might be different from XSEDE account, my XSEDE: ethanjyx, my PSC: yjiang5).

reset PSC password for first usage at: https://apr.psc.edu/autopwdreset/autopwdreset.html then connect to Greenfield:

`ssh <your_psc_user_name>@greenfield.psc.xsede.org`

Compile and run locally

```
g++ -o helloworld  -fopenmp helloworld.cpp
export OMP_NUM_THREADS=2
./helloworld
```

PBS script

We use PBS script to submit on greenfield.

example.sh:

```
#!/bin/csh

#PBS -l nodes=1:ppn=15

#  nodes must be 1, will fail if it is not

#  ppn must be a multiple of 15, because Greenfield processors each contain 15 cores

#PBS -l walltime=1:00

#  Combine standard output and error into one file

#PBS -j oe

#PBS -q batch

cd $HOME

#run my executable

setenv OMP_NUM_THREADS 4

./helloworld
```

Read more:

https://www.psc.edu/index.php/computing-resources/greenfield/job-scripts

There is a more detailed OpenMP job script on this page.

Submitting your job

```
qsub example.sh
qstat
qstat -a | awk '{print NR" "$0}' | grep $USER
(returns your position in the queue)
```

just like hw1

HW2 serial approaches

1. DFS

Use **`stack`**, or **`vector`** in C++ to maintain a stack

2. BFS

Use **`queue`**, or **`deque`** in C++ to maintain a queue

3. Priority queue

`priority_queue`

You can use **`set or unordered_set`** in C++ to keep track of visited / unvisited node ids.