An OpenMP Exercise on Blue Waters

Aaron Weeden, Shodor
2015
http://tinyurl.com/acca-cs-omp

If you have not already read the slides for Parallel Computing and OpenMP: Terminology and Examples, I recommend you do so first.

Also, if you have not already read A Blue Waters Usage Guide, I recommended you do so first.

The example used in this exercise uses many files. You may find it helpful to reference this flowchart to see how they all interact.

For this exercise, you will need two terminal windows connected to Blue Waters. To connect to Blue Waters, follow the instructions in the *Logging In* section of A Blue Waters Usage Guide. Then, open a second window and repeat the process. On Windows, you can open a second PuTTY window by simply launching the program a second time. On Mac, you can open a second window in Terminal by typing **Command-N**.

Request interactive job

In the first window, request an interactive job for 1 hour using 1 node with 32 cores. This can be accomplished using the following command:

```
qsub -I -l nodes=1:ppn=32:xe -l walltime=01:00:00<ENTER>
```

Request batch job

In the second window, request a batch job that will run the OpenMP version of the forest fire model five times each for thread counts between 1 and 32 on a single node with 32 cores. The example script will also generate an output file with each thread count and the average wall time it takes to run with that many threads:

With tabs:	`qsub ~awee<TAB>f<TAB>/o<TAB>s<TAB>p<TAB><ENTER>`
Without tabs:	`qsub ~aweeden/fire/omp/scale-omp.pbs<ENTER>`

This will give back a job ID. Check the status of the job using this command:

```
qstat <job id><ENTER>
```

You should notice that the status of the job (second-to-last column) is **Q**, i.e. it is waiting in the queue. You can run the same **qstat** command from time to time and notice the status change from **Q** to **R** (running) to **C** (complete).

Copy example code

While you wait for the job to finish, in the second window, copy the example fire model code directory into your home directory (if you have not already done so as explained in A Blue Waters Usage Guide). This can be accomplished using the following command:

With tabs:	**cp -r ~awee<TAB>f<TAB> ~<ENTER>**
Without tabs:	**cp -r ~aweeden/fire ~<ENTER>**

Confirm the example code directory is now in your home directory by listing the contents of your home directory:

```
ls ~<ENTER>
```

You should see a directory named **fire**. Change into this directory and into the **omp** directory it contains:

```
cd ~/fire/omp<ENTER>
```

Confirm you are now in the **omp** directory:

```
pwd<ENTER>
```

You should get back **/u/training/<your username>/fire/omp**

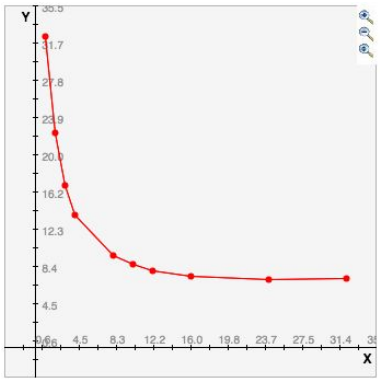
Create strong scaling plot

Inside the **omp** directory is a sample file that is the result of running a batch job like the one you submitted earlier. Open this file in the vi text editor:

```
vi scale-omp.out<ENTER>
```

The contents of the file are pairs of data: the number of OpenMP threads used and the average wall time in seconds to run the program with that many threads. Highlight the contents of the file using your mouse/trackpad. On Windows, this causes PuTTY to copy what you have selected. On Mac, press **Command-C** to copy.

Go to this website in a web browser: http://shodor.org/interactivate/activities/SimplePlot/. Click in the text box below the word **Data**, and paste. Click the button that says **Plot/Update**. You should get a graph that looks like the following:



Close the file in vi by typing the following:

```
:q!<ENTER>
```

Run interactively

Check on the status of your jobs:

```
qstat -u <your username><ENTER>
```

The interactive job will be named **STDIN**. If that job has a status of **R** (running), then follow the steps in this section. Otherwise, skip them for now and come back to them later once the job is running.

These steps should be followed in the first window, the window in which you requested an interactive job.

Change directories to the example OpenMP code directory:

```
cd ~/fire/omp<ENTER>
```

Decide how many OpenMP threads you want to use when running the fire model. There are 32 cores on the node you requested, so the number of threads should be between 1 and 32, inclusive. You can use higher numbers than 32, but then there will be more than 1 thread per core, which will probably hurt performance. Set the number of threads by running this command:

```
export OMP_NUM_THREADS=<number of threads><ENTER>
```

Run the OpenMP program, using the same number of threads you picked above:

```
aprun -n 1 -d <number of threads> ./fire-omp -r 1300 -c 1300 -t 1300<ENTER>
```

This will run for a few seconds (up to around half a minute) and print the final total percentage of trees that burned (100%) in a forest with 1300 rows, columns, and time steps.

Run the same command again, but this time keep track of how much wall clock time elapses by adding **time** to the front of the command (remember that you can type the up-arrow key to get back the same command you just entered):

```
time aprun -n 1 -d <number of threads> ./fire-omp -r 1300 -c 1300 -t 1300<ENTER>
```

This will print the **real**, **user**, and **sys** time. The **real** time is the wall clock time and is more reliable for measuring performance than the **user** and **sys** times.

Change the number of threads using the **export OMP_NUM_THREADS=** command and run again using **time aprun** with the new number of threads after the **-d** (remember you can use the up-arrow and down-arrow keys to scroll through your command history and enter commands again). What is the new run time?

If you keep changing the number of threads, do you get data similar to the data you saw in the **scale-omp.out** file?

There is a sample ASCII visualization of the fire model that you can view using the following command:

```
less fire-omp.out<ENTER>
```

To advance the visualization, hold down the spacebar. To rewind, hold **Control-B**. To jump to the beginning, type lower-case **g**. To jump to the end, type capital **G**. To quit, type **q**.

Generate your own ASCII visualization data by using the **-o** option followed by a filename that does not yet exist:

```
aprun -n 1 -d <number of threads> ./fire-omp -o fire-omp.2.out<ENTER>
```

This will create a file called **fire-omp.2.out**, which you can also view using the **less** command:

```
less fire-omp.2.out<ENTER>
```

Change the burn probability to 50% and generate a new visualization file:

```
aprun -n 1 -d <number of threads> ./fire-omp -b 50 -o fire-omp.3.out<ENTER>
```

This will create a file called **fire-omp.3.out**, which you can also view using the **less** command:

```
less fire-omp.3.out<ENTER>
```

Notice how the visualization is different when the burn probability is 50% as opposed to the default of 100%.

Change the number of time steps to be larger and generate a new visualization file:

```
aprun -n 1 -d <number of threads> ./fire-omp -b 50 -t 50 -o fire-omp.4.out<ENTER>
```

This will create a file called **fire-omp.4.out**, which you can also view using the **less** command:

```
less fire-omp.4.out<ENTER>
```

Once you are finished running the interactive job, you can end it by typing **Control-D** or the following command:

```
exit<ENTER>
```

Check the batch job's output

Check on the status of your jobs:

```
qstat -u <your username><ENTER>
```

The batch job will be named **scale-omp.pbs**. If that job has a status of **C** (complete), then follow the steps below. Otherwise, come back to these steps later once the job is complete.

Change back to your home directory:

```
cd<ENTER>
```

Confirm the output and error files from the job now appear in the list of files:

```
ls<ENTER>
```

You should see files in the list with **scale-omp.pbs.o<job ID>** (this is the output file) and **scale-omp.pbs.e<job ID>** (this is the error file).

You should also see a file called **scale-omp.out**. Open this file in the vi text editor (Instructions for using vi are available in A Blue Waters Usage Guide):

```
vi scale-omp.out<ENTER>
```

Confirm that the contents of the file look very similar to the **scale-omp.out** file you opened earlier.

Run another batch job

Change directories to the example code directory:

```
cd ~/fire/omp<ENTER>
```

Open the file **fire-omp.pbs** in vi:

```
vi fire-omp.pbs<ENTER>
```

This file is a script that requests a batch job that sets the number of OpenMP threads and runs the program. Initially it is set to run with 32 threads. You can change this number to a smaller number of threads by replacing all occurrences of the number 32 in the file with the number you choose. You will also want to change the expected wall time to be larger than the average wall time for the number of threads you chose that you saw when you created the strong scaling plot. The format for the walltime is **HH:MM:SS**.

Once you have made the changes, you can request a new batch job by running the following command:

```
qsub fire-omp.pbs<ENTER>
```

This will give back the job ID.

You can continue to monitor the status of the job using the following command:

```
qstat <job ID><ENTER>
```

Once the batch job is complete, open the output file it produced:

```
vi fire-omp.pbs.o<job ID><ENTER>
```

After the “prologue” it will tell you what percentage of the trees were burned. This should be 100%.

Close the output file without saving changes:

```
:q!<ENTER>
```

Open the error file for the job:

```
vi fire-omp.pbs.e<job ID><ENTER>
```

The **`time`** command writes to the error stream by default, so the **`real`**, **`user`**, and **`sys`** time will be shown in this error file. Compare the **`real`** time to the time you saw earlier in the **`scale-omp.out`** file for the corresponding number of threads.

This gives you one data point (or ⅕ of one if you are going to take an average of 5 runs for each data point). See if you can do multiple runs with different numbers of threads and generate data points like those found in the **`scale-omp.out`** file without using the automated **`scale-omp.pbs`** or **`scale-omp.sh`** scripts to create this file.

Either in an interactive or batch job, explore running the **`fire-omp`** executable with different parameters as explained in A Blue Waters Usage Guide**.**

If you wish to view the source code for the fire model, it is available in the file **`~/fire/omp/fire-omp.c`**.

To see how the OpenMP version of the code compares to the serial version, download this file and open it in a web browser. The serial version is shown on the left, and the OpenMP version is shown on the right, with the differences highlighted.