Exercise: Tiling

In this exercise, we'll use the same conventions and commands as in the previous exercises. You should refer back to the previous exercise description for details on various Unix commands.

In the exercise, you'll *benchmark* matrix-matrix multiplication algorithms, choosing various matrix sizes and various ways of performing the multiplication. *Benchmark* means to run timing tests.

Specifically, you'll benchmark the following methods:

- a naïve algorithm that performs the matrix-matrix multiplication the way you would normally do it by hand, which you'll run on various matrix sizes;
- a tiling algorithm, for which you'll be selecting not only various matrix sizes, but also various tile sizes;
- if you're running the Fortran90 version, the Fortran90 intrinsic routine `MATMUL`, for various matrix sizes.

1. If you aren't currently logged in to `boomer.oscer.ou.edu`, then log in; otherwise, change directory to your home directory:

   `cd  ~yourusername`

   OR

   `cd  ~`

2. Check to make sure that you're in your home directory:

   ```
   pwd
   /home/yourusername
   ```

   where `yourusername` will be replaced with your user name.

3. You should already have your own copy of the `PPCC2012` directory, as a subdirectory of your home directory. Check to make sure that you do:

   ```
   ls
   PPCC2012
   ```

   You should see a list of one or more files and subdirectories, including `PPCC2012`.

4. Change directory into your `PPCC2012` directory, like this:

   `cd  PPCC2012`

5. Make sure that you're in your `PPCC2012` directory, like this:

   ```
   pwd
   /home/yourusername/PPCC2012
   ```

6. List the contents of your `PPCC2012` subdirectory:

   ```
   ls
   Intro
   ```

7. Copy the subdirectory named `Tiling` from Henry's `PPCC2012` directory into your `PPCC2012` directory:

   `cp  -r  ~hneeman/PPCC2012/Tiling/  ~/PPCC2012/`

8. Confirm that the `Tiling` subdirectory was copied into your `PPCC2012` directory:

   ```
   ls
   Intro   Tiling
   ```

9. Change directory into your `PPCC2012` directory, like this:

   ```
   cd Tiling
   ```

10. Confirm that you're in your `Tiling` subdirectory:

    ```
    pwd
    /home/yourusername/PPCC2012/Tiling
    ```

11. See what files or subdirectories (if any) are in the current working directory (`Tiling`):

    ```
    ls
    C      Fortran90
    ```

12. Change directory into either your `C` subdirectory or your `Fortran90` subdirectory:

    ```
    cd C
    ```

    OR

    ```
    cd Fortran90
    ```

13. Confirm that you're in your `C` or `Fortran90` subdirectory:

    ```
    pwd
    /home/yourusername/PPCC2012/Tiling/C
    ```

    OR the output of the `pwd` command might be:

    ```
    /home/yourusername/PPCC2012/Tiling/Fortran90
    ```

14. See what files or subdirectories (if any) are in the current working directory:

    ```
    ls
    makefile         matmatmult.c          second_cpu.c  second_wall.c
    matmatmult.bsub  matmatmult_input.txt  second.h
    ```

    OR the output of the `ls` command might be:

    ```
    makefile         matmatmult.f90        second_cpu.c  second_wall.c  timings.h
    matmatmult.bsub  matmatmult_input.txt  second_cpu.f  timings.f
    ```

15. Make (compile) your executables. The command to compile will be:

    ```
    make
    ```

16. Edit the batch script `matmatmult.bsub` so that it contains your username and your e-mail address. (If you've forgotten how, see "Introduction to Using OSCER's Linux Cluster Supercomputer," section IV, pages 8-9.)

17. Edit the input file `matmatmult_input.txt` so that it contains your preferred problem size and type:
    a. The first three numbers are, respectively:
        i. the number of rows in the product matrix;
        ii. the number of columns in the product matrix;
        iii. the number of columns of the first matrix to multiply by, which is also the number of rows of the second matrix to multiply by.
    b. The next number is the preferred matrix-matrix multiplication algorithm:
        i. choose 1 for the naïve algorithm;
        ii. choose 2 for the tiling algorithm;
        iii. for Fortran90 only, choose 3 for the Fortran90 intrinsic routine `MATMUL`.
    c. If you've chosen the tiling algorithm, then the last three numbers are the dimensions of the tiles, which correspond to the dimensions of the matrices. (For other algorithms, the last three numbers are ignored.)
18. Submit the batch job:

    `bsub < matmatmult.bsub`

19. Once the batch job completes, examine the standard output file to see the timing for your run.
20. Run many more runs, on each of the available algorithms, each for several different problem sizes:
    a. Start by finding the largest problem that reports a runtime of zero seconds (that is, immeasurably small runtime).
    b. Work your way up in problem size to as big as you want, but the biggest problem you should do should be at most about 12 GB total. (Beyond that, your timings will become extremely long, because you'll spend most of your runtime swapping in and out of virtual memory swap disk, which would be very very bad).
    c. For the tiling algorithm, also try many different tile sizes, from very small up to the same size as the matrices.
21. When you've completed enough runs to satisfy yourself, use your favorite graphing program (for example, Microsoft Excel) to create a graph (or graphs) of your various runs, so that you can compare the various methods visually.
22. If you're feeling especially adventuresome, you can edit your `makefile` to incorporate the `–r8` option in the `CFLAGS` or `FFLAGS` macros, which means to do all real (floating point) variables in double precision instead of single precision. Then do:

    `make clean`

    Then compile again:

    `make`

23. Repeat the same timing experiments. (Be careful not to run problems that are too large, which will be easier to do in double precision, because you'll consume twice as many bytes for the same matrix dimensions.) Compare these new double precision runs against the original single precision runs.
24. You can also try running these experiments using the various compilers available on the Linux cluster supercomputer.