Multiplication of two square matrices of size n where n is a power of 2

Version 2 with cuda technology incorporating shared memory:
In order to enable coalescing of memory, we must use a shared memory kernel. Use one thread to compute each element of the solution matrix. The kernel must have all of the features from Version 1 plus the usage of shared memory setup dynamically. The code can be found in the textbook. Make sure the kernel you select coalesces memory.

Make use of command line arguments argv and argc. To automate the process of testing a variety of tile widths with a variety of matrix sizes, you must setup the executable code to accommodate the following command line arguments. To use nvvp, the nvidia profiler, I don't think you can use I/O redirection so you must use command line arguments.

```
cuda_mult_v2  <tile width> <input file> <output file>
```

cuda_mult_v2 – name of executable

tile width – 4, 8, 16, 32 (use square tiles – why is 32 the max?)

matrix file format– contains the size (N) of a square matrix and one N x N matrix. We will only consider values of n that are powers of 2. Initially, the possible values are 2^9 through 2^{14}. Side note: If you test value of N from 2^2 through 2^4 be ready for CUDA errors: invalid configuration.

Layout

1St number in the file – N – for a NxN matrix

Each line following will contain random numbers between 0.00 and 99.00. I will use a %6.2f format which will provide at least one space between all numbers. If you use the same format in the output you can use the diff command to check your answer.

What to submit?
1) The source code containing the kernel/device code. I will compile and supply my own data sets to test your kernel for correctness.

2) Report differences between version 1 and version 2.