Parallel Programming 2013/14 - Lab exercise 3

This lab introduces simple profiling techniques for CUDA programs. Please use the LG04 lab for this exercise (latest version of CUDA is malfunctioning on UG04 machines).

1. Download the following two programs into your working directory:

   `http://www.cs.bham.ac.uk/~hxt/2013/parallel-programming/matmul_plain.cu`

   `http://www.cs.bham.ac.uk/~hxt/2013/parallel-programming/matmul_tiled.cu`

   These two programs attempt to profile two CUDA implementations of the matrix multiplication algorithm[1]. The kernel in the second program is optimized with tiling (as discussed in the lecture). Each kernel is profiled when multiplying two square matrices with dimensions `1024 x 1024` each.

2. While increasing `BLOCK_DIM` for both the kernels in powers of two up to 32 (2, 4, 8, 16, 32), record the best running time for each kernel for each `BLOCK_DIM` (say, best of 3 invocations for each setting).

Kernel\BLOCK_DIM	2	4	8	16	32
Plain					
Tiled					

3. What is the optimal `BLOCK_DIM` for the plain kernel? What about the tiled kernel? Given these optimal configurations, how much speedup does the tiled kernel achieve?

4. Can you increase `BLOCK_DIM` beyond 32 for either of the kernels?

5. Download the following archives into your workspace:

   `http://www.cs.bham.ac.uk/~hxt/2013/parallel-programming/matmul_plain_nsight.zip`

   `http://www.cs.bham.ac.uk/~hxt/2013/parallel-programming/matmul_tiled_nsight.zip`

   These two archives contain two NVIDIA Nsight (Eclipse) projects corresponding to the kernels above.

6. Launch NVIDIA Nsight with the command `nsight` (make sure you have loaded the CUDA module beforehand). Once `nsight` is fully loaded (if it asks for a workspace, point it to an empty directory), load the *matmul_plain* project into the workspace with the following steps:

   Right-click anywhere within **Project Explorer** and select **Import**. On the import dialogue, select **General->Existing Projects into Workspace** and click **Next**. Click the option **Select archive file** and point to *matmul_plain_nsight.zip*. Select *matmul_plain* on the **Projects** listing and click **Finish**.

7. With the *matmul_plain* project selected in **Project Explorer**, click **Project -> Build All**. Once the build completes successfully, click **Run -> Profile As -> Local C/C++ Application**. This will trigger a perspective switch in the IDE (from C/C++ development mode to profiling mode). Don't be alarmed, let it do its thing.

8. Get familiar with the profiling perspective. Experiment with the various profiling metrics available (play around a bit).

9. Click on the `MatMulKernel` timeline, enable the **Analysis** tab (if not already visible). Profile the application once more to analyse the kernel memory utilization (click on the chart button next to the **Kernel Memory** entry).

10. Try to interpret the kernel memory profiling results (with the help from the documentation and the instructor).

11. Repeat the same steps for the *matmul_tiled* project and compare the results. (NOTE:- You can switch back into C/C++ development perspective from a button located at the top right corner of the IDE).

12. Experiment with other available profiling metrics (**Kernel Compute**, **Memory Access Pattern** etc.)

References

[1] NVIDIA. CUDA C Programming Guide. Available at `http://docs.nvidia.com/cuda/cuda-c-programming-guide#shared-memory`, 2013.