Benchmarking Fermi Microarchitecture

Paolo Ienne, Andrea Miele
Ewaida Moshen, Clément Humbert, Tristan Overney

November 12, 2014

1 Goals

The goals of this research is to expose the Fermi microarchitecture details as implemented in Nvidia Fermi cards such as: pipeline length, instructions latency, scheduling patterns.

2 Methods

To achieve the aforementioned goals, a serie of specially crafted CUDA kernels were used. These usually contain large batches of dependent instructions that were timed with the assistance of the `clock64()` function offered by the CUDA API.

The benchmark programs have been ran on a machine equipped with an: Nvidia GeForce GTX 580.

3 Integers multiplication

This section contains the results obtained through the previously described methods using large batches of integer multiplications.

3.1 Benchmark running times against number of threads

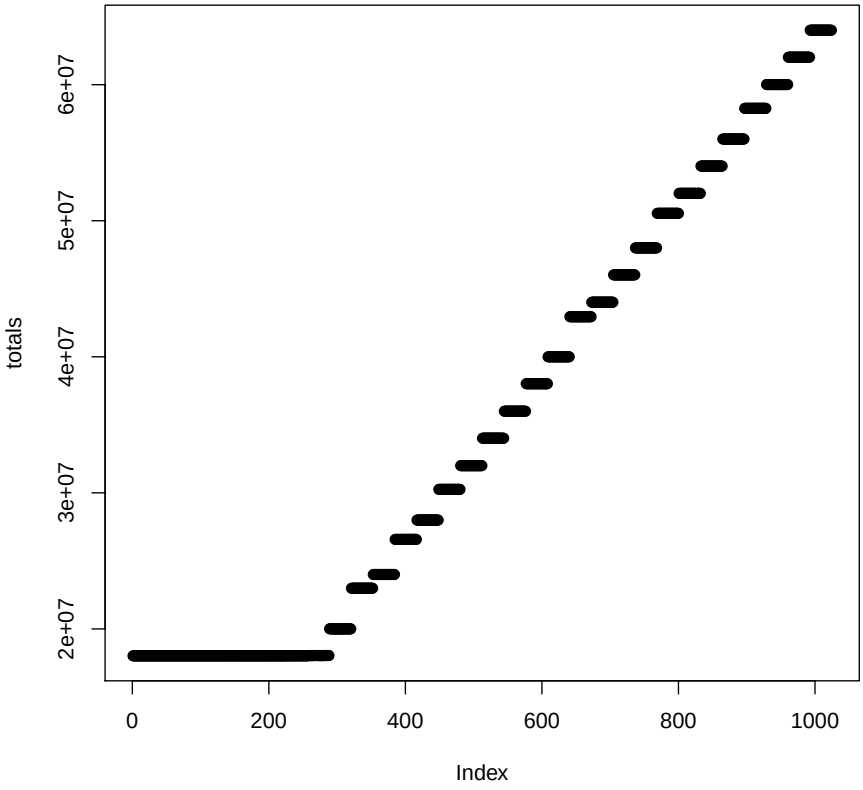

Figure 1: Running times of benchmark against number of threads

3.2 Benchmark running times divided by number of multplications

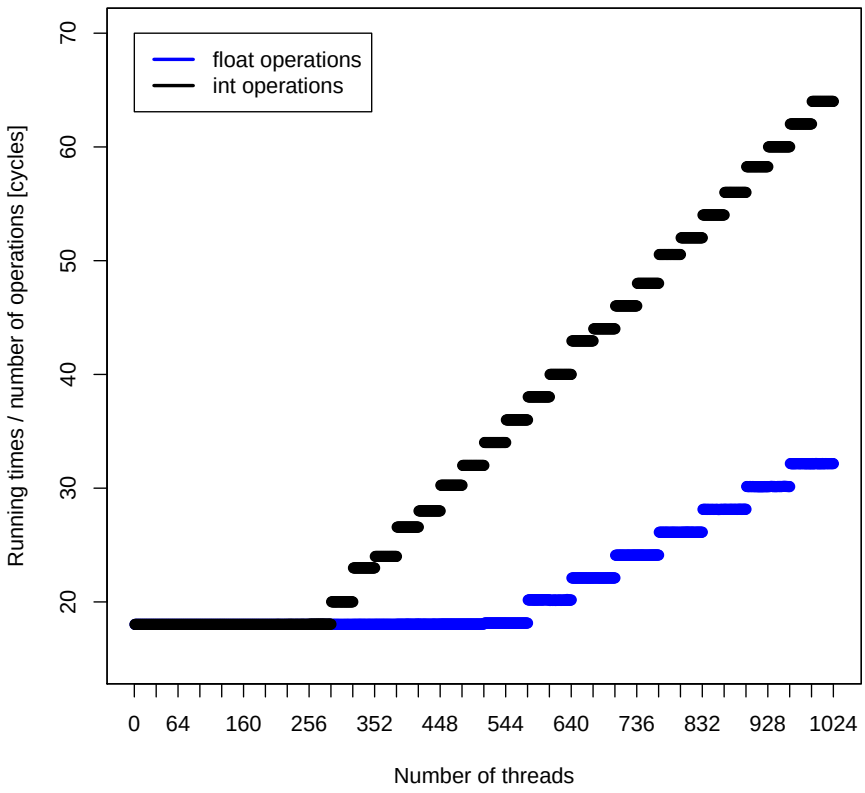

Figure 2: Running times of benchmark divided by number of operations against number of threads

4 Mixing floating points and integer multiplication

4.1 Description of the experiment

Informations have been found which were implying that the throughput for integer multiplication was half the floating point multiplication throughput because only one of the two 16 cores group of an SM was provisionned with integer multiplier. The following result are an attempt to verify those informations.

4.2 Benchmark running times, 1 floating points for 1 integer multiplication

If indeed only 1 out of 2 cores group can run integer multiplication then adding the same amount of multiplication but in floating point should not increase the total time spent executing our multiplications as the floating point multiplication can be run on the other core group (the one that does not possess integer multiplication).

One million multiplication of each kind has been ran on 1 to 1024 threads to see if the results were comparable to the graph were there was only integer multiplication.

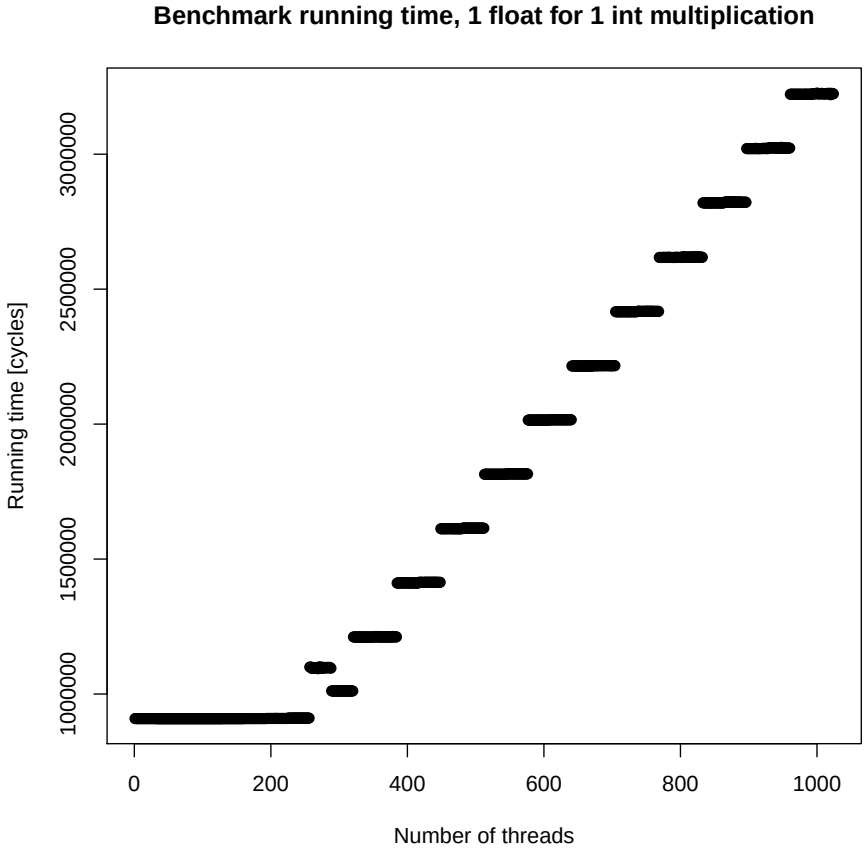

Figure 3: Integer/Floating point multiplication ratio: 1

4.3 Benchmark running times with mixed floating point and integer multiplications

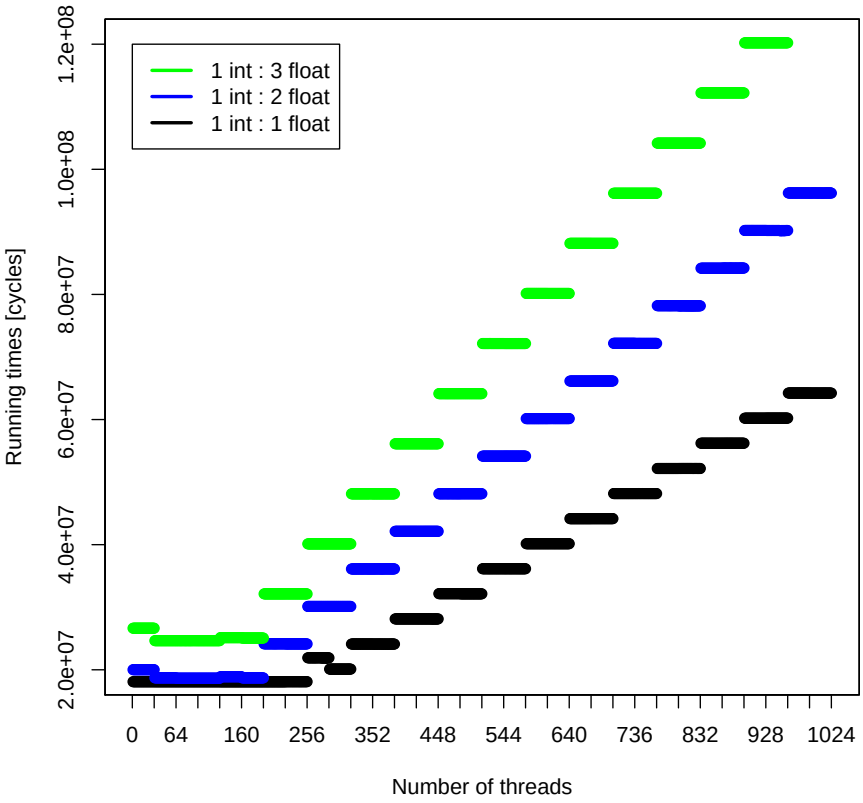

Figure 4: Running times of benchmarks with a mix of floating points and integers multiplications

5 Interpretation

6 Additionnal graphics

6.1 Integer multiplication: 1024 threads starting times

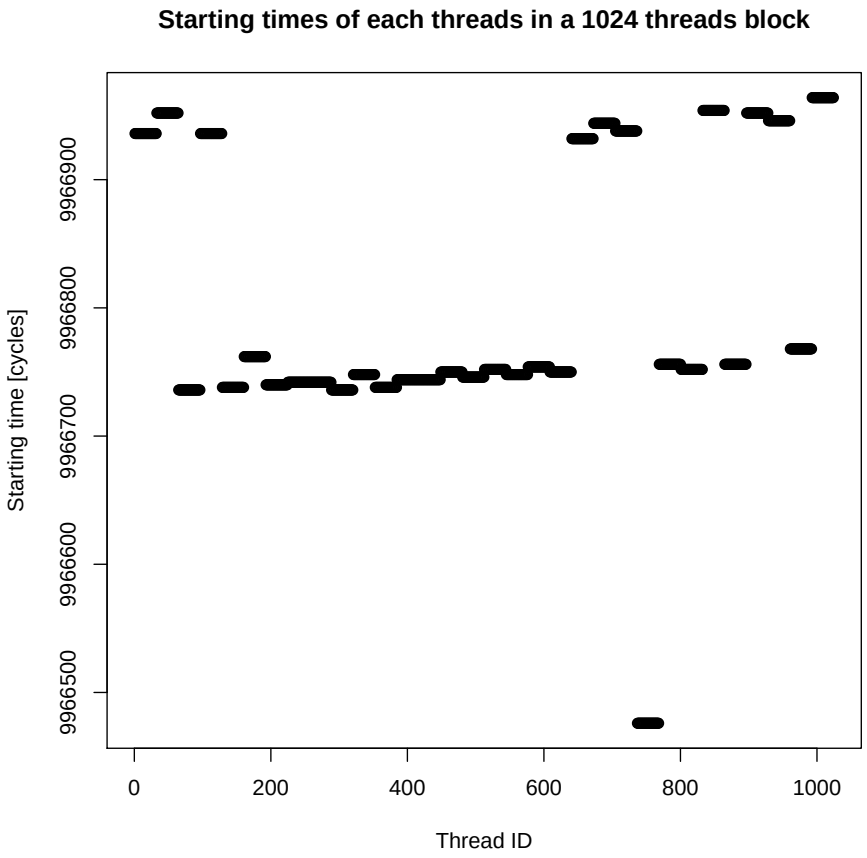

Figure 5: Order in which thread batches are started