TDT4200

Assignment 2

Sam Mathias Weggersen og Anders Lien

Part 1, Theory

Problem 1, Architectures & Programming Models

a)
i)
Nvidia Maxwell's cores execute in something Nvidia calls SIMT, which means Single Instruction, Multiple Thread. Much like classical SIMD processors, all cores in the same group execute the same instruction at the same time. The Maxwell GM107 has 640 cuda cores. All cores are also homogeneous. Maxwell is using a Streaming Multiprocessor (SM) called SMM, which has 4 warps, each with an instruction buffer and 32 cores.

ii)
ARM big.LITTLE has 2 different kinds of CPUs, a Little and a Big. Therefore, this is a hetrogeneous system. The Little CPU handles simple threads like UI, Camera, Call, send SMS etc. While the Big CPU will start more heavy threads, such as playing video, games, etc. The device will be much more energy efficiency when using the smaller CPU, as stated on ARMs website, "The latest big.LITTLE software and platforms can save 75% of CPU energy in low to moderate performance scenarios".

iii)
Vilje is a supercomputer located at NTNU. It has 22464 homogeneous cores. 1404 nodes that have 16 cores (dual eight-core) on each node. The nodes do the computation, while a cluster makes them look like a single system. We can look at Vilje as a single supercomputer. Each core is directly connected to each other, and we therefore have nonuniform memory access (NUMA).

iv)
Most modern-day CPUs have homogeneous cores. A typical high end processor, such as Intel i7, have 4 cores and 8 threads. Intels latest CPUs are using the Broadwell architecture, with 14 nanometer die shrink. Typical modern-day CPUs doesn't use NUMA, as NUMA is basically useless in a computer with at most one CPU.

b)
SIMT is an extension to the SIMD model found in Flynn's Taxonomy. SIMT is a parallel execution model that is intended to reduce instruction fetching overhead. When tasks wait for memory access, the processor will switch between tasks, reducing the overall 'experienced' latency in memory-access operations. SIMT is being used in Nvidia Maxwell GPUs. It is not realized in the other architectures.

c)
Nvidia Maxwell: SIMD with extensions, which is called the SIMT-classification
Vilje@NTNU: MIMD. Vilje uses a cluster of MIMD machines, like many other supercomputers.
ARM big.LITTLE & Modern-day CPU: MIMD. A modern-day cpu uses a MIMD architecture, where each processor can execute SIMD instructions.

Problem 2, CUDA GPGPUs

a)
Thread is simply the execution of a kernel with an index. Each thread uses its index to access elements in some datatype, such as an array. Therefore will threads cooperatively processes the entire data set.

Block is a group of threads. We can't say much about the execution of threads within a block, since the execution could be done concurrently, serially and in no particular order. Using the _syncthreads() function you can sync the threads so that it makes a thread stop at a certain point in the kernel until all the other threads in the same block reach that point.

Grid is composed of blocks which are completely independent, and does not synchronize between them.

b)
Calculation:
```
tot_gpu = 35n * log2(n) + 2n/r + 5n/r
tot_cpu = 3500n * log2(n)

35n * log2(n) + 7n/r = 3500n * log2(n)
3465n log2(n) = 7n/r
log2(n)  = (7n/r) / 3465n
log2(n) = (7/3465r)
n = 2^(7/3465r)
```

We see that if there is no bandwidth cost (r = ∞), the GPU would always be faster than the CPU.

c)
kernel2 will execute faster. In kernel1 the threads that are picked out are all from different blocks, while in kernel 2, blocks will run a task, which increases performance due to the locality principle.

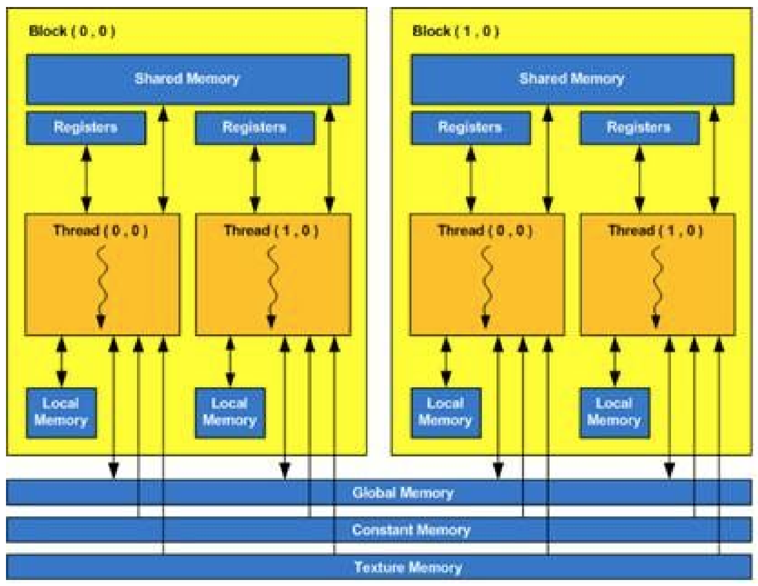
d)
i)
A warp is a grouping of threads that are scheduled to run concurrently.

ii)
Occupancy = Active Warps / Maximum Active Warps

iii)
Memory coalescing is taking multiple memory transactions and combining it into a single transaction. Instead of all threads just taking one element each from memory, one simply request all the following memory elements. This might require you to change the code so the next thread takes the following element in the list (memory).

iv)
Each thread has a local memory which resides in device memory (global memory for all blocks), but is accessible from that thread only.

v)
Each block on the device has a shared memory, see figure under. It's on-chip, therefore it is much faster than the off-chip global memory. Other blocks can't access this memory, and threads within the same block can cooperate by using it. Typically it is much smaller than the global (device) memory.



Part 2, Code

Problem 1, CUDA Intro

c)
Timings (s):
Transfer to device: 0.000117
Transfer from device: 0.000557
Total program run-time: 0.060780

To device/Total: 0.001925 ≈ 0.2%
From device/Total: 0.009164 ≈ 0.9%
(To device + From device)/Total: 0.011089 ≈ 1,1%

We could have implemented shared memory, as access to it is much faster than global memory access because it is located on chip.