PDS: Homework Assignment 1

Andreas De Lille

Note: I used qsub -I, so i worked on a shared node (there were no free nodes available)

Q1.1 : Gflops for N=1018

offset	**0**	**1**	**2**	**3**	**4**	**5**
10^9 Flops	1.46226	0.27762	0.553083	0.824286	1.08924	1.46226

There is a huge drop in performance once the offset is introduced. The cycle after a[i] = s * a[i-1] needs the value of a[i-1] (now a[i]). Since this value isn't calculated yet, the instruction has to wait. This becomes less problematic when the offset increases, e.g. a[i] = s * a[i-5] => the value corresponding with a[i-5] is only needed after 5 cycles thus the the stall time decreases.

Q2.1 There is a difference in performance due to the different cache sizes. Four vectors of size N=1024 can fit into the L1 cache (1024 double * 4 vectors * 8 bytes/double = 32Kbyte).
L2 cache is 256Kbyte large, so we can squeeze 4 vectors of size 8192 inside. The L3 cache can hold 4 vectors with size N = 655360. The high value for N=1 is due to the loop overhead.
Q2.2 SIMD instructions are only possible if there is no memory overlap. Otherwise the SIMD instructions would (try to) write 2 values simultaneously to the same memory space.
Q2.3: yes (with additional instructions to check for overlap)
Q2.4 Very small N : yes (increased loop overhead), bigger N : no
Q2.5 Yes (For large N however the slower caches and memory become the bottleneck.)
Q2.6 If the restrict keyword is absent then the compiler will make 2 versions. One with SIMD and one without. The program checks for overlap at runtime and uses SIMD whenever possible. If the restrict keyword is added then the compiler doesn't check for overlap at runtime.

Q3.1 strided: 0.2457 Gflops, linear: 1.96078 GFlops, BLAS: 6.25 GFlops
Q3.2: 0.936229 * 10^9
Q3.3: 49.6% -> Very high because 256 is too big to fit in the cache and is a power of 2. This results in equal cache line indexes and causes 'cache trashing'; the swapping of the cache lines.
Q3.4: 6.2% -> Much lower due to the fact that 255 isn't a power of 2, therefore the sequential columns have different cache indexes. The chance that they are still in cache is much higher.
Q3.5: 64*64*8 = 32KByte, this would fit in the L1 cache. There are still cache misses because 64 is a power of 2 which causes cache thrashing.
Q4.1 & 4.2

algorithm	**strided**	**linear**	**blas**	**blocked**
10^9 flops	0.795624	1.70032	17.5824	1.91617
D1 miss rate	56%	6.4%	17.1%	2.6%
explanation	loop order != save order	loop order = save order	More misses but hits are faster processed	smaller blocks fit in cache (spatial locality)

Q5.1: 6-7 hours