6.11 Real stuff: Benchmarking Intel Core i7 960 versus NVIDIA Tesla GPU

(Original section[1])

A group of Intel researchers published a paper [Lee et al., 2010] comparing a quad-core Intel Core i7 960 with multimedia SIMD extensions to the previous generation GPU, the NVIDIA Tesla GTX 280. The figure below lists the characteristics of the two systems. Both products were purchased in Fall 2009. The Core i7 is in Intel's 45-nanometer semiconductor technology while the GPU is in TSMC's 65-nanometer technology. Although it might have been fairer to have a comparison by a neutral party or by both interested parties, the purpose of this section is *not* to determine how much faster one product is than another, but to try to understand the relative value of features of these two contrasting architecture styles.

Figure 6.11.1: Intel Core i7-960, NVIDIA GTX 280, and GTX 480 specifications (COD Figure 6.22).

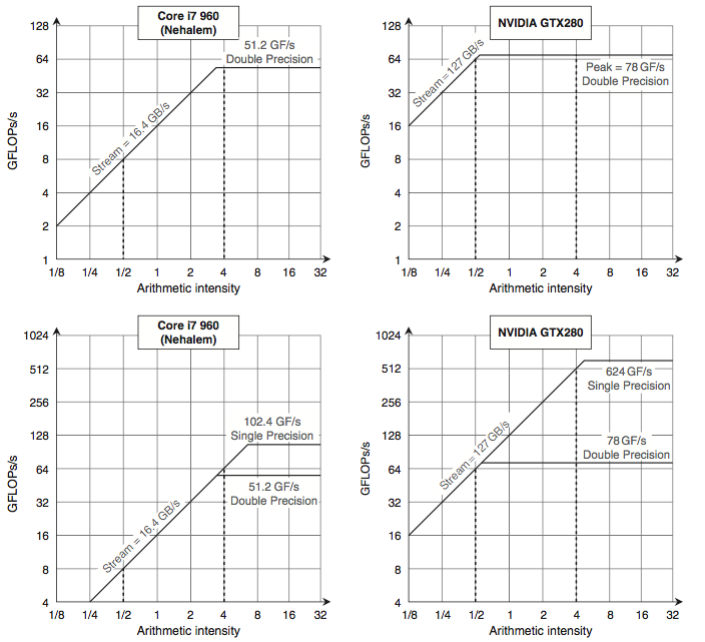
The rightmost columns show the ratios of the Tesla GTX 280 and the Fermi GTX 480 to Core i7. Although the case study is between the Tesla 280 and i7, we include the Fermi 480 to show its relationship to the Tesla 280 since it is described in this chapter. Note that these memory bandwidths are higher than in the figure below because these are DRAM pin bandwidths and those in the figure below are at the processors as measured by a benchmark program. (From Table 2 in Lee et al. [2010].)

	Core i7-960	GTX 280	GTX 480	Ratio 280/i7	Ratio 480/i7
Number of processing elements (cores or SMs)	4	30	15	7.5	3.8
Clock frequency (GHz)	3.2	1.3	1.4	0.41	0.44
Die size	263	576	520	2.2	2.0
Technology	Intel 45 nm	TSMC 65 nm	TSMC 40 nm	1.6	1.0
Power (chip, not module)	130	130	167	1.0	1.3
Transistors	700 M	1400 M	3030 M	2.0	4.4
Memory brandwith (GBytes/sec)	32	141	177	4.4	5.5
Single-precision SIMD width	4	8	32	2.0	8.0
Double-precision SIMD width	2	1	16	0.5	8.0
Peak Single-precision scalar FLOPS (GFLOP/sec)	26	117	63	4.6	2.5
Peak Single-precision SIMD FLOPS (GFLOP/Sec)	102	311 to 933	515 or 1344	3.0–9.1	6.6–13.1
(SP 1 add or multiply)	N.A.	(311)	(515)	(3.0)	(6.6)
(SP 1 instruction fused multiply-adds)	N.A.	(622)	(1344)	(6.1)	(13.1)
(Rare SP dual issue fused multiply-add and multiply)	N.A.	(933)	N.A.	(9.1)	–
Peak double-precision SIMD FLOPS (GFLOP/sec)	51	78	515	1.5	10.1

The rooflines of the Core i7 960 and GTX 280 in the figure below illustrate the differences in the computers. Not only does the GTX 280 have much higher memory bandwidth and double-precision floating-point performance, but also its double-precision ridge point is considerably to the left. The double-precision ridge point is 0.6 for the GTX 280 versus 3.1 for the Core i7. As mentioned above, it is much easier to hit peak computational performance the further the ridge point of the roofline is to the left. For single-precision performance, the ridge point moves far to the right for both computers, so it's considerably harder to hit the roof of single-precision performance. Note that the arithmetic intensity of the kernel is based on the bytes that go to main memory, not the bytes that go to cache memory. Thus, as mentioned above, caching can change the arithmetic intensity of a kernel on a particular computer, if most references really go to the cache. Note also that this bandwidth is for unit-stride accesses in both architectures. Real gather-scatter addresses can be slower on the GTX 280 and on the Core i7, as we shall see.

Figure 6.11.2: Roofline model [Williams, Waterman, and Patterson 2009] (COD Figure 6.23).

These rooflines show double-precision floating-point performance in the top row and single-precision performance in the bottom row. (The DP FP performance ceiling is also in the bottom row to give perspective.) The Core i7 960 on the left has a peak DP FP performance of 51.2 GFLOPs/sec, a SP FP peak of 102.4 GFLOPs/sec, and a peak memory bandwidth of 16.4 GBytes/sec. The NVIDIA GTX 280 has a DP FP peak of 78 GFLOPs/sec, SP FP peak of 624 GFLOPs/sec, and 127 GBytes/sec of memory bandwidth. The dashed vertical line on the left represents an arithmetic intensity of 0.5 FLOP/byte. It is limited by memory bandwidth to no more than 8 DP GFLOPs/sec or 8 SP GFLOPs/sec on the Core i7. The dashed vertical line to the right has an arithmetic intensity of 4 FLOP/byte. It is limited only computationally to 51.2 DP GFLOPs/sec and 102.4 SP GFLOPs/sec on the Core i7 and 78 DP GFLOPs/sec and 624 SP GFLOPs/sec on the GTX 280. To hit the highest computation rate on the Core i7 you need to use all four cores and SSE instructions with an equal number of multiplies and adds. For the GTX 280, you need to use fused multiply-add instructions on all multithreaded SIMD processors.



The researchers selected the benchmark programs by analyzing the computational and memory characteristics of four recently proposed benchmark suites and then "formulated the set of *throughput computing kernels* that capture these characteristics." The figure below shows the performance results, with larger numbers meaning faster. The Rooflines help explain the relative performance in this case study.

Figure 6.11.3: Raw and relative performance measured for the two platforms (COD Figure 6.24).

In this study, SAXPY is just used as a measure of memory bandwidth, so the right unit is GBytes/sec and not GFLOP/sec. (Based on Table 3 in [Lee et al., 2010].)

Kernel	Units	Core i7-960	GTX 280	GTX 280/ i7-960
SGEMM	GFLOP/sec	94	364	3.9
MC	Billion paths/sec	0.8	1.4	1.8
Conv	Million pixels/sec	1250	3500	2.8
FFT	GFLOP/sec	71.4	213	3.0
SAXPY	GBytes/sec	16.8	88.8	5.3
LBM	Million lookups/sec	85	426	5.0
Solv	Frames/sec	103	52	0.5
SpMV	GFLOP/sec	4.9	9.1	1.9
GJK	Frames/sec	67	1020	15.2
Sort	Million elements/sec	250	198	0.8
RC	Frames/sec	5	8.1	1.6
Search	Million queries/sec	50	90	1.8
Hist	Million pixels/sec	1517	2583	1.7
Bilat	Million pixels/sec	83	475	5.7

Given that the raw performance specifications of the GTX 280 vary from 2.5 × slower (clock rate) to 7.5 × faster (cores per chip) while the performance varies from 2.0 × slower (Solv) to 15.2 × faster (GJK), the Intel researchers decided to find the reasons for the differences:

- *Memory bandwidth.* The GPU has 4.4 × the memory bandwidth, which helps explain why LBM and SAXPY run 5.0 and 5.3 × faster; their working sets are hundreds of megabytes and hence don't fit into the Core i7 cache. (So as to access memory intensively, they purposely did not use cache blocking as in COD Chapter 5 (Large and Fast: Exploiting Memory Hierarchy).) Hence, the slope of the rooflines explains their performance. SpMV also has a large working set, but it only runs 1.9 × faster because the double-precision floating point of the GTX 280 is only 1.5 × as fast as the Core i7.
- *Compute bandwidth.* Five of the remaining kernels are compute bound: SGEMM, Conv, FFT, MC, and Bilat. The GTX is faster by 3.9, 2.8, 3.0, 1.8, and 5.7 ×, respectively. The first three of these use single-precision floating-point arithmetic, and GTX 280 single precision is 3 to 6 × faster. MC uses double precision, which explains why it's only 1.8 × faster since DP performance is only 1.5 × faster. Bilat uses transcendental functions, which the GTX 280 supports directly. The Core i7 spends two-thirds of its time calculating transcendental functions for Bilat, so the GTX 280 is 5.7 × faster. This observation helps point out the value of hardware support for operations that occur in your workload: double-precision floating point and perhaps even transcendentals.
- *Cache benefits. Ray casting* (RC) is only 1.6 × faster on the GTX because cache blocking with the Core i7 caches prevents it from becoming memory bandwidth bound (see COD Sections 5.4 (Measuring and improving cache performance) and 5.15 (Going faster: Cache blocking and matrix multiply)), as it is on GPUs. Cache blocking can help Search, too. If the index trees are small so that they fit in the cache, the Core i7 is twice as fast. Larger index trees make them memory bandwidth bound. Overall, the GTX 280 runs search 1.8 × faster. Cache blocking also helps Sort. While most programmers wouldn't run Sort on a SIMD processor, it can be written with a 1-bit Sort primitive called *split*. However, the split algorithm executes many more instructions than a scalar sort does. As a result, the Core i7 runs 1.25 × as fast as the GTX 280. Note that caches also help other kernels on the Core i7, since cache blocking allows SGEMM, FFT, and SpMV to become compute bound. This observation re-emphasizes the importance of cache blocking optimizations in COD Chapter 5 (Large and Fast: Exploiting Memory Hierarchy).

- *Gather-Scatter.* The multimedia SIMD extensions are of little help if the data are scattered throughout main memory; optimal performance comes only when accesses to data are aligned on 16-byte boundaries. Thus, GJK gets little benefit from SIMD on the Core i7. As mentioned above, GPUs offer gather-scatter addressing that is found in a vector architecture but omitted from most SIMD extensions. The memory controller even batches accesses to the same DRAM page together (see COD Section 5.2 (Memory technologies)). This combination means the GTX 280 runs GJK a startling 15.2 × as fast as the Core i7, which is larger than any single physical parameter in COD Figure 6.22 (Intel Core i7-960, NVIDIA GTX 280, and GTX 480 specifications). This observation reinforces the importance of gather-scatter to vector and GPU architectures that is missing from SIMD extensions.
- *Synchronization.* The performance of synchronization is limited by atomic updates, which are responsible for 28% of the total runtime on the Core i7 despite its having a hardware fetch-and-increment instruction. Thus, Hist is only 1.7 × faster on the GTX 280. Solv solves a batch of independent constraints in a small amount of computation followed by barrier synchronization. The Core i7 benefits from the atomic instructions and a memory consistency model that ensures the right results even if not all previous accesses to memory hierarchy have completed. Without the memory consistency model, the GTX 280 version launches some batches from the system processor, which leads to the GTX 280 running 0.5 × as fast as the Core i7. This observation points out how synchronization performance can be important for some data parallel problems.

It is striking how often weaknesses in the Tesla GTX 280 that were uncovered by kernels selected by Intel researchers were already being addressed in the successor architecture to Tesla: Fermi has faster double-precision floating-point performance, faster atomic operations, and caches. It was also interesting that the gather-scatter support of vector architectures that predate the SIMD instructions by decades was so important to the effective usefulness of these SIMD extensions, which some had predicted before the comparison. The Intel researchers noted that six of the 14 kernels would exploit SIMD better with more efficient gather-scatter support on the Core i7. This study certainly establishes the importance of cache blocking as well.

Now that we have seen a wide range of results of benchmarking different multiprocessors, let's return to our DGEMM example to see in detail how much we have to change the C code to exploit multiple processors.

(*1) This section is in original form.