CONTACT NAME
Chi Zeng: zeng@college.harvard.edu

Institute for Applied Computational Science
Harvard School of Engineering and Applied Sciences

Micro-benchmarking the C2070

R. Meltzer, C. Zeng, C. Cecka

HARVARD School of Engineering and Applied Sciences

Motivation

Understanding the memory hierarchy is important to optimizing the performance of any program. This is especially true for GPU programs, which are typically memory-bound. Beyond the sizes, little information is provided by Nvidia about the cache structures of the C2070. We investigated this cache structure, in order to verify details in the documentation and to discover those left out. We determined the size, associativity, and latency for both cache levels as well as latency for global memory and page faults.

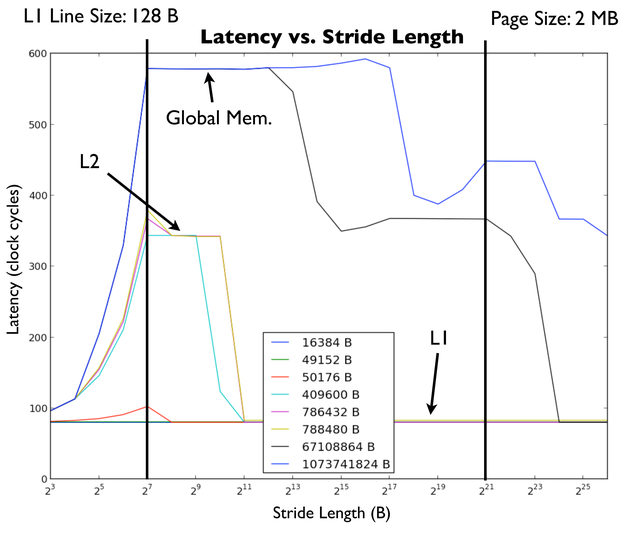
Confirming Cache Sizes

We performed stride analysis on the C2070. We altered the sizes of arrays and noted how varying the stride length affected latency. The arrays were initialized in Python with the stride pattern such that each access provided the index for the next access:

```
array = (numpy.arange(length) + stride) % length
```

Using Cheetah to unroll the loops, the most important section of the stride-analysis kernel is below. We used `$num_repeats = 128` and `$unrolling = 256`.

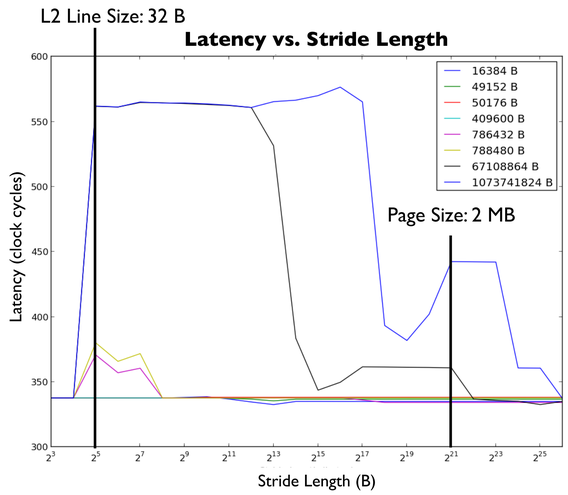
```
for(i = 0; i < $num_repeats; i++) {
  #for _ in xrange($unrolling):
  k = array[k];
  #end for
}
```



- L1 latency ~ 80 cycles
- L2 latency ~ 350 cycles
- Global memory latency ~ 580 cycles
- Latency plateau at 128 B stride indicates the line size, as lines are no longer reused on consecutive accesses.
- Latency spike for large strides plateauing at 2 MB indicates the page size.

Turning off L1

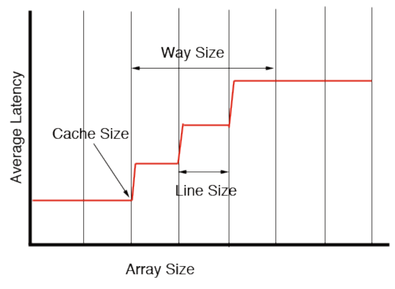
We repeated the analysis with the L1 cache turned off using the compiler tag `-Xptxas -dlcm=cg`



- The line size is 32 B instead of 128 B.
- L2 latency decreased, on average, 6-11 cycles because transferring data to the L1 cache is no longer needed.

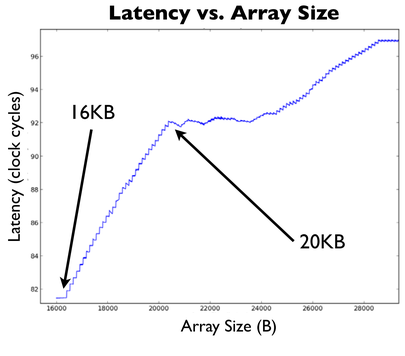
Determining Associativity

We measured and plotted average latency over array size to determine the associativities of various caches. The following illustration shows how we interpreted our graphs.



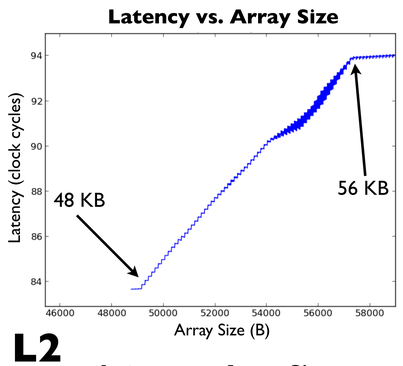
Using this method, we determined the structures of various caches. Our results are summarized in the table.

16 KB L1



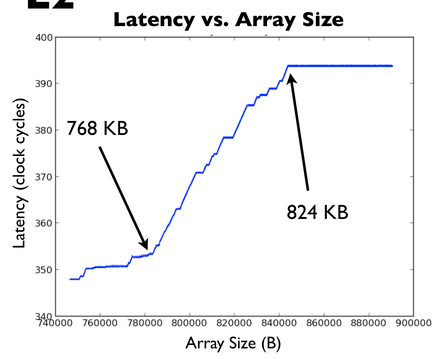
- Line size confirmed to be 128 B.
- Way size is 4096 B = 32 (number of sets) x 128 B (line size).
- Associativity is 4 = 16 KB (size of cache) / 4 KB (way size).

48 KB L1



- Line size confirmed to be 128 B.
- Way size is 8192 B = 64 (number of sets) x 128 B (line size).
- Associativity is 6 = 48 KB (size of cache) / 8 KB (way size).

L2



- Final plateau for L2 cache defines a way size of 57,344 B with 1792 sets.
- Could mean 14 separate caches each with 128 sets.
- On average, caches are 13.7-way set associative.
- Possibly indicates 10 14-way, 128-set associative caches and 4 13-way, 128-set associative caches.

Although at first unusual, the C2070 is "based on" the Fermi architecture that typically has 16 SMs rather than the 14 of the C2070. With the same structure and 16 caches, each would be 13-way associative.

Cache	Size (kb)	Latency (clock cycles)	Associativity	Number of Sets	Line Size (bytes)
Small L1	16	80	4-way	32	128
Large L1	48	80	6-way	64	128
L2	768	350	13 5/7-way	1792	32

Conclusion

We were able to verify the cache sizes provided in [4], with the L1 cache configurable to 16 or 48 KB, with a L2 cache of 768 KB. We determined the page size to be 2 MB.

The official documentation does not include such details as line size, page size, precise latency, or associativity for the varied cache levels. Our findings for these details are summarized in the table above. The data for the L2 cache was much noisier than the L1 cache data, but we were still able to draw reasonable conclusions about its structure.

Future Works

- Compare results to those of a different Fermi-architecture GPU.
- Perform similar analyses for Kepler- and/or Tegra-architecture GPUs.

References

Volkov, Vasily, and James Demmel. "LU, QR and Cholesky Factorizations using Vector Capabilities of GPUs." Electrical Engineering and Computer Sciences University of California at Berkeley (2008): n. pag. Print.

Wong, Henry. "Demystifying GPU Microarchitecture through Microbenchmarking." Department of Electrical and Computer Engineering, University of Toronto (2012): n. pag. Print.

NVIDIA Corp. NVIDIA CUDA Programming Guide, 4.3, 2012

NVIDIA Corp. "Nvidia's Next Generation CUDA Compute Architecture: Fermi.", 2009