Project 1

1. Latency at different memory level

The MLC does not naively support cache selection, but we can use the appropriate buffer size to enter the desired cache level.

(1) Test L1 cache latency
Command: s
mlc --idle_latency -b32 -c1
Output:
Using buffer size of 0.031MiB
Each iteration took 3.5 base frequency clocks (1.0ns)

(2) Test L2 cache latency
Command:
mlc --idle_latency -b1m -c1
Output:
Using buffer size of 1.000MiB
Each iteration took 14.2 base frequency clocks (4.1ns)

(3) Test L3 cache latency
Command:
mlc --idle_latency -b12m -c1
Output:
Using buffer size of 12.000MiB
Each iteration took 58.6 base frequency clocks (16.8ns)

(4) Memory latency
Command:
mlc --idle_latency -b32m -c1
Output:
Using buffer size of 32.000MiB
Each iteration took 269.4 base frequency clocks (77.1ns)

2. Max Bandwidth under different granularity and R/W ratio

1) 64B (default)
Command:
mlc --max_bandwidth -b32m -l64
Output:
ALL Reads: 30360.66
3:1 Read-Writes : 29274.96
2:1 Read-Writes : 28863.53
1:1 Read-Writes : 28317.75

2) 128B

Command:
mlc --max_bandwidth -b32m -l128
Output:
ALL Reads: 27851.47
3:1 Read-Writes : 27168.17
2:1 Read-Writes : 27067.55
1:1 Read-Writes : 24859.73

3) 256B
Command:
mlc --max_bandwidth -b32m -l256
Output:
ALL Reads: 25611.47
3:1 Read-Writes : 25114.40
2:1 Read-Writes : 24820.39
1:1 Read-Writes : 23348.85

An increased portion of write will decreases the effective maximum bandwidth.

3. Memory Latency vs Throughout

The mlc provide a built-in function to test the memory latency under difference load (b/w)
Command:
mlc --loaded_latency
Output:

Latency(ns)	Bandwidth (MB/s)
521.57	30398.8
516.52	30415.8
501.75	30401.6
461.86	30397.6
434.01	30223.3
404.28	29969.5
403.58	29717.2
399.24	28981.9
381.57	28449.3
240.27	27207.2
148.51	21937.2
121.56	16218.1
110.49	12815.9
105.49	10123.5
101.03	7205.9
98.20	5414.4
96.11	4020.0
93.57	2565.4

93.30	1533.8

Table 1: Read Latency vs. Memory Bandwidth.

We can plot the data to have a more straightforward view:

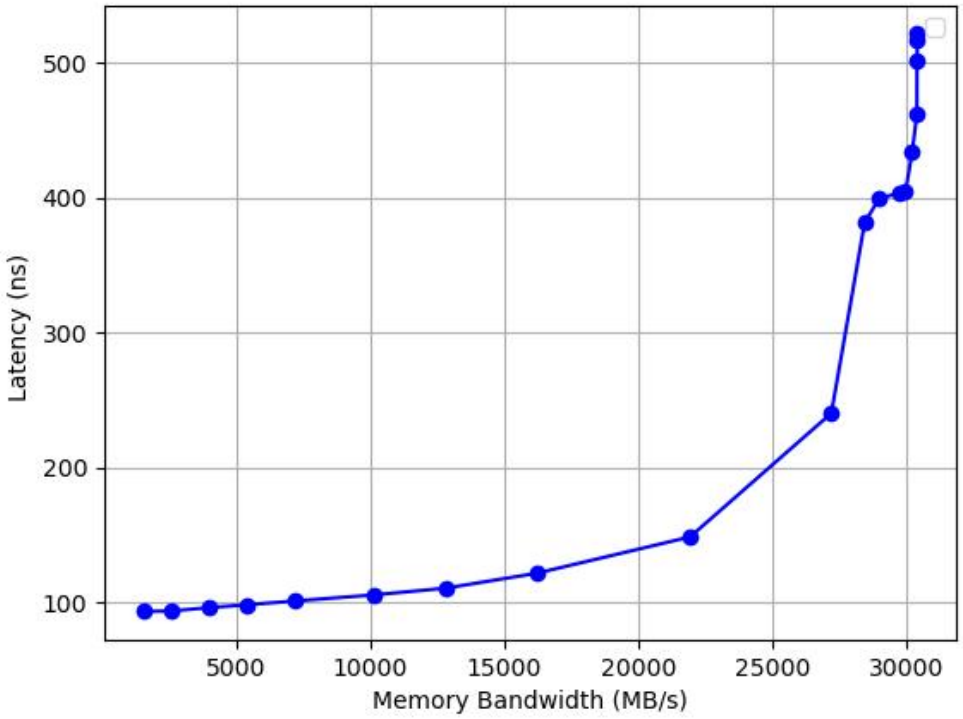

Figure 1: Read Latency vs. Memory Bandwidth.

The result aligns with the queuing theory that the increase of delay is highly non linear to the increase of utilization. As the bandwidth approaches its full capacity, the latency increases dramatically.

4. Cache Miss Ratio

Intel VTune Profiler is a powerful tool to analyze the performance of an application. In this experiment, I created 2 similar C++ programs that do matrix traverse. The only difference between the 2 programs is that one traverse sequentially and the other traverses randomly. The memory access section can gives the status of the memory and cache.

A. Sequential:

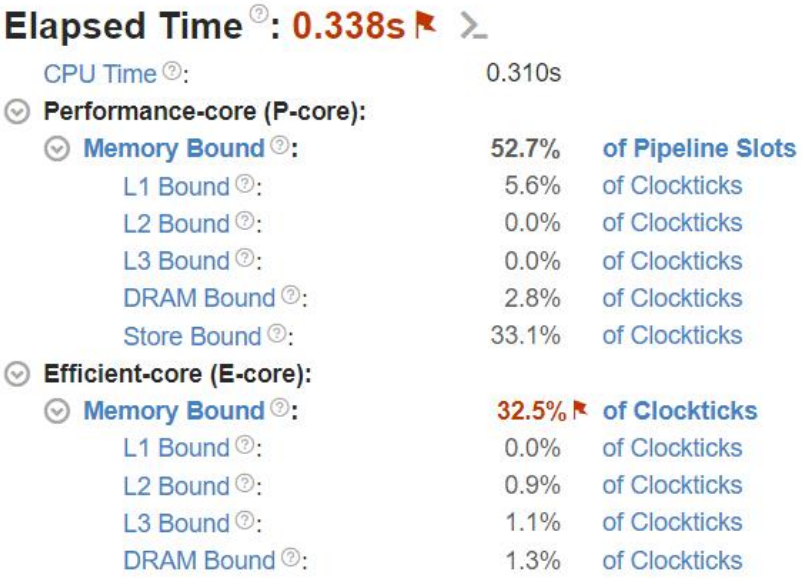

Figure 2: Cache Miss Test for Sequential Traverse

B. Random:

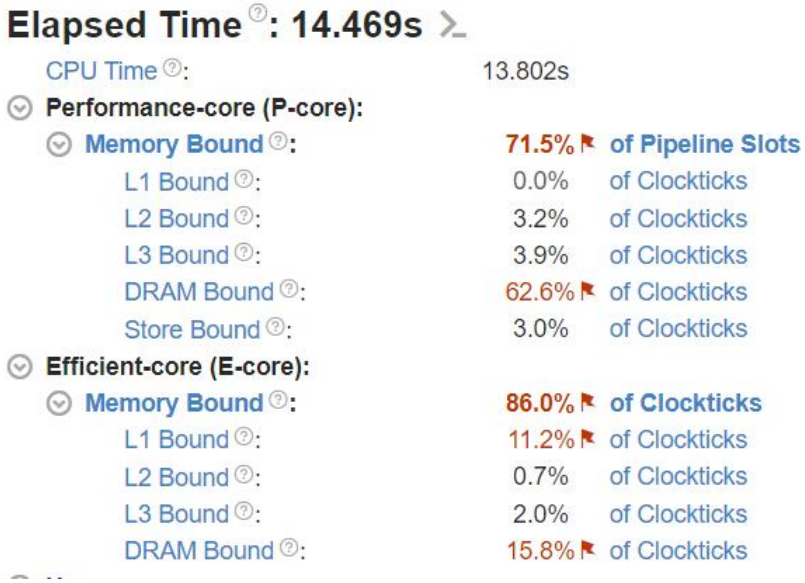

Figure 3: Cache Miss Test for Random Traverse

The column-major approach results in more memory bound, which is due to excessive loads (cache miss) and it takes much longer to complete. The results meet our exceptions. A high cache miss ratio will significantly slow down the program.

5. TLB Miss Ratio

In this experiment, I also created 2 similar C++ programs that do matrix traverse. The only difference between the 2 programs is that one uses a contiguous array and accesses elements continuously(A); the other accesses data with a stride equal to the number of integers per page.(B)

The Micro-architecture Exploration section can gives the status of the CPU.

	A	B (High TLB miss)
Time	0.602s	3.525s
Load STLB Hit	0.0% of Clockticks	0.0% of Clockticks
Load STLB Miss	0.0% of Clockticks	35.5% of Clockticks

Table 2. Result for TLB Miss Test.

Figure 4: VTuner Profiler Data for continuous access.

Figure 5: VTuner Profiler Data for cross-page access.

A higher TLB miss rate can also delay the execution of the program.