# Cache & Memory Performance Profiling

ECSE 4320

Your Name

## Content

Experiment Setup 2
Zero-Queue Latency 3
Pattern & Granularity Sweep 4
Read/Write Mix Sweep 5
Intensity Sweep 6
Working-Set Size Sweep 7
Cache-Miss Impact 8
TLB-Miss Impact 9

# Experiment Setup

**Timing Measurement:**

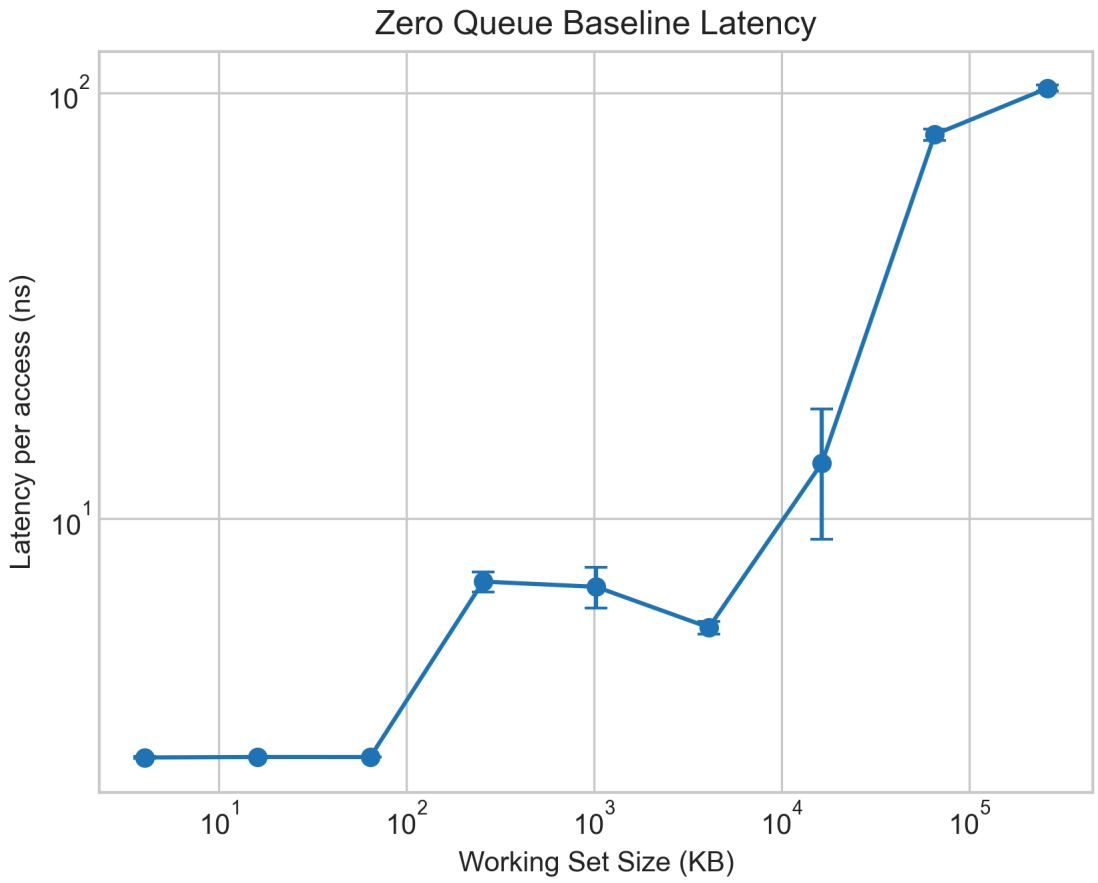
- Execution time is measured using `mach_absolute_time()`.

**Conditions:**

- Model: M2 Mac
- OS: Sequoia 15.6
- Powersource: Wall outlet
- Ram: 16 GB

## Zero-Queue Latency

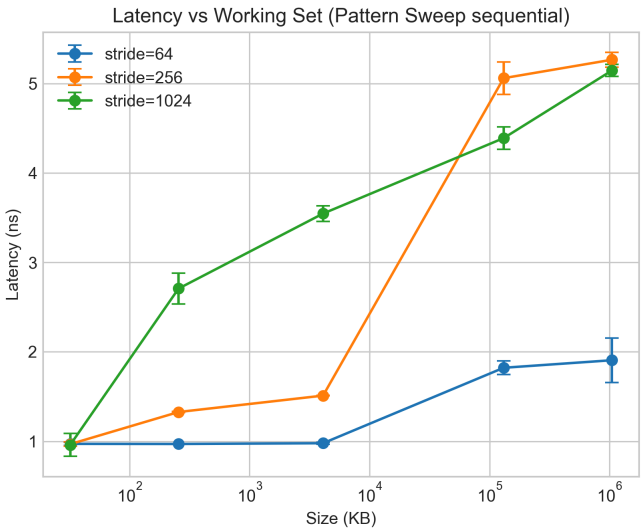
We can see in the when the data size is very small the (less than 64KB which is the size of L1 cache) that the time is constant. After that the time starts increasing because it needs to use L2 cache. After about 4MB it runs out of L2 and need to use L3 which is much slower.

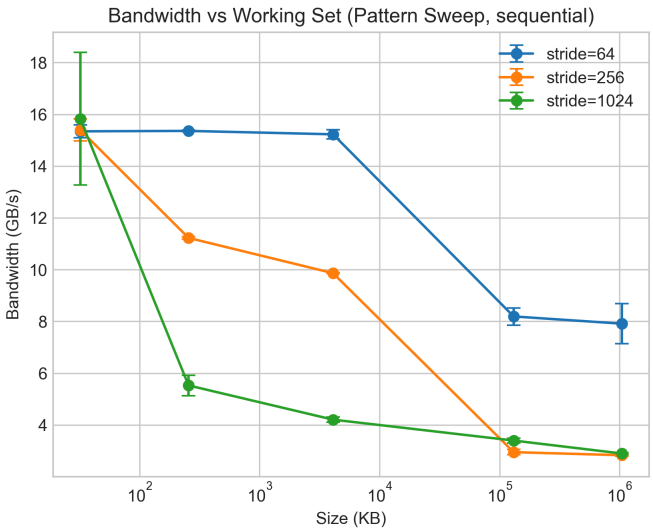


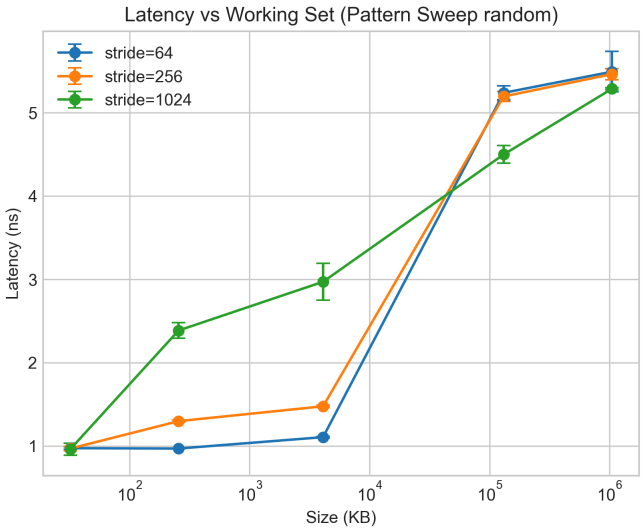
# Pattern & Granularity Sweep

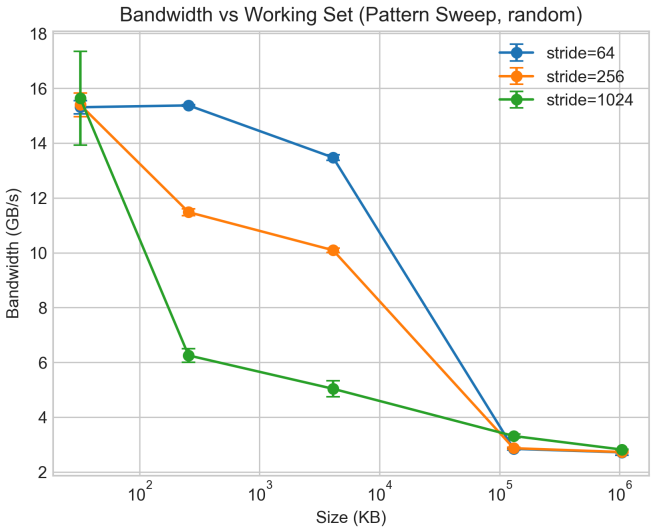
Evaluated sequential vs. random access patterns across strides (64B, 256B, 1024B).

We can see that the for large strides the sequential sweep looks the sam as the random sweep. Only the 64 byte stride seems to have a large performance boost which has a lower latency and higher bandwidth than everything else.



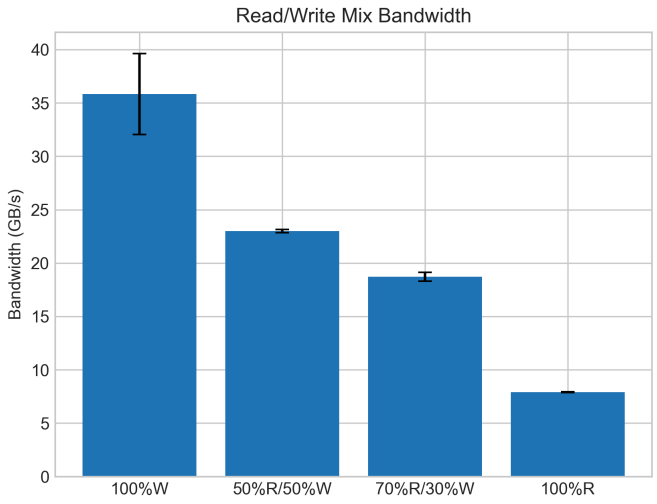



## Read/Write Mix Sweep

Tested 100%W, 50R/50W, 70R/30W, 100%R write ratios.

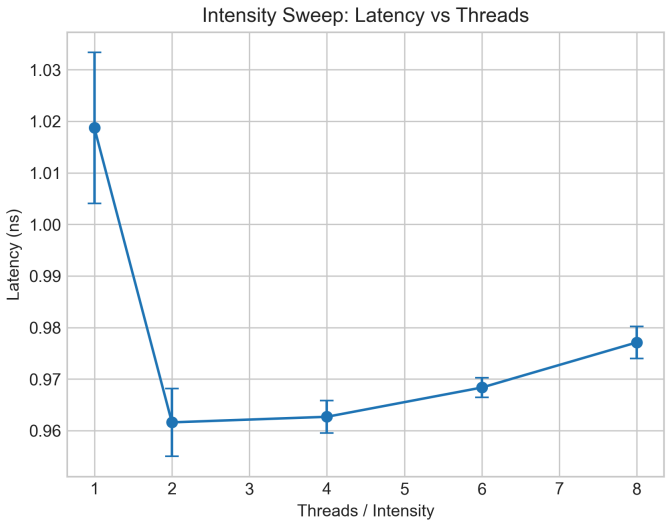
We can see that 100% writes achieve the highest bandwidth, while 100% reads got the lowest, this makes sense because reads require the core to wait. Mixed workloads are inbetween these two extremes.

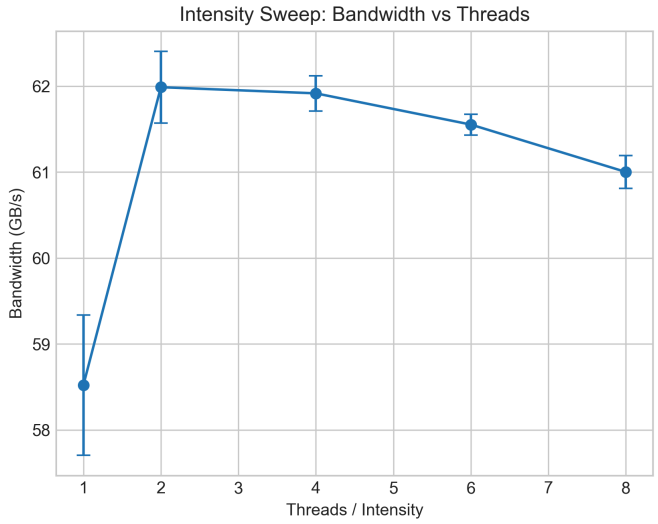


# Intensity Sweep

Loaded-latency sweep measuring bandwidth and latency with more threads.

Bandwidth increases from 1 to 2 threads, then saturates around 61-62 GB/s for higher thread counts. Mean latency per element decreases slightly with 2 threads, then rises gradually as threads increase further. The “knee” in the throughput-latency curve occurs at 2 threads.

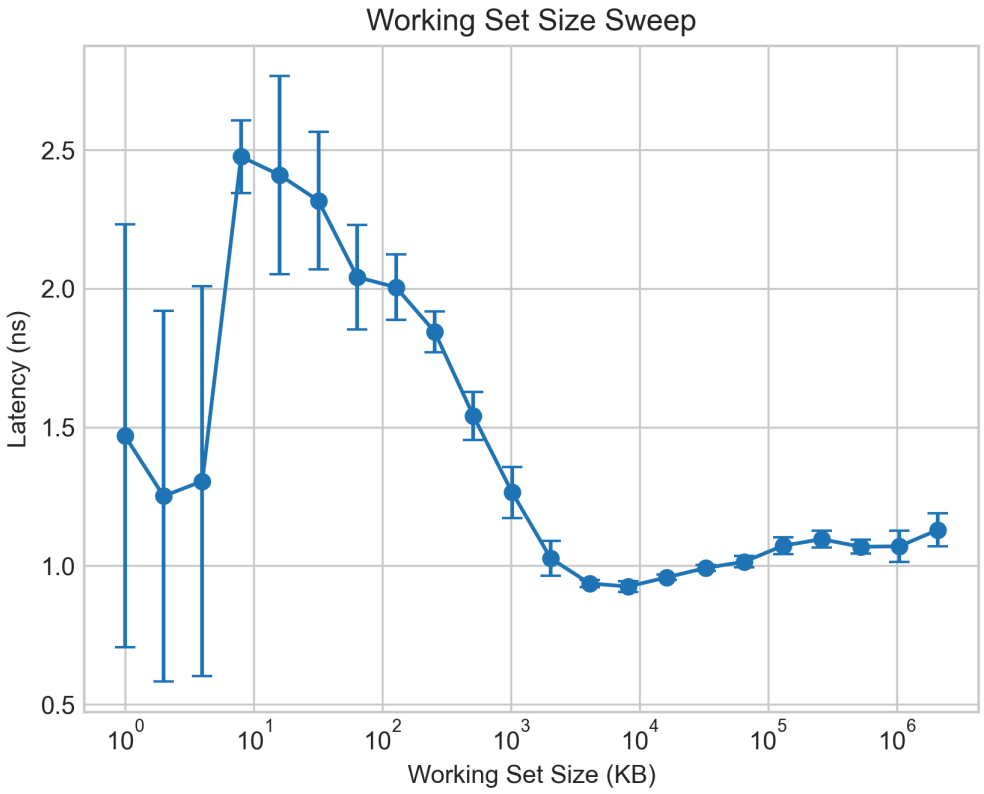


## Working-Set Size Sweep

Measured latency across increasing working-set sizes from 1 KB to 2 GB.

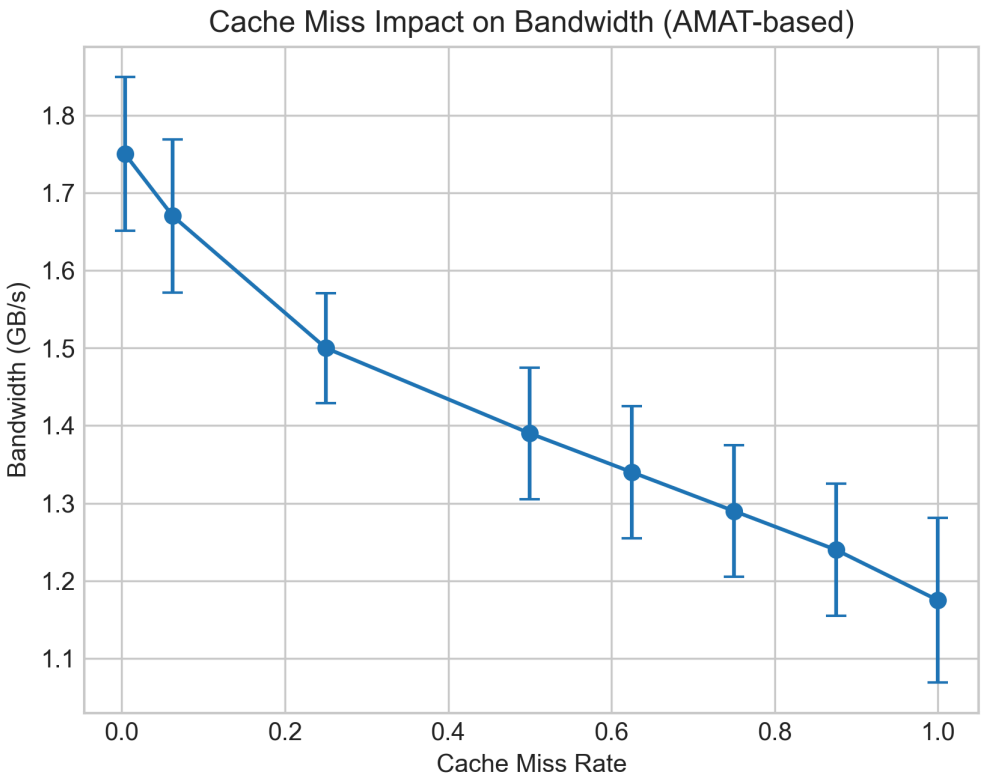
We can see that when the memory size is very small the latency is low. Before the size leaves L1 cache it greatly increase. It then slowly decreases till the L1 cache boundary at 64KB and starts decreasing at a faster rate. When it leaves L2 at 4MB the latency flattens out and then starts increase from then on.



## Cache-Miss Impact

Used lightweight kernel with controlled cache miss rates to measure performance sensitivity.

In the graph we can see that the performance decreases as cache-miss ratio increases. This makes sense because this would mean that the core is waiting longer for memory.



# TLB-Miss Impact

Varied page locality and used huge pages to measure TLB sensitivity.

We can see that the larger page sizes increased execution time while decreasing bandwidth. This is because allocations are more likely to be placed far away in memory and so decreases the chance of a cache hit.