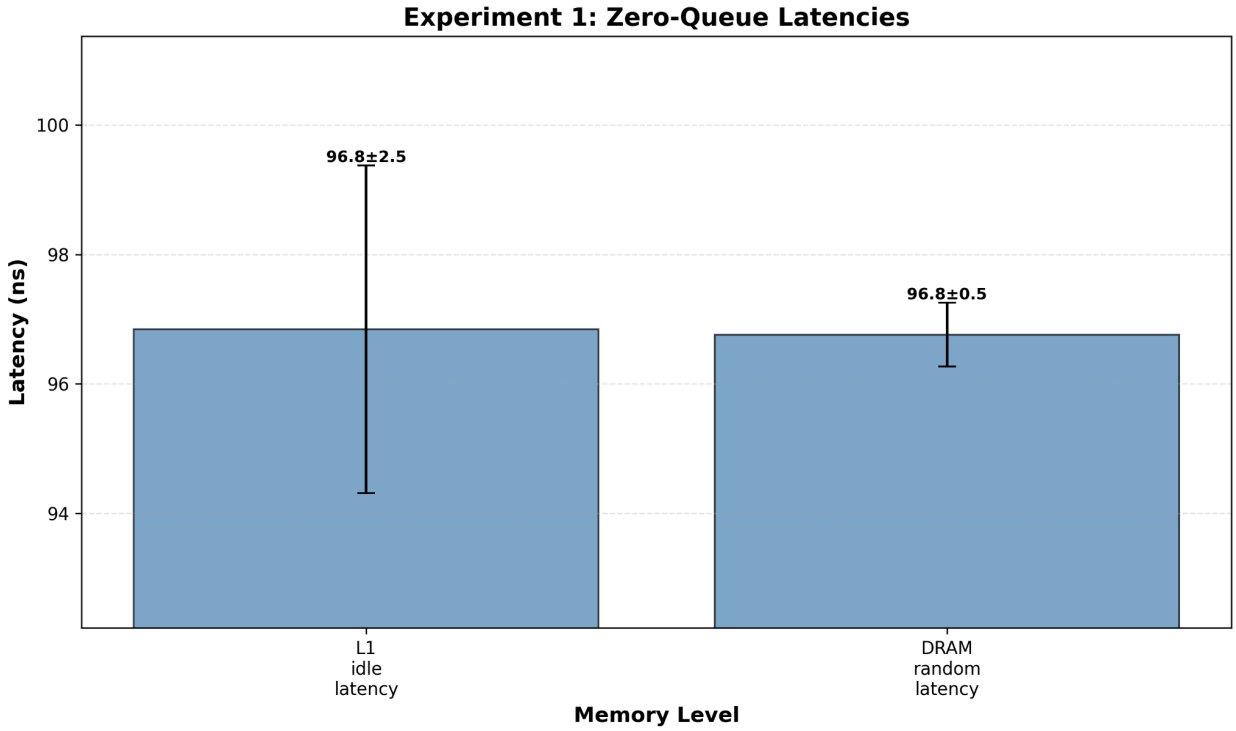
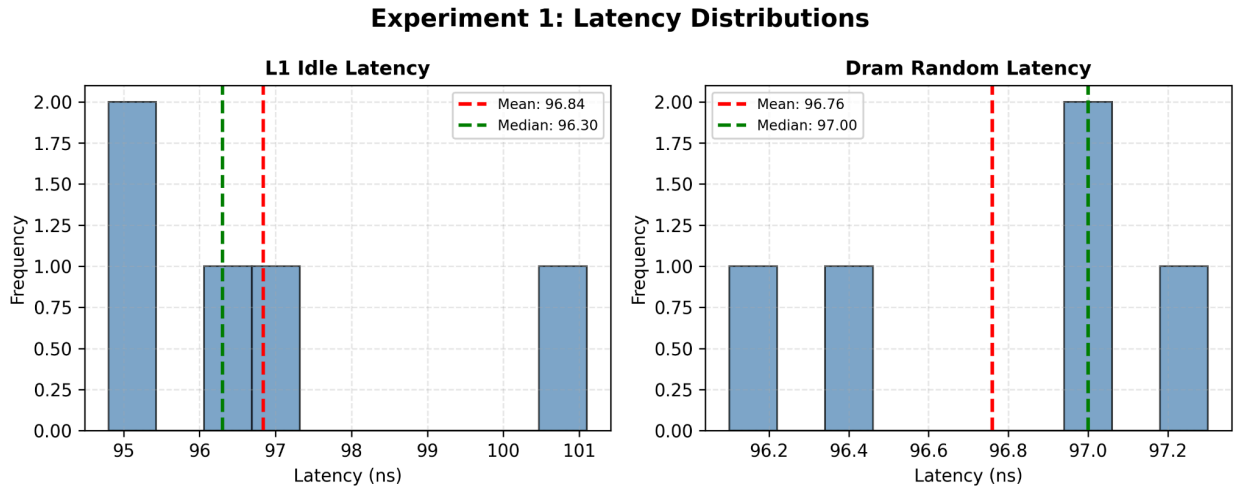
1. Zero-queue baselines

For the baselines, initial latency measurements were captured. As seen in the below diagram (and throughout the experiments), I was unable to get true cache latencies, and so measurement speeds tend to follow DRAM speeds and memory sizes. The below two graphs show that latency for both L1 cache and DRAM access was around 97 ns. The table below shows this comparison and converts the timing to CPU cycles (about 320 cycles for each).

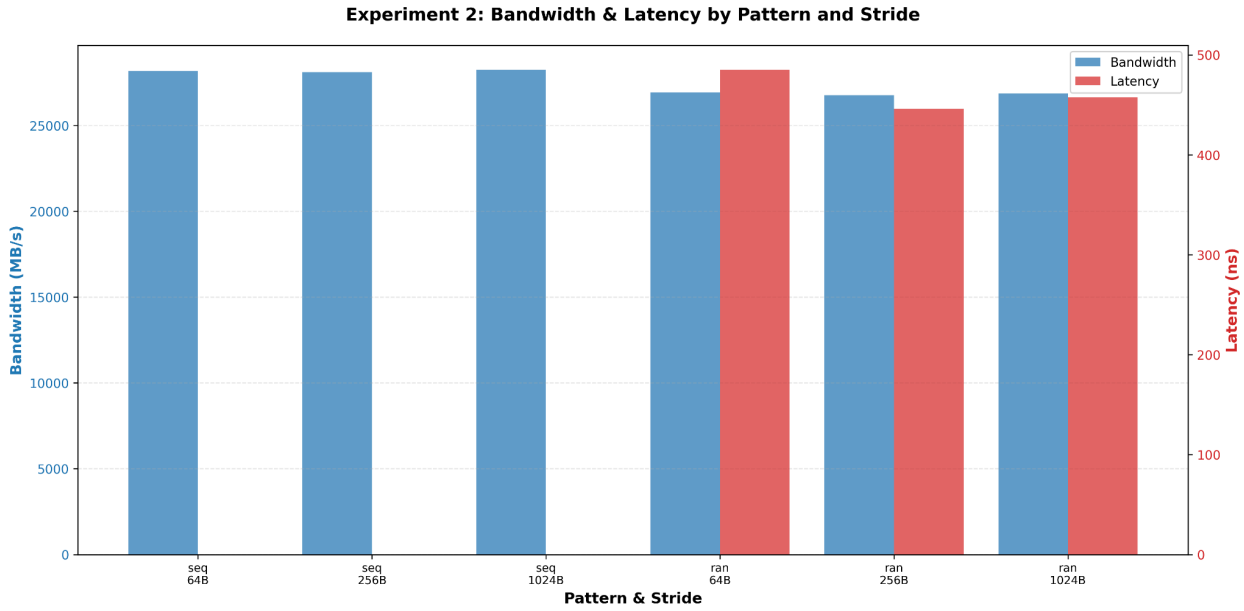


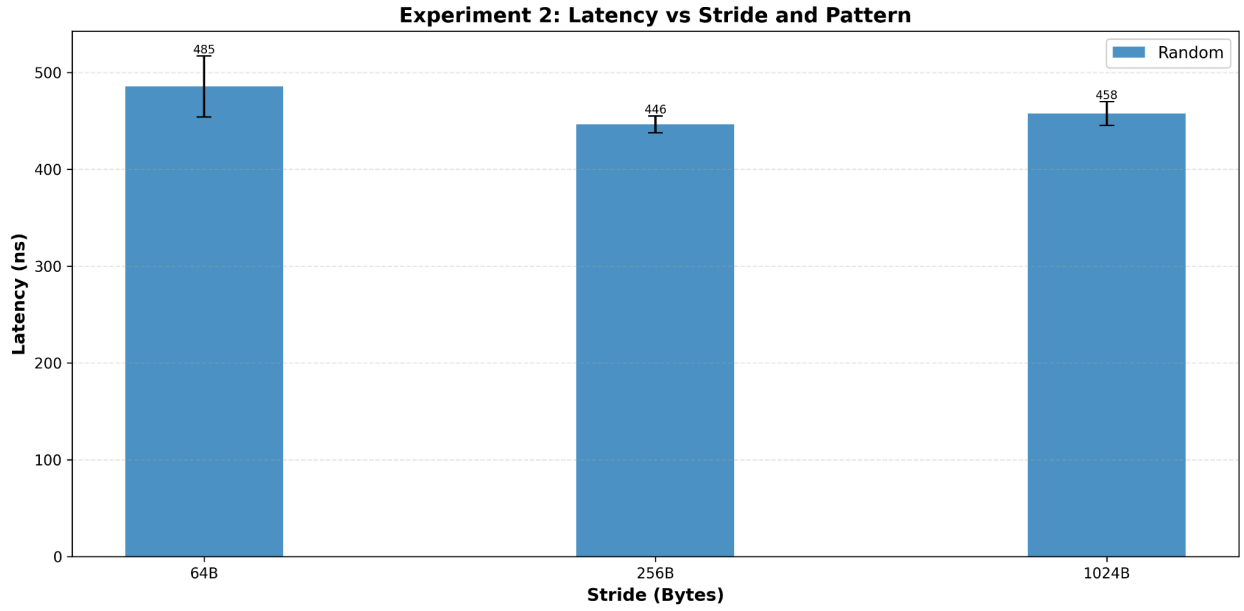



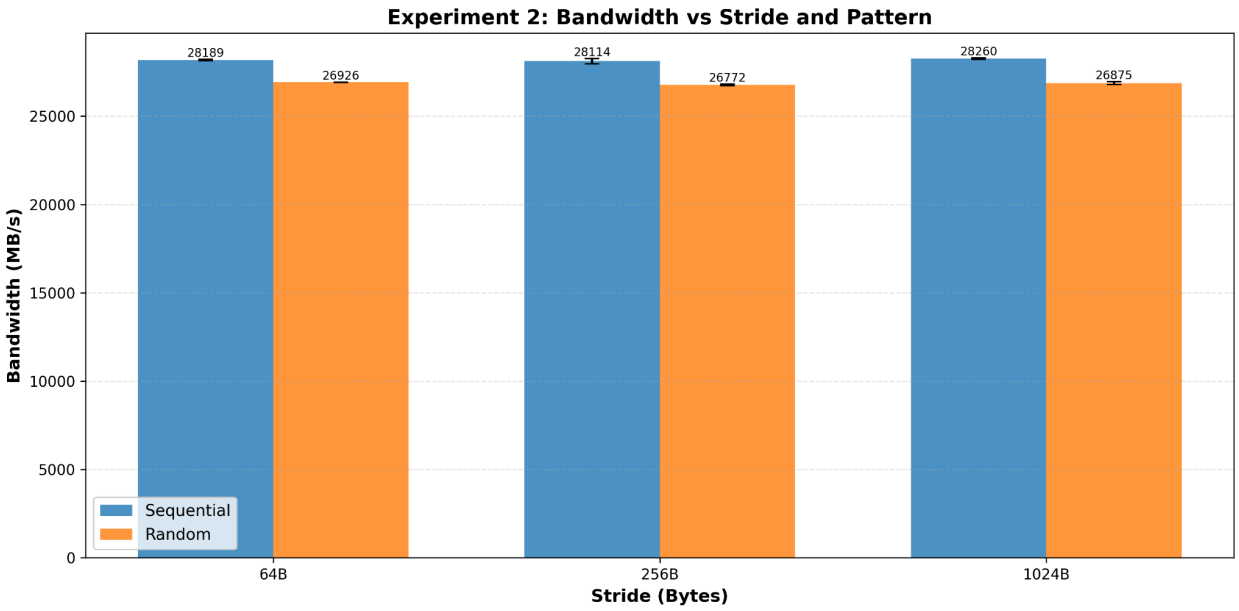
```
================================================================================
EXPERIMENT 1: ZERO-QUEUE LATENCY BASELINES
================================================================================

CPU Frequency: 3301 MHz

Metric                          Mean (ns)    Cycles     Std Dev      CV (%)
--------------------------------------------------------------------------------
L1 Idle Latency                     96.84      319.7        2.53     2.61
Dram Random Latency                 96.76      319.4        0.49     0.51
--------------------------------------------------------------------------------

Note: Cycles calculated assuming 3301 MHz CPU frequency
      1 cycle = 0.303 ns
```

2. Pattern & granularity sweep

Next, the bandwidth and latencies for different pattern sizes were showcased. Data appears to be generally very similar between patterns, with the 64B pattern showing slightly higher bandwidth but also higher latency. This is for both random and sequential accesses. One consistent result is that for the random access, bandwidth was lower than for sequential access, which makes sense as the data needs to be gathered and put together from multiple locations.



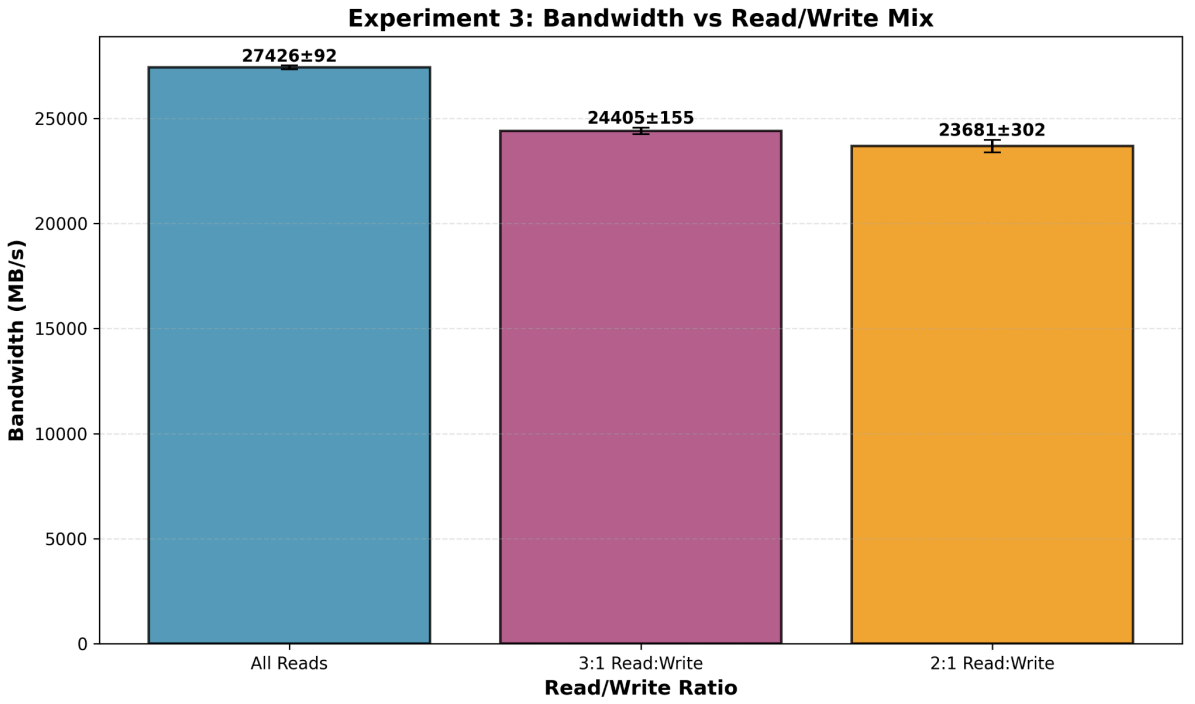



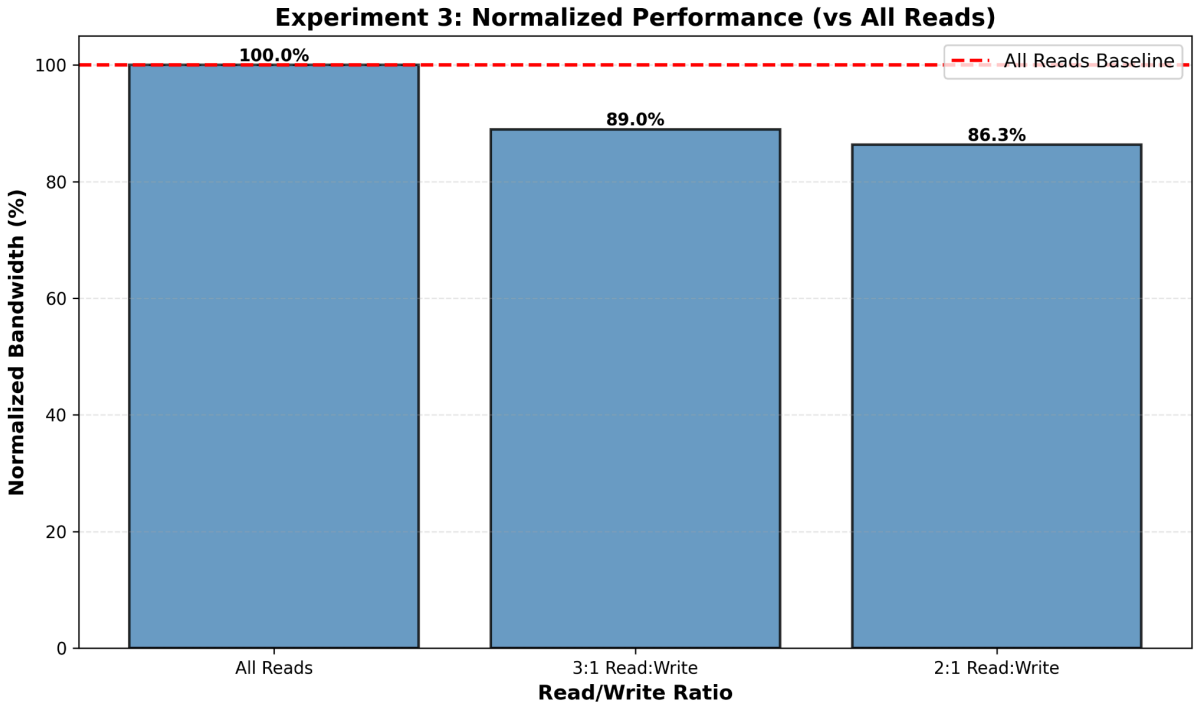



```
================================================================================
EXPERIMENT 2: PATTERN & GRANULARITY SWEEP RESULTS
================================================================================

Pattern      Stride     Bandwidth (MB/s)     Latency (ns)
--------------------------------------------------------------------------------
sequential   64B        28189.0 +- 42.0       N/A
sequential   256B       28114.1 +- 155.2      N/A
sequential   1024B      28260.1 +- 34.1       N/A
random       64B        26925.7 +- 14.0       485.4 +- 31.5
random       256B       26771.8 +- 32.2       446.3 +- 8.6
random       1024B      26875.1 +- 83.3       457.5 +- 12.3
--------------------------------------------------------------------------------
```

3. Read/write mix sweep

Then, the performance of reads versus writes was calculated, and as the ratio of actions went from favoring reads to writes, we see bandwidth decrease. This is showing that reads are quicker than writes, allowing more data to be read than written in the same amount of time. Note that the case for 100% writes was failing for me, so I provided the data I have.

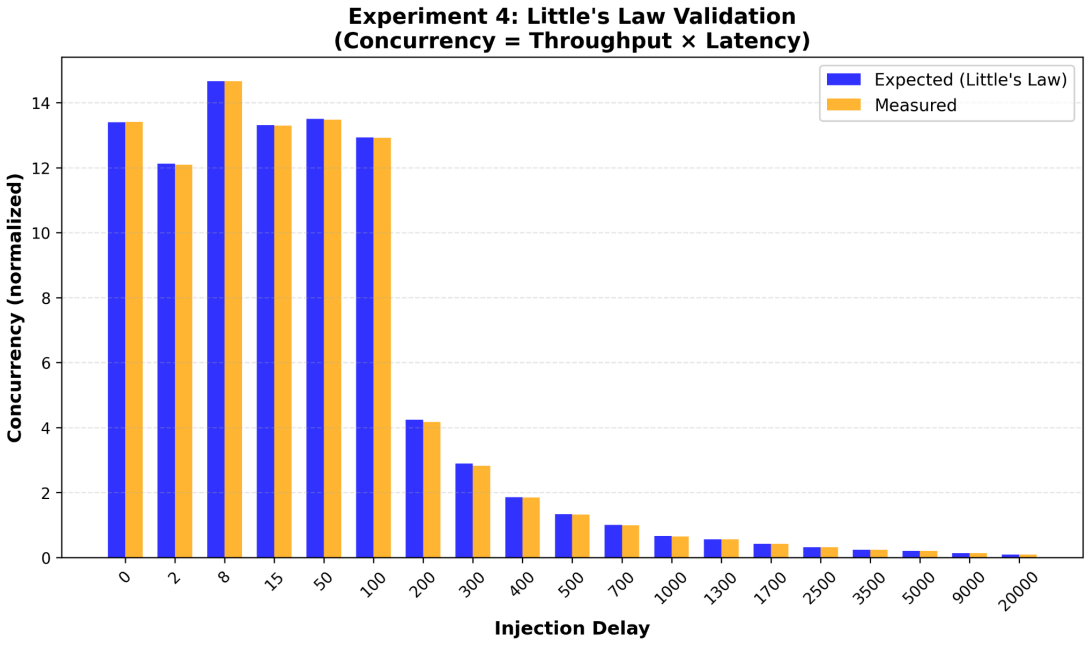


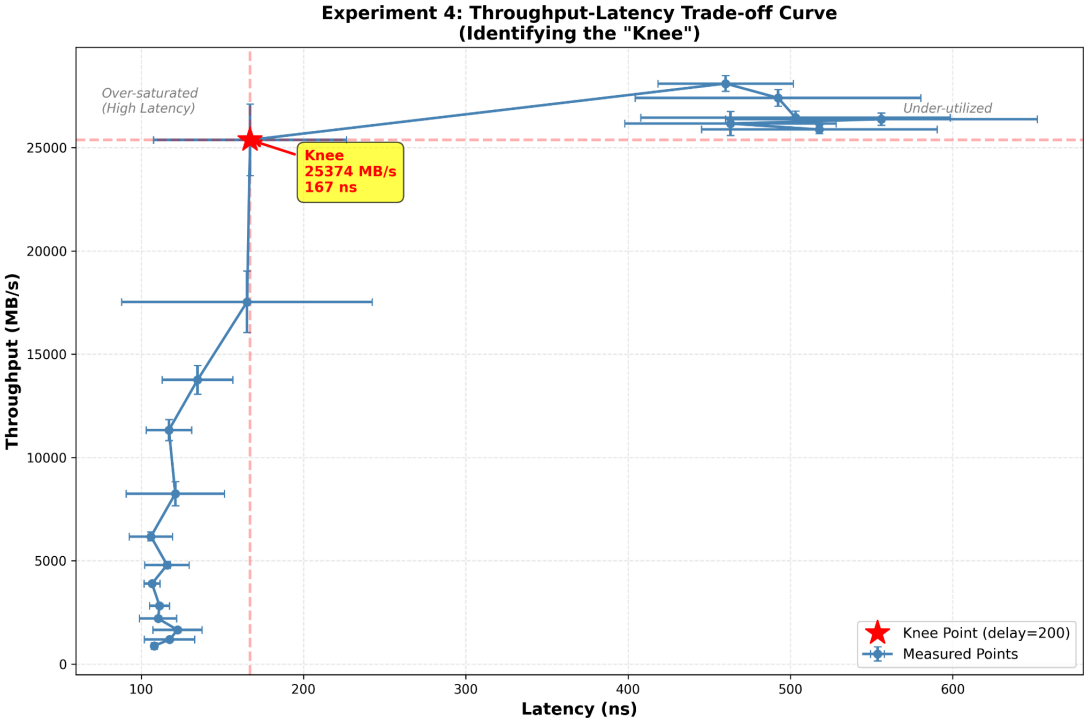



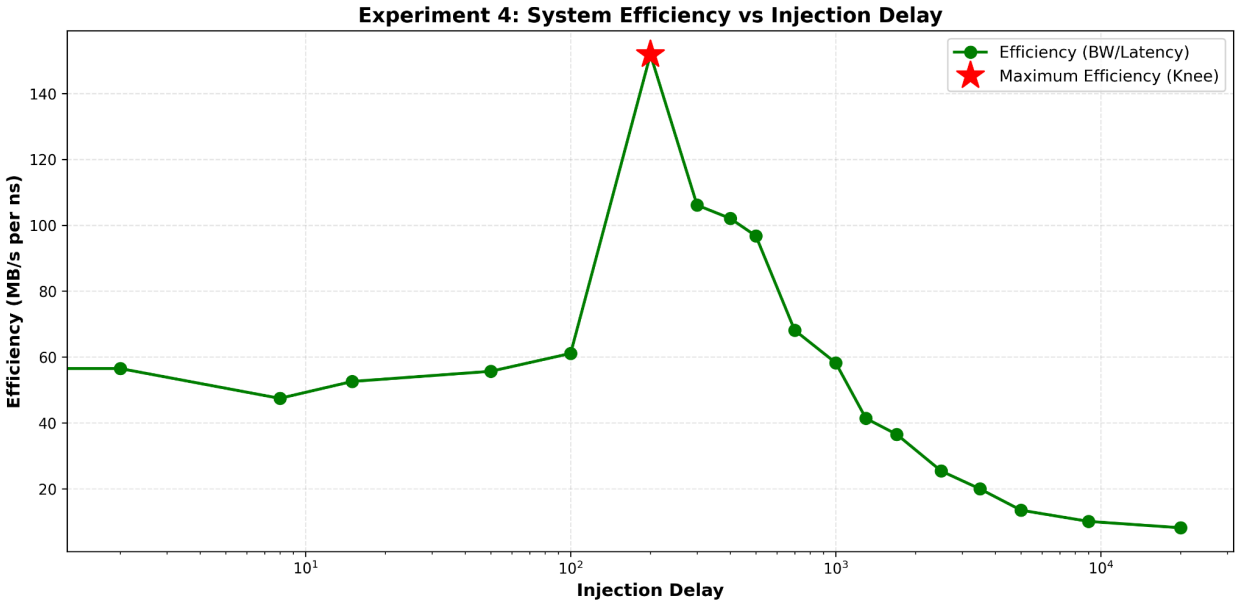
```
================================================================================
EXPERIMENT 3: READ/WRITE MIX SWEEP RESULTS
================================================================================

Ratio           Read%    Write%   Bandwidth (MB/s)     CV (%)
--------------------------------------------------------------------------------
All Reads       100%     0%       27425.8 ? 92.5       0.34
3:1 Read:Write  75%      25%      24404.6 ? 154.7      0.63
2:1 Read:Write  67%      33%      23681.0 ? 302.3      1.28
--------------------------------------------------------------------------------

Baseline (100% Read): 27425.8 MB/s

Performance vs 100% Read:
  All Reads      : 100.0% (+0.0% change)
  3:1 Read:Write :  89.0% (+11.0% change)
  2:1 Read:Write :  86.3% (+13.7% change)
```

4. Intensity sweep

As seen in the below graph, the experimental results very strongly corresponded with the expected results via Little's Law. As the delay increased, the concurrency, or amount of data processed at once, decreased. Combining this with the throughput gave us a knee value of 167ns. This knee is very apparent when looking at both the throughput and efficiency comparisons.



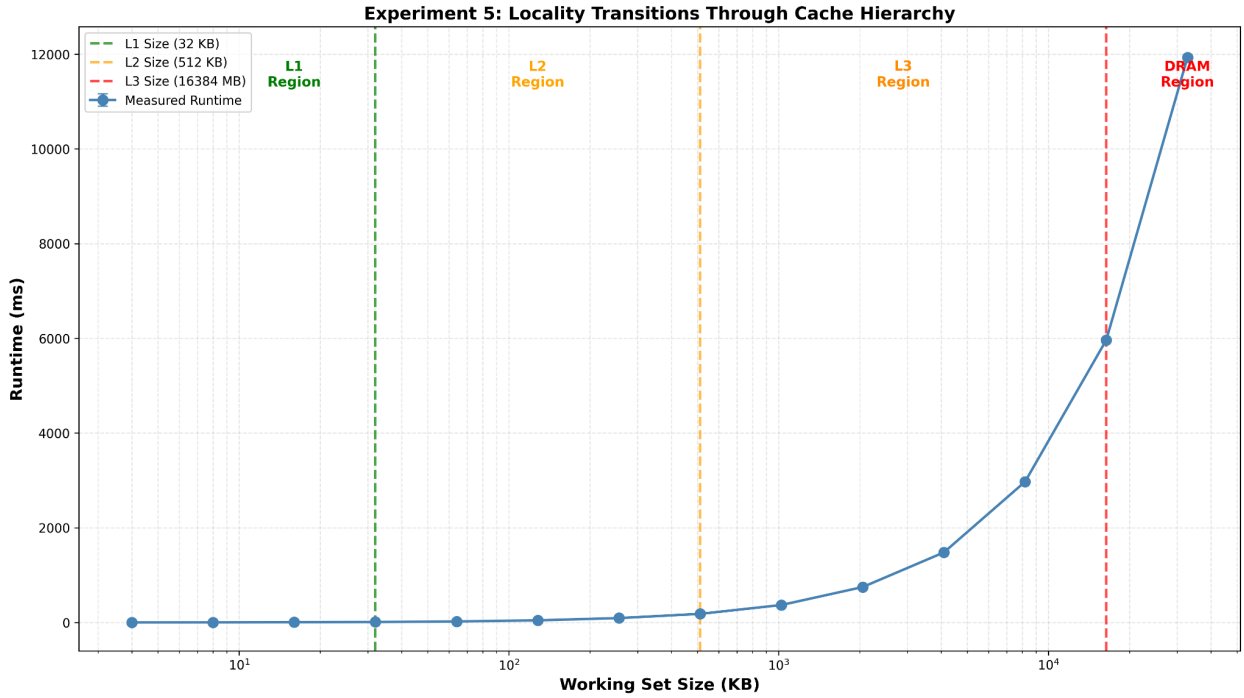






```
============================================================
EXPERIMENT 4: INTENSITY SWEEP - DETAILED ANALYSIS
============================================================

Overall Performance Range:
  Latency: 106.0 ns (min) to 556.1 ns (max)
  Bandwidth: 880.4 MB/s (min) to 28101.4 MB/s (max)

============================================================
KNEE POINT IDENTIFICATION
============================================================
  Injection Delay: 200
  Latency: 167.11 ns
  Bandwidth: 25373.57 MB/s (25.37 GB/s)
  Efficiency: 151.8405 MB/s per ns (maximum)

============================================================
THEORETICAL PEAK COMPARISON
============================================================
  Theoretical Peak: 51200 MB/s (51.2 GB/s)
  Achieved at Knee: 25374 MB/s (49.6% of theoretical)
  Maximum Achieved: 28101 MB/s (54.9% of theoretical)
```

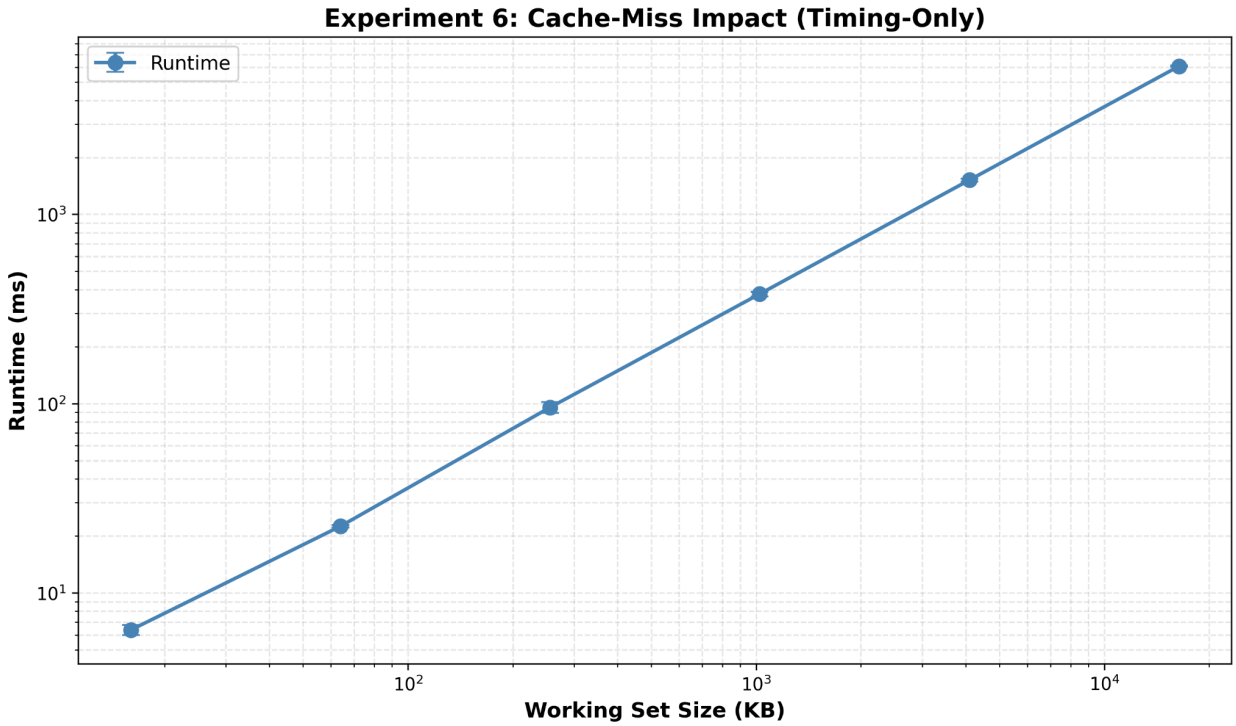
5. Working-set size sweep

Next, the working-set was varied to attempt to transfer the data using different parts of the memory hierarchy. As mentioned previously, this tended to follow the DRAM speeds, and so the speedups expected from solely using cache were not found. As seen in the graph, runtime tracked linearly with working set size, meaning that the data was transferred at the same rate, and only took longer as there was more data to transfer. The experiments would need to be modified in order to take advantage of the cache and see speedups.



6. Cache-miss impact

Similarly, due to the data utilizing DRAM speeds for all cases, no cache-miss impact was seen and runtime scaled with working set size. If caching was used, we would expect sharper transitions in the runtime for data before and after a cache level increase. Instead, data followed a steady bandwidth.



7. TLB-miss impact

Finally, a similar effect is seen in the TLB miss comparison, as the speeds for both page sizes were very close, around 36 ms.