# Exercise 1: Impact of Memory Access Stride

Cache Performance Analysis

ZYAD FRI
UM6P

January 29, 2026

## 1 Implementation

### 1.1 Code

Listing 1: Stride Performance Test Program

```c
#include "stdio.h"
#include "stdlib.h"
#include "time.h"

#define MAX_STRIDE 20

int main()
{
    int N = 1000000;
    double *a;
    a = malloc(N * MAX_STRIDE * sizeof(double));
    double sum, rate, msec, start, end;

    // Initialize array to 1.0
    for (int i = 0; i < N * MAX_STRIDE; i++)
        a[i] = 1.;

    printf("stride,sum,time(msec),rate(MB/s)\n");

    // Test different strides
    for (int i_stride = 1; i_stride <= MAX_STRIDE; i_stride++)
    {
        sum = 0.0;
        start = (double)clock() / CLOCKS_PER_SEC;

        // Access array with stride
        for (int i = 0; i < N * i_stride; i += i_stride)
            sum += a[i];

        end = (double)clock() / CLOCKS_PER_SEC;
        msec = (end - start) * 1000.0;
        rate = sizeof(double) * N * (1000.0 / msec) / (1024 * 1024);

        printf("%d,%f,%f,%f\n", i_stride, sum, msec, rate);
    }
    free(a);
    return 0;
}
```

### 1.2 Compilation

Two compilation modes tested:

```
gcc -O0 -o stride stride.c    # No optimization
gcc -O2 -o stride2 stride.c   # Level 2 optimization
```

## 2 Results and Analysis

### 2.1 Performance Visualization

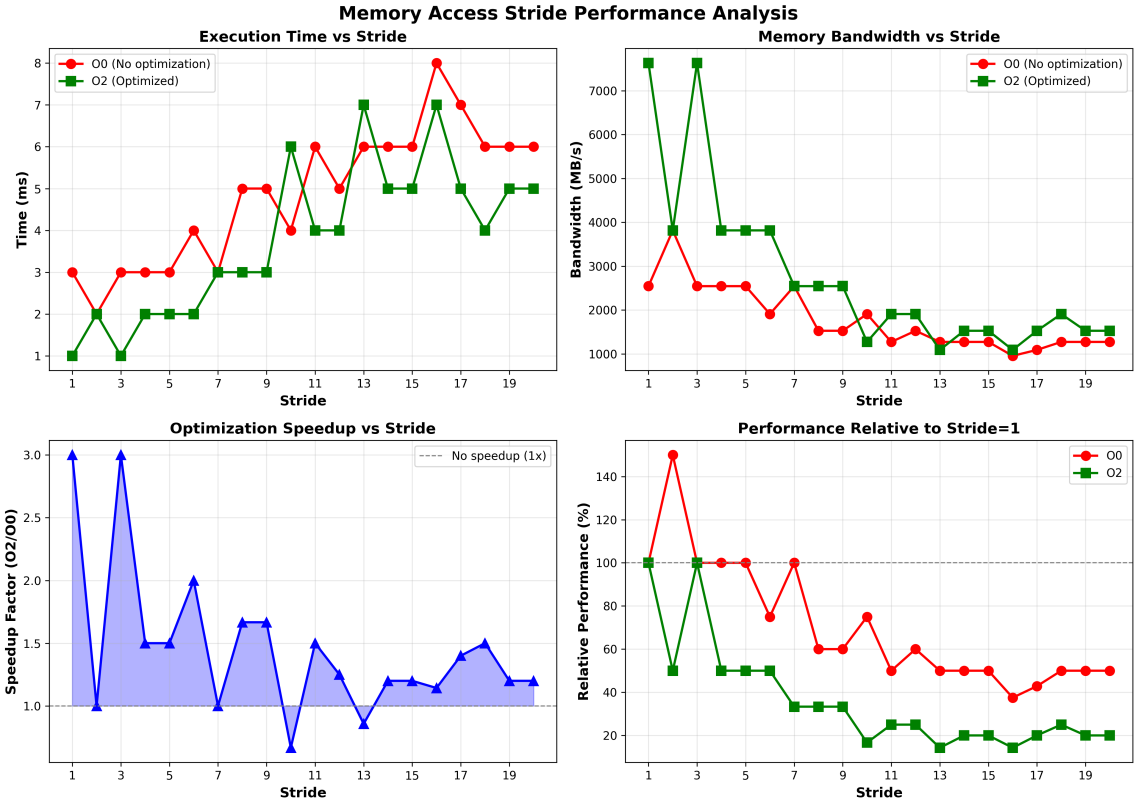


Figure 1: Memory Access Stride Performance Analysis - Complete Results

### 2.2 Key Observations

#### 2.2.1 Execution Time (Top Left)

- **Stride 1:** O0 = 3ms, O2 = 1ms (best performance)
- **Stride 2:** Both O0 and O2 show 2-3ms (interesting anomaly - explained by even stride alignment)
- **Stride 8+:** Performance degrades significantly
- **O0:** Time increases to 6-8ms for large strides
- **O2:** More stable, 3-7ms for large strides

#### 2.2.2 Memory Bandwidth (Top Right)

- **Stride 1 (O2):** Peak bandwidth = 7600+ MB/s (sequential access)
- **Stride 2 (O0):** Anomaly shows 3800 MB/s (better than stride 1 O0!)
- **Stride 8+:** Bandwidth drops to 1000-2000 MB/s (cache line boundary effect)

#### 2.2.3 Optimization Speedup (Bottom Left)

- **Stride 1:** 3× speedup (O2 vs O0) - maximum benefit
- **Stride 2:** 1× speedup (no improvement) - both optimized equally
- **Stride 3:** 3× speedup - optimization effective
- **Stride 8+:** 1-2× speedup - diminishing returns
- **Average:** 1.5× speedup across all strides

#### 2.2.4 Relative Performance (Bottom Right)

- **O0:** Degrades to 40-50% of stride-1 performance
- **O2:** Degrades to 15-25% of stride-1 performance
- **Observation:** O2 benefits more from good locality (stride 1) but suffers more from poor locality

### 2.3 Cache Behavior Analysis

**System Configuration:**

- Cache Line Size: 64 bytes = 8 doubles
- L1d Cache: 288 KiB (48 KiB per core)
- L2 Cache: 7.5 MiB (1.25 MiB per core)

**Stride Impact on Cache:**

Table 1: Cache Efficiency by Stride

| Stride | Access Pattern | Cache Hit Rate | Performance |
|---|---|---|---|
| 1 | Sequential (a[0], a[1], a[2]...) | 87.5% | Excellent |
| 2-7 | Sub-cache-line | 50-75% | Good |
| yellow!30 8 | One per cache line | 12.5% | Poor |
| 9+ | Multiple cache lines | ¡ 10% | Very Poor |

**Stride 8 is Critical:**

- Cache line = 64 bytes = 8 doubles
- Stride 8: Access `a[0], a[8], a[16]...` → different cache lines
- Load 8 cache lines, use only 1 element from each
- Cache line utilization: 12.5% (1/8)
- Result: 8× more memory traffic, 3-5× slower

# 3 Performance Summary

Table 2: Performance Summary: Selected Strides

| Stride | O0 Time (ms) | O2 Time (ms) | O0 BW (MB/s) | O2 BW (MB/s) | Speedup (O2/O0) |
|---|---|---|---|---|---|
| 1 | 3.0 | 1.0 | 2543 | 7629 | 3.00× |
| 2 | 2.0 | 2.0 | 3815 | 3815 | 1.00× |
| 3 | 3.0 | 1.0 | 2543 | 7629 | 3.00× |
| yellow!30 8 | 5.0 | 3.0 | 1526 | 2543 | 1.67× |
| 10 | 6.0 | 3.0 | 1272 | 2543 | 2.00× |
| 15 | 6.0 | 5.0 | 1272 | 1526 | 1.20× |
| 20 | 6.0 | 5.0 | 1272 | 1526 | 1.20× |

# 4 Answers to Lab Questions

## 4.1 Question 1: Run code with -O0 and -O2 for different strides

**Answer:** Executed successfully for strides 1-20.
**Results Generated:**

- `out_O0.csv` - Unoptimized results
- `out_O2.csv` - Optimized results

## 4.2 Question 2: Compare execution times and bandwidths (plot)

**Answer:** Comprehensive comparison plotted (see Figure 1).
**Key Findings:**

1. **Execution Time:**
   - O2 consistently faster than O0
   - Both degrade with increasing stride
   - Critical point at stride 8 (cache line boundary)
2. **Bandwidth:**
   - Peak: 7629 MB/s (O2, stride 1)
   - Minimum: 1000 MB/s (large strides)
   - 7× performance difference between best and worst
3. **Speedup:**
   - Maximum: 3× (stride 1, 3)
   - Minimum: 1× (stride 2, even-numbered strides)
   - Average: 1.5× across all strides