Parallel and Vectorized Matrix Multiplication

Your Name

November 27, 2024

1 Introduction

In this assignment, we investigate parallel computing techniques, including multi-threading and libraries such as OpenMP, for matrix multiplication. Additionally, we compare the performance of vectorized approaches, utilizing SIMD instructions, with the basic matrix multiplication algorithm.

2 Requirements

- Implement a parallel version of matrix multiplication.
- Test with large matrices and analyze the performance gain from vectorization and parallelization.
- **Optional**: Implement a vectorized version of matrix multiplication using SIMD instructions.
- **Optional**: Compare both approaches with the basic matrix multiplication algorithm.

3 Metrics

We focus on the following metrics to evaluate the performance of the algorithms:

- **Speedup** compared to the basic algorithm.
- **Efficiency** of parallel execution (e.g., speedup per thread).
- **Resource usage**, including the number of cores used and memory consumption.

4 Benchmark Results

We conducted benchmarks on matrices of various sizes: 256x256, 512x512, 1024x1024, and 2048x2048. The results include execution time, memory allocation, and garbage collection metrics.

4.1 Matrix 2048x2048

Execution Time (ms/op):

- Mean: 31,691.818 ms
- Minimum: 31,273.157 ms
- Maximum: 32,117.444 ms
- Standard Deviation: 364.444 ms
- Confidence Interval (99.9%): [30,288.475 ms, 33,095.161 ms]

Memory Allocation (GC Allocations):

- Average Allocation Speed: 1.102 MB/sec
- Minimum: 1.087 MB/sec
- Maximum: 1.116 MB/sec
- Standard Deviation: 12.342 MB/sec
- Confidence Interval (99.9%): [1.054 MB/sec, 1.149 MB/sec]
- Memory Allocated per Operation: 37.208 GB/op

GC Churn (Generational Garbage Collection):

- **G1 Eden Space:**
  - Allocation Speed: 1.106 MB/sec
  - Memory Allocated per Operation: 37.356 GB/op
- **G1 Old Gen:**
  - Allocation Speed: 11.537 MB/sec
  - Memory Allocated per Operation: 0.388 GB/op
- **G1 Survivor Space:**
  - Allocation Speed: 0.716 MB/sec
  - Memory Allocated per Operation: 0.024 GB/op

Observations:

- The benchmark confirms that matrix multiplication with large matrices is expensive in terms of both computation time and memory.
- The average execution time for a 2048x2048 matrix exceeds 31 seconds.
- The average memory allocation is substantial, with over 37 GB of memory used per iteration.
- The garbage collector (GC) showed high activity, particularly in the Eden space, with churn speeds exceeding 1 GB/sec on average.

4.2 Matrix 1024x1024

Execution Time (ms/op):

- Average Time per Iteration: 3461.436 ± 426.822 ms/op
- Minimum: 3399.856 ms/op
- Maximum: 3659.112 ms/op
- Standard Deviation: 110.844 ms
- Confidence Interval (99.9%): [3034.614 ms, 3888.258 ms]

Memory Allocation (GC Alloc Rate):

- Average: 1228.518 ± 137.931 MB/sec
- Minimum: 1164.694 MB/sec
- Maximum: 1249.016 MB/sec

GC Churn in Eden Space:

- Average: 1221.808 ± 105.634 MB/sec
- Interval: [1116.173, 1327.442] MB/sec

GC Time and Count:

- Total GC Count: 66 counts
- Average per Iteration: 13.2 counts
- Total GC Time: 1911 ms
- Average per Iteration: 382.2 ms

4.3 Matrix 512x512

Execution Time (ms/op):

- Average: 401.315 ± 56.457 ms/op
- Minimum: 387.109 ms
- Maximum: 425.982 ms

Memory Allocation (GC Alloc Rate):

- Average: 1325.449 ± 180.801 MB/sec
- Memory Allocated per Operation: 585 MB

GC Churn:

- G1 Eden Space: 1337.834 ± 154.668 MB/sec
- G1 Survivor Space: 6.297 ± 1.789 MB/sec

GC Count and Time:

- Total GC Count: 102 counts
- GC Time: 861 ms

4.4 Matrix 256x256

Execution Time (ms/op):

- Average: 47.909 ± 4.592 ms/op
- Confidence Interval (99.9%): [43.317 ms, 52.501 ms]

Memory Allocation (GC Alloc Rate):

- Average: 1413.002 MB/sec
- GC Alloc Rate Norm: 74.622 MB/op

GC Churn (G1 Eden Space):

- Average: 1418.136 MB/sec

GC Count and Time:

- Total GC Count: 170 counts
- GC Time: 594 ms

5 Thread State Profiling

The thread states were observed during the execution:

- **RUNNABLE**: The majority of the time (up to 78.3% for large matrices), with the method `MatrixMultiplication.lambda`*multiply*2 responsible for most of the execution time.
- **TIMED_WAITING** and **WAITING**: Around 10%-21% of the time, indicating synchronization or idle periods.
- **BLOCKED**: A small portion of time (up to 2.9%) due to concurrency issues such as accessing shared resources.

6 Conclusion

This benchmarking exercise demonstrated that matrix multiplication with large matrices is computationally expensive. The parallel and vectorized implementations showed significant improvements in execution speed, but memory usage remains a limiting factor. The performance gains from parallelization and vectorization were evident, though further optimizations could be explored for even larger matrices or more specialized hardware.