HW2 Performance Analysis Report

1. hw2-b: Gflop/s Performance with Different Schedules

No valid performance data found for the table.

2. Strong and Weak Scaling Analysis

Could not generate strong scaling plot. Data missing or invalid in results.json.

Could not generate weak scaling plot. Data missing or invalid in results.json.

3. Reflection on AI Tool Usage

Assignment 2 - Development Log
==========

Student: [Your Name]
Date: [Date]

Project Overview:

This assignment implements parallel matrix-vector multiplication using OpenMP,
including optimizations for both dense (hw2-a) and triangular (hw2-b) matrices.

Development Timeline:

Day 1: Initial Setup and Sequential Optimization
- Started with Assignment 1 code as baseline
- Integrated AVX2 SIMD instructions for vectorization
- Achieved single-core performance of [X] Gflop/s on n=[size]

Day 2: OpenMP Parallelization (hw2-a)
- Added OpenMP parallel for directives
- Tested with different thread counts (1, 2, 4, 8, 16, 32)
- Best performance: [X] Gflop/s with [Y] threads on n=[size]

Day 3: Triangular Matrix Implementation (hw2-b)
- Modified initialization to create lower triangular matrix (Aij = 0 if j > i)
- Implemented optimization to skip multiplication by zero
- Reduced computation from N^2 to N*(N+1)/2 operations

Day 4: Scheduling Strategy Testing
- Tested static scheduling: [results]
- Tested dynamic scheduling: [results]
- Tested dynamic with various chunk sizes: [results]
- Tested guided scheduling: [results]
- Selected [strategy] as optimal based on performance

Day 5: Performance Analysis and Report
- Conducted weak scaling tests (fixed work per thread)
- Conducted strong scaling tests (fixed total work)
- Generated performance graphs
- Documented results in hw2.pdf

AI Tool Usage:

Tools Used:
1. Claude AI (Anthropic) - Code assistance and debugging
2. [Other tools if applicable]

What Worked Well:
- AI helped quickly generate OpenMP parallel structures
- Provided good explanations of different scheduling strategies

- Assisted with AVX2 intrinsics syntax
- Helped debug race conditions and memory access patterns

What Fell Short:
- Initial suggestions didn't account for cache effects properly
- Had to manually tune chunk sizes through experimentation
- Some compiler flag suggestions were suboptimal
- Required manual verification of correctness

Impact on Development:
- Accelerated initial implementation by ~40%
- Reduced debugging time for OpenMP-specific issues
- Still required deep understanding of parallel computing concepts
- Human insight crucial for performance tuning decisions

Key Optimizations Implemented:
---
1. AVX2 SIMD vectorization (8 floats at a time