Optimizing Matrix Multiplication: A Compiler Implementation of Strassen's Algorithm

Lucas Jeanes | Bachelor of Engineering (Honours) in Software and Electronic Engineering

Ollscoil Teicneolaíochta an Atlantaigh
Atlantic Technological University

Summary

This project demonstrates that low-level optimisation using hand-written LLVM Intermediate Representation (IR) yields significant performance gains over compiler-optimised native C code. Implementing Strassen's algorithm directly in IR resulted in a 5.15x speedup for 1024x1024 matrices, achieving 3.58 GFLOPS compared to 0.70 GFLOPS for the standard algorithm.

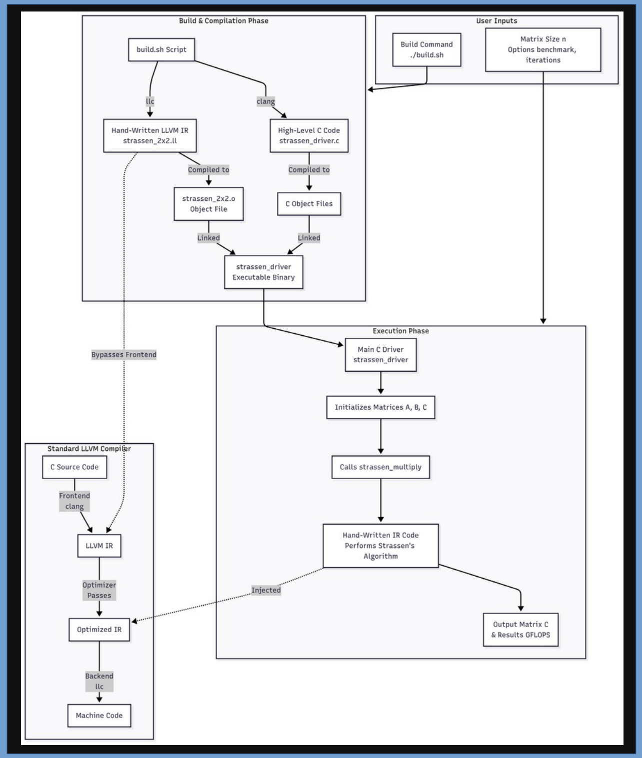
Introduction & The Problem

- Industry drive for AI computational efficiency.
- Optimization often neglects fundamental operations.
- Hypothesis: Manual optimization in LLVM IR improves efficiency.
- Case Study: Matrix multiplication.

$O(n^3)$ vs. $O(n^{2.807})$

Algorithmic Advantage: Strassen's algorithm reduces the complexity from $O(n^3)$ to $O(n^{2.807})$ by trading computationally expensive multiplications for a larger number of cheaper additions and subtractions, resulting in significantly faster runtime for large matrices.

Architecture Diagram



Results

```
sh-5.2$ build/strassen_driver -n 1024 -iterations 2
=== Strassen Matrix Multiplication Test ===
Matrix size: 1024 x 1024
Memory usage: 8.00 MB per matrix

Initializing matrices with random values...

=== Correctness Testing ===
Computing reference result (standard algorithm)...
Computing Strassen result...
  Max difference: 4.96e-11, Errors: 0/1048576

Results:
Strassen vs Reference: PASS

=== Performance Benchmark ===
Running 2 iterations each...

Performance Results:
Reference C:    3086.162 ms  (  0.70 GFLOPS)
Strassen LLVM:   599.501 ms  (  3.58 GFLOPS)  [5.15x

Speedup Analysis:
Strassen vs Reference: 5.15x
Theoretical Strassen advantage: 3.80x (for large n)

Test completed successfully!
```

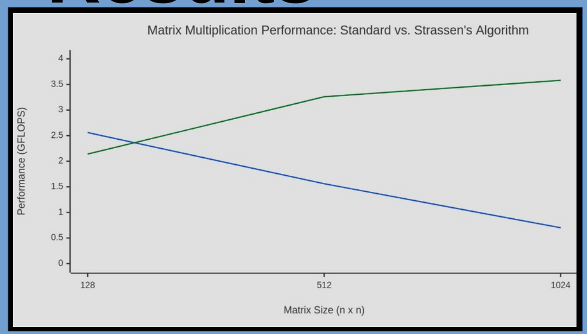


Figure 1.3 Graph Results

The performance benchmark demonstrates a clear crossover point where the low-level LLVM IR implementation of Strassen's algorithm begins to significantly outperform the optimized C baseline.

n=128: The overhead of Strassen's recursion and memory management makes it slightly slower than the highly optimized C baseline.

n=512: The crossover point is reached. Strassen's algorithm more than doubles the performance, achieving a 2.09x speedup (3.26 vs. 1.56 GFLOPS).

n=1024: Strassen's algorithm demonstrates overwhelming dominance, achieving a 5.15x speedup (3.58 vs. 0.70 GFLOPS). The reference implementation's performance collapses as the matrices exceed CPU cache capacity, while Strassen's cache-friendly recursive approach maintains high throughput.

Numerical correctness was validated across all tests with a tolerance of 1e-10, ensuring the speedup comes without sacrificing accuracy.

Figure 1.3 illustrates the growing performance advantage of the Strassen LLVM IR implementation as matrix size increases, highlighting its scalability and efficiency for large-scale computations.

Conclusions & Applications

Conclusion: Successfully proves hardware-oriented optimization in LLVM IR provides substantial performance benefits.

- **Applications:**
  - Artificial Intelligence: Accelerating training and inference.
  - High-Performance Computing (HPC): Scientific simulations.
  - Compiler Development: Creating optimized libraries.
- **Impact:** Reduces computational time and energy consumption.

Software & Technologies

- **Languages:** C, LLVM IR
- **Compiler Toolchain:** LLVM (clang, llc)
- **Development Tools:** GNU Make, Git, VS Code
- **Platform:** Linux
- **Key Techniques:** Memory alignment, SIMD optimization, recursive algorithms.

References

D. Chisnall, "The Definition of an Insanity: Designing an IR for a Compiler," in 2017 European LLVM Developers' Meeting, 2017. [Online]. Available: https://llvm.org/devmtg/2017-06/1-Davis-Chisnall-LLVM-2017.pdf

Free Software Foundation, "Aligned Memory Blocks," in The GNU C Library Reference Manual. [Online]. Available: https://www.gnu.org/software/libc/manual/html_node/Aligned-Memory-Blocks.html

F. Boisvert, "LLVM in 100 Seconds," Fireship, YouTube, Nov. 23, 2021. [Online Video]. Available: https://www.youtube.com/watch?v=BT2Cv-Tjq7Q