SpMV Calculation using GPU(Apple Metal) and CPU(OpenMP)

Zehong Wang

2023-05-01

1 Introduction

This project implements a Sparse Matrix-Vector Multiplication (SpMV) calculator using CSR and COO formats and optimizations on both CPU and GPU. The program loads a sparse matrix and a vector from a file, and calculates the matrix-vector product.

- The GPU part is implemented using **Apple Metal Shading Language**
- The CPU part is implemented using **OpenMP**

2 Hardware Requirement

This program can only run on **Apple Chip Laptop**, such as M1, M1 Pro, M1 Max, M2, M2 Pro, M2 Max.

3 Environment Setup

```
# Install llvm version of clang to support OpenMP
brew install llvm
# Install OpenMP
brew install libomp
# Install GSL
brew install gsl
# Install cmake
brew install cmake
# Install XCode Command Line Tools to support Metal Compiler
xcode-select --install
```

4 Runnning Command

The program **error tips are very friendly**, so you can just run the program and see the tips.

```
cmake .
make && make kernel
./spmv xxx.mtx
```

5 File Structure

5.1 spmv.metal

This is the Metal kernel using Metal Shading Language. It provides three kernel functions to perform Sparse Matrix-Vector Multiplication (SpMV) using different formats and optimizations on GPU.

- **spmv_csr**
  This kernel function calculates the SpMV product using the CSR format. It takes in row pointers, column indices, values, input vector 'x', and an output vector 'result'. The kernel processes each row of the sparse matrix in parallel.

- **spmv_coo**
  Optimization 1 for 'spmv_csr'. This kernel function calculates the SpMV product using the COO format. It processes the non-zero elements of the sparse matrix in parallel. It takes row indices, column indices, values, input vector 'x', and an output vector 'result' (**with atomic float type**).

- **spmv_csr_loop_unrolling**
  Optimization 2 for 'spmv_csr'. This kernel function applies loop unrolling optimization. The loop unrolling is performed with a step of 4 to potentially improve performance.

5.2 spmv_calculator.h / spmv_calculator.cpp

The **SpmvCalculator** is responsible for performing SpMV calculations using different approaches:

- GPU with CSR format
- GPU with CSR format and loop unrolling optimization
- GPU with COO format
- CPU with serial processing

- CPU with OpenMP parallel processing

It also provides a method for verifying the results.

5.3 logger.h / logger.cpp

The **Logger** provides static methods for logging messages of various levels: ‘info‘, ‘error‘, ‘debug‘, and ‘warn‘. It also includes a method, ‘time‘, for measuring and logging the time taken by specific operations.

6 Research and Design

- The Apple GPU with Metal Shading Language (MSL) can run up to **1024** threads. Technically, it can achieve a **1000x** speedup, confirmed by running all the computations inside a single kernel; the code logic was omitted.
- GPU MSL kernel optimization is not as efficient as CPU optimization, such as shading overhead and kernel compiler optimization. Therefore, it experiences performance loss.
- The basic algorithm uses CSR and assigns each row to a thread, which then serially loops through the non-zero values in the row to compute the final results.
- Optimization 1: Employ COO format instead of CSR, distributing all non-zero values to each GPU thread and atomically summing the results. While this algorithm should outperform the CSR format, atomic reduction overhead and memory cache issues prevent significant improvement.
- Optimization 2: Since Metal GPU compiler optimization is less mature than C++, loop unrolling can lead to significant speedup due to SIMD optimization.

7 Performance

7.1 Conclusion

- CPU OpenMP with 4 threads achieves a **4x** speedup compared to the CPU Serial Version (**TEST-1** vs **TEST-2**, **TEST-6** vs **TEST-7**).
- GPU COO runs **slightly faster** than GPU CSR (**TEST-3** vs **TEST-5**, **TEST-8** vs **TEST-10**).
- GPU CSR with loop unrolling optimization achieves a **2x** speedup (**TEST-3** vs **TEST-4**, **TEST-8** vs **TEST-9**).
- CPU Serial and OpenMP versions attain a **6x** speedup with ‘-O3‘ compiler optimization (**TEST-1** vs **TEST-6**, **TEST-2** vs **TEST-7**).

- GPU CSR provides a **25x** speedup compared to the CPU Serial without compiler optimization and a **4x** speedup compared to CPU Serial with '-O3' optimization (**TEST-1** vs **TEST-4**, **TEST-6** vs **TEST-9**).
- Comparing the best versions, the GPU still runs faster than the CPU (**TEST-7** vs **TEST-9**).

7.2 Running Results

- Results Table

CPU Compiler Optimization	Test Number	Test Item	Time (ms)
[-O0]	1	CPU Serial Version	7467
	2	CPU OpenMP Version	1858
	3	GPU CSR Version	625
	4	GPU CSR with Loop Unrolling Version	297
	5	GPU COO Version	606
[-O3]	6	CPU Serial Version	1152
	7	CPU OpenMP Version	398
	8	GPU CSR Version	627
	9	GPU CSR with Loop Unrolling Version	294
	10	GPU COO Version	578

- CPU applies [-O0]

```
➜  MetalSpMV git:(main) ✗ ./spmv data/af_shell10.mtx
[INFO] Loaded Metal Library for SpMV.
[INFO] Loaded Metal Kernel Function: spmv_coo
[INFO] Loaded Metal Kernel Function: spmv_csr_loop_unrolling
[INFO] Loaded Metal Kernel Function: spmv_csr
[INFO] Loading matrix and vector...
[BENCHMARK] Loaded matrix and vector. 8225ms
[INFO] Running SpMV on CPU Serial Version...
[BENCHMARK] Finished SpMV on CPU Serial Version. 7467ms  TEST-1
[INFO] Running SpMV on CPU OpenMP Version...
[BENCHMARK] Finished SpMV on CPU OpenMP Version. 1858ms  TEST-2
[INFO] Running SpMV on GPU CSR Version...
[BENCHMARK] Finished SpMV on GPU CSR Version. 625ms      TEST-3
[INFO] Running SpMV on GPU CSR Version optimized with Loop Unrolling...   TEST-4
[BENCHMARK] Finished SpMV on GPU CSR Version optimized with Loop Unrolling. 297ms
[INFO] Running SpMV on GPU COO Version...
[BENCHMARK] Finished SpMV on GPU COO Version. 606ms TEST-5
[INFO] Running Verification...
[INFO] CPU OpenMP Version verification passed!
[INFO] GPU CSR Version verification passed!
[INFO] GPU CSR Version optimized with Loop Unrolling verification passed!
[INFO] GPU COO Version verification passed!
[INFO] Finished Verification!
```

- C++ applies [-O3]

```
➜  MetalSpMV git:(main) ✗ ./spmv data/af_shell10.mtx
[INFO] Loaded Metal Library for SpMV.
[INFO] Loaded Metal Kernel Function: spmv_coo
[INFO] Loaded Metal Kernel Function: spmv_csr_loop_unrolling
[INFO] Loaded Metal Kernel Function: spmv_csr
[INFO] Loading matrix and vector...
[BENCHMARK] Loaded matrix and vector. 7611ms
[INFO] Running SpMV on CPU Serial Version...
[BENCHMARK] Finished SpMV on CPU Serial Version. 1152ms  TEST-6
[INFO] Running SpMV on CPU OpenMP Version...
[BENCHMARK] Finished SpMV on CPU OpenMP Version. 398ms   TEST-7
[INFO] Running SpMV on GPU CSR Version...
[BENCHMARK] Finished SpMV on GPU CSR Version. 627ms      TEST-8
[INFO] Running SpMV on GPU CSR Version optimized with Loop Unrolling...   TEST-9
[BENCHMARK] Finished SpMV on GPU CSR Version optimized with Loop Unrolling. 294ms
[INFO] Running SpMV on GPU COO Version...
[BENCHMARK] Finished SpMV on GPU COO Version. 578ms      TEST-10
[INFO] Running Verification...
[INFO] CPU OpenMP Version verification passed!
[INFO] GPU CSR Version verification passed!
[INFO] GPU CSR Version optimized with Loop Unrolling verification passed!
[INFO] GPU COO Version verification passed!
[INFO] Finished Verification!
```

8 Available Matrix Collection

- nlpkkt80
- af_shell10
- StocF-1465