Course: CS205-C/C++ Programming Language

Project II

Matrix Multiplication

by

Sreyny Tha-12113053

Instructor
Prof. Shiqi Yu

Department of Computer Science and Engineering
Southern University of Science and Technology

1 Introduction

Matrix Multiplication is simple operation in linear algebra, but it is crucial for deep learning as well as computer science projects. We aim to implement a matrix multiplication in Java and in C programming languages. Then, we analyze the implementation and result of both codes.

2 Implementation

In this project, I use 1 dimensional array to implement matrix multiplication in both c and Java as shown in the picture below. For one dimension array representation, I just shift the array by the number of matrix's column, then do the matrix multiplication from the random generated matrices.

Matrix Multiplication In Java

```java
public class MatrixMultiplication{
1 usage
public static void matrixMultiplication(float [] A, float []B, float[] C, int m, int n, int p){
    for(int i=0;i<m;i++){
        for(int j=0;j<p;j++){
            C[i*p+j]=0;
            for(int k=0;k<p;k++){
                C[i*p+j]+=A[i*n+k]*B[k*p+j];
            }
        }
    }
}
no usages
public static void print_result(float [] result, int m, int p){
    for(int i=0;i<m;i++){
        for(int j=0;j<p;j++){
            System.out.printf("%.2f ",result[i*p+j]);
        }
        System.out.println();
    }
}
no usages
public static void print_matrix(float []matrix, int size){
    for(int i=0;i<size;i++){
            System.out.printf("%.2f ",matrix[i]);
    }
    System.out.println();
}
```

Matrix Multiplication In C

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

void matrix_multiplication(float * A, float * B, float *C,int m, int n, int p){
    for(int i=0;i<m;i++){
        for(int j=0;j<p;j++){
            C[i*p+j]=0;
            for(int k=0;k<n;k++){
                C[i*p+j]+=A[i*n+k]*B[k*p+j];
            }
        }
    }
}
void print_result(float * result, int rows, int columns){
    for(int i=0;i<rows;i++){
        for(int j=0;j<columns;j++){
            printf("%.2f ", result[i*columns+j]);
        }
        printf("\n");
    }
}

void print_matrix(float * vector, int size){
    for(int i=0;i<size;i++){
        printf("%.2f  ", vector[i]);
    }
    printf("\n");
}
```

3 Analysis

3.1 Experiment Data

Now, let's observe the result given by both programs C and Java. Matrix Multiplication is simple operation in linear algebra, but it is crucial for deep learning as well as computer science projects. We aim to implement a matrix multiplication in Java and in C programming languages. Then, we analyze the implementation and result of both codes. Matrix Multiplication take time complexity $O(n^3)$.

Java and C program are compiled on :

- jDK 21 for JAVA
- gcc.exe (MinGW.org GCC-6.3.0-1) 6.3.0
- gcc (Ubuntu 11.4.0-1ubuntu1 22.04) 11.4.0

I use optimization compiler -O3 for C and run on both Window and Linux for C

code. The size of the two matrices are set to same (n x n and n x n) as dimension in the table.

Dimension	Java Program	C Program
100	0.0058736 s	– window: 0.002000 s – Linux: 0.000608 s
500	0.1132345 s	– window: 0.177000 s – Linux: 0.104390 s
1000	0.7119601 s	– window: 1.41400 s – Linux: 0.722775 s
2000	7.6720098 s	– Window: 12.409000 s – Linux: 8.997278 s
3000	63.1702381 s	– window: 47.82300 s – Linux: 47.548550 s
4000	165.8619823 s	– Window: 132.588000 s – Linux: 185.002437 s
5000	360.7012062 s	– Window: 309.347000 s – Linux: 414.857719 s
6000	670.8859163 s	– Window: 568.02900 s s – Linux: 757.887499 s
7000	1165.6270165 s	– Window: 1059.93000 s – Linux: 1300.940603 s

Table 1: Experiment Data

3.2 Experiment Result

Based on the result of experiment above, we can infer that:

- for smaller array size, Java can compile as fast as C program
- as the size of matrices getting bigger, C is faster than java
- for smaller array size, compilation time on Linux is faster than Window
- as the size of matrices getting bigger, Linux is slower than Window
- In C we have deallocate the memory after using it, but in java there is a garbage collector to clear memory after running. -

3.3 Understand Advantages of Java and C Compilers

Java advantages

1. Array Bounds Checking: Java performs array bounds checking, which can lead to safer code but might introduce a slight overhead.

2. Automatic Array Initialization: In Java, arrays are automatically initialized to default values (0 for integers). This can save initialization steps and potentially improve cache locality.

3. JIT Compilation and Runtime Optimizations: Just like in the previous example, the JVM's JIT compiler can optimize the code at runtime. This includes loop unrolling, method inlining, and other optimizations that can improve performance.

4. Garbage Collection Efficiency: Java's garbage collector, when optimized and tuned, can efficiently manage memory and reduce overhead. This can be particularly beneficial for matrix multiplication, where memory allocation and deallocation are frequent.

5. Efficient Library Usage: Java's libraries for mathematical operations are highly optimized. Libraries like Apache Commons Math or optimized functions in the Java standard library can be used for matrix operations.

6. Predictable Memory Access: Java's array handling is predictable, especially when working with 1D arrays. Predictable memory access patterns can lead to better cache utilization.

C advantages

1. Manual Memory Management: In C, we have direct control over memory allocation and deallocation. This can sometimes lead to more efficient memory usage, especially in performance-critical applications.

.**2. Compiler Optimizations:** C compilers offer a wide range of optimizations, including loop unrolling, vectorization, and instruction scheduling. With the right compiler flags (-O3, architecture-specific optimizations), C code can be highly optimized.

3.4 Analysis Result

The reason that Java is slower in compilation for large size of matrix are:

1. Bytecode Generation: Larger matrices lead to more complex bytecode, requiring additional processing time.

2. Multiple Steps: Java compilation involves parsing, analysis, bytecode generation, and optimization. Each step takes more time with larger and more complex matrices.

3. Optimization Overhead: Java compilers optimize code during compilation. Optimizing large matrices requires complex analysis, increasing compilation time.

4. Just-In-Time (JIT) Compilation: Java performs some optimizations during initial compilation. These JIT optimizations add overhead, especially for large matrix calculations.

5. Memory and Processing: Java needs more memory and processing power for large matrices.

While C program:

1. Direct Compilation to Machine Code: C code is directly compiled into machine code without an intermediate step like bytecode. This direct compilation process is faster and more efficient.

2. Simpler Compilation Steps: C compilers have fewer and simpler compilation steps compared to Java. here is less overhead in translating C code directly into machine instructions.

3. Optimization Flags: C compilers offer a range of optimization flags (like -O1, -O2, -O3, etc.). Using -O3 optimization, C aggressively optimizes code without spending excessive time on analysis.

3.5 Linux vs. Window

For Linux costs more than time window when size of Matrix getting big, the reasons can be following:

1. Different version of GCC on Linux and Window
2. Different compiler eventhough both are GCC
3. Different in Memory Management System
4. Different System Libraries for compiler and linker

4 Conclusion

In conclusion, for smaller array sizes, Java demonstrates comparable compilation speeds to C. However, as the size of matrices increases, C exhibits faster compilation speeds for matrix operations, overshadowing Java's performance. These differences are caused by Java's compilation process complexities, C's direct machine code generation.

5 References

The list of reference

1. Options That Control Optimization
https://gcc.gnu.org/onlinedocs/gcc/Optimize-Options.html

2. Java Compilation Process
https://www.geeksforgeeks.org/compilation-execution-java-program/

3. C Compilation Process
https://www.javatpoint.com/compilation-process-in-c

4. Chat GPT: I got Chat GPT assistance to explain me more details for result of my experiements.