# CS 5220
# Project 1 - Matrix Multiplication

Weici Hu(wh343)
Sheroze Sheriffdeen(mss385)
Qinyu Wang(qw78)

September 17, 2015

## 1 Introduction

In this project, we tried several methods to fine-tune square matrix multiplication. Based on the `dgemm_blocked.c`, we tried unrolling index, modifying loop sequence to take advantage of SSE, and experimenting with different optimization flags.

## 2 Optimization

### 2.1 Block Multiplication with Multiple Block Sizes

#### 2.1.1 Approach

Working off of `dgemm_blocked.c`, we tried different block sizes to examine the performance changes.

#### 2.1.2 Results

Figure 1 shows the performance of different approaches with various block sizes. (block sizes are a multiple of 2). The performance gain for varying block sizes is not immense but a block size of 64 performs better than other block sizes.

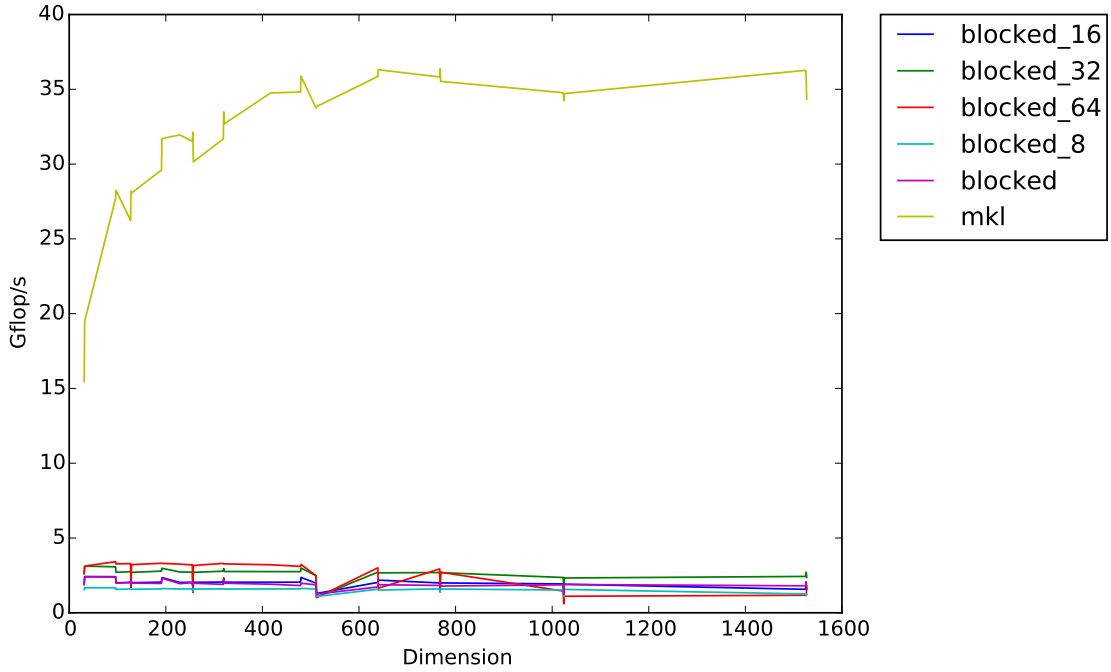


Figure 1: Block size variation

In addition, we attempted block sizes that are not a multiple of 2. Figure 2 is the comparison of the performance against a block size of 64.

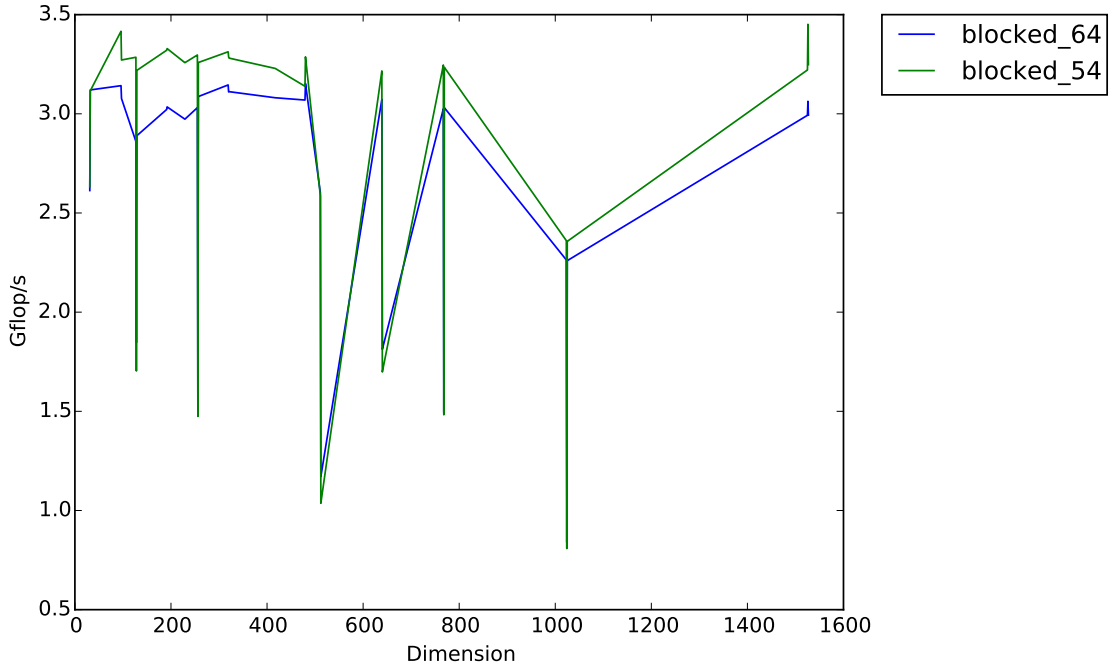


Figure 2: Block size variation

## 2.2 Block Multiplication with Manual Loop Unrolling

### 2.2.1 Approach

In this approach, we manually unrolled 4 computations in the inner most loop of the matrix multiplication of a block.

### 2.2.2 Results

Figure 3 compares the performance of the unrolled blocked version against the vanilla blocked approach. The unrolled versions clearly perform better than the original blocked approach but not by much.

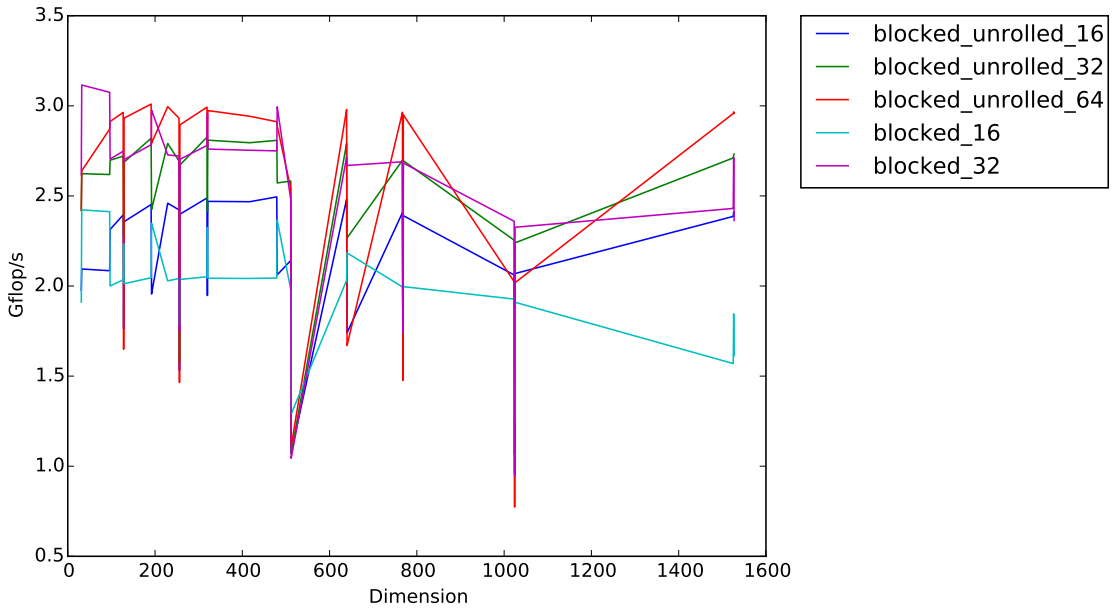


Figure 3: Unrolled blocks vs regular blocks

## 2.3 Compiler Optimization Flags

### 2.3.1 Approach

Using the blocked approach as a baseline, we examine the effect of various compiler flags on the multiplication.

### 2.3.2 Results

Figure 4 shows the performance of blocked multiplication with the flags, `-O3 -march=native -funroll-loops`.

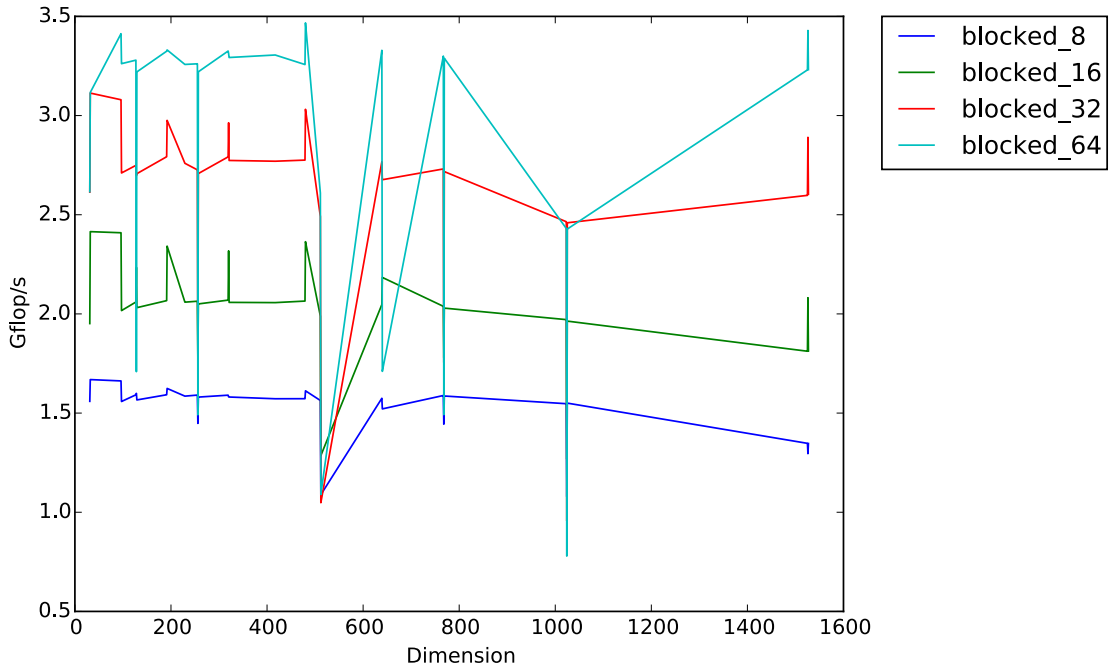


Figure 4: Compiler flags `-O3 -march=native -funroll-loops`

Figure 5 shows the performance of blocked multiplication with the flags, `-O3 -funroll-loops` vs just `-O3`. It appears that merely having the `-O3` flag performs as well as having loops unrolled.

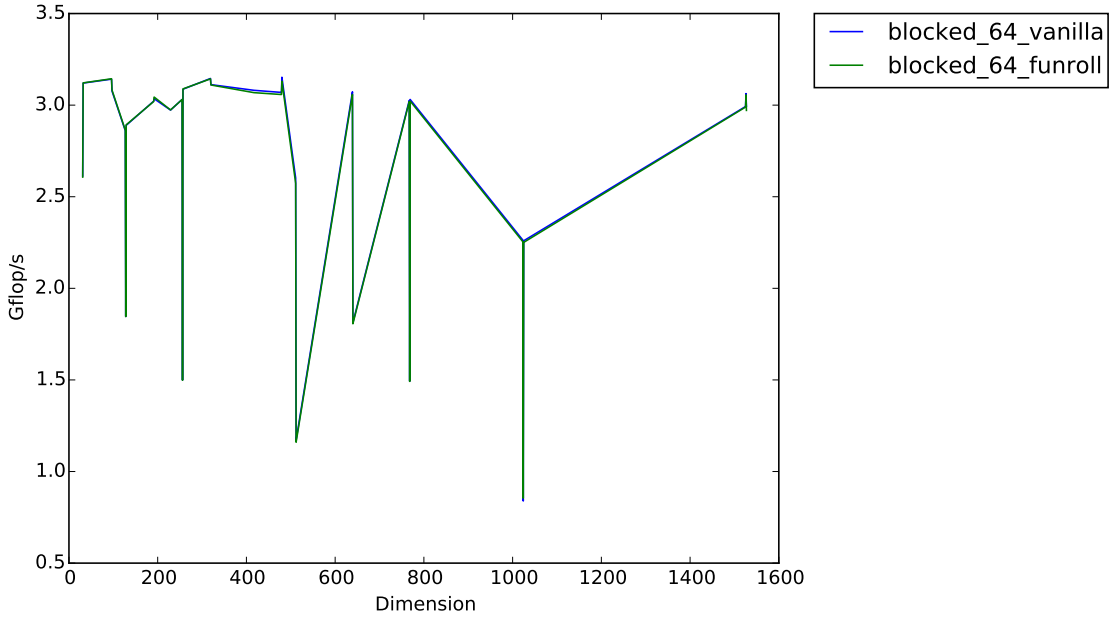


Figure 5: Compiler flags `-O3 -funroll-loops`

### 2.4 Loop reordering

#### 2.4.1 Approach

The current block multiplication in the innermost loop does not have unit stride with $i, j, k$ loop ordering.

#### 2.4.2 Results

### 2.5 AVX Instructions

#### 2.5.1 Approach

//Describe what we did here.

#### 2.5.2 Results

//Add graphs here.

## 3 Next Steps

### 3.1 Copy Optimization