Implementation of Cholesky Factorization using Intel AVX insructions

Oleksandr Zaytsev
Ukrainian Catholic University
Faculty of Applied Sciences
Lviv, Ukraine
oleks@ucu.edu.ua

Abstract

Intel Advanced Vector Extensions (Intel AVX) is a set of instructions for doing Single Instruction Multiple Data (SIMD) operations on Intel architecture CPU[**?**].

In this work I demonstrate how AVX instructions can be used to speed up the execution of widely used numerical algorithms. Cholesky factorization was chosen as an example due to it importance.

Keywords SIMD, AVX, parallel algorithms, numerical methods, Cholesky factorization

1 Introduction

Software packages like R call native functions from LAPACK etc.

1.1 Cholesky Factorization

Every symmetric positive-definite matrix A can be decomposed into the product

$$A = LL^\top$$

Where L is a lower-triandular matrix.

2 Data generation

I used R to generate 6 symmetric positive-definite matrices with real-value entries. The matrices are square and have the following 250, 500, 1000, 2500, 5000, 7500 rows respectively. These matrices are then stored in binary files and loaded by other modules of this project.

I chose the approach of pre-generated matrices as opposed to the idea of generating a new matrix for each experiment, because this way the measurements will be more accurate. I compare the performance of two algorithms on exactly the same input.

3 Implementation

There are two places in the algorithm that can be parallelized:

- Inner product
- Division by diagonal elements

```
double sum;
__m256d ax = _mm256_loadu_pd(&a[0]);
__m256d bx = _mm256_loadu_pd(&b[0]);
__m256d cx = _mm256_mul_pd(ax, bx);
__m256d hsum = _mm256_add_pd(cx,
    _mm256_permute2f128_pd(cx, cx, 0x1));
_mm_store_sd(&sum,
    _mm_hadd_pd(
        _mm256_castpd256_pd128(hsum),
        _mm256_castpd256_pd128(hsum)));
```

4 Experimental results

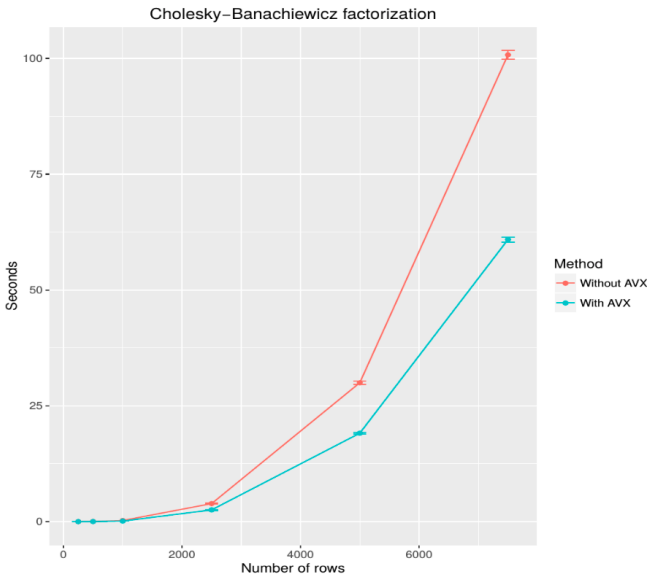

Figure 1. Comparing the performance of two algorithms