Report ACT project 2

Tomas Sunaert, Mauro Gascon, Jacopo Masotti

January 2025

1 Introduction

In the following project we will analyze how different matrix multiplication algorithms scale with respect to the size of the matrix as well as the amount of non zero elements.

2 Size dependent scaling

In this section we will explore how the amount of FLOPs and cpu time will increase for three matrix multiplication algorithms, dense, sparse, and dgemm. In order to do this we created sparse matrices of varying size with the same degree of filling and measure the cpu time and FLOPs for each algorithm. By taking the logarithm of the size and cpu time/FLOPs we can easily extract the scaling using linear region as we know that.

$$log(y) = log(a * x^n) = log(a) + nlog(x) \tag{1}$$

As we care about the asymptotic scaling of our algorithms it is important to sample point in the asymptotic regime since points outside of this can be slowed down by operations that scale better than the asymptotic scaling. Because of this we will fit our linear regression to the last 20-50 points. As is shown in figure 1 we find that even though our sparse multiplication algorithm uses less FLOPs then its dens and dgemm counter parts its also scales as $\mathcal{O}(n^3)$. with respect to the size of the matrix. When it comes to cpu time however, the sparse algorithm falls short scaling at a rate of $\mathcal{O}(n^{3.99})$, whilst dense achieves a scaling of $\mathcal{O}(n^{3.01})$ and dgemm achieves $\mathcal{O}(n^{2.98})$.

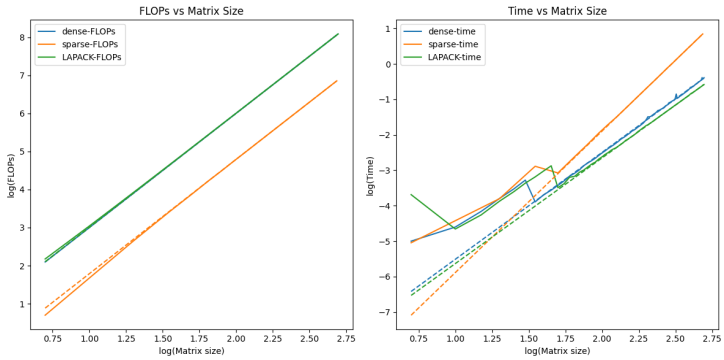

Figure 1: Size dependent scaling of matrix multiplication

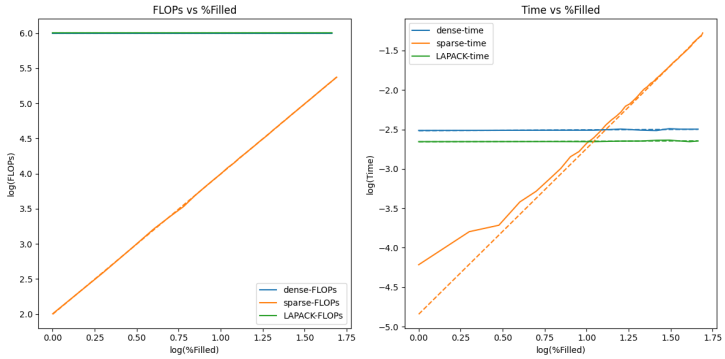

Figure 2: Filling degree dependent scaling of matrix multiplication

3 Filling dependent scaling

When examining the effect of the degree of filling on the amount of FLOPs and cpu time the first thing that becomes apparent is that the dense and the dgemm algorithm appear to be constant. This is expected as these algorithms have no info on which elements are zero. The amount of FLOPs for the sparse algorithm are always smaller then dgemm or dense. Converging converging with the other algorithms when completely filled. The sparse algorithm scales with $\mathcal{O}(n^{1.99})$ with respect to the percentage filled. The same can not be said for the cpu time however, here the sparse algorithm scales as $\mathcal{O}(n^{2.01})$. But crucially surpasses the dgemm algorithm as at a filling rate of around 10 percent for a size matrix of this size.