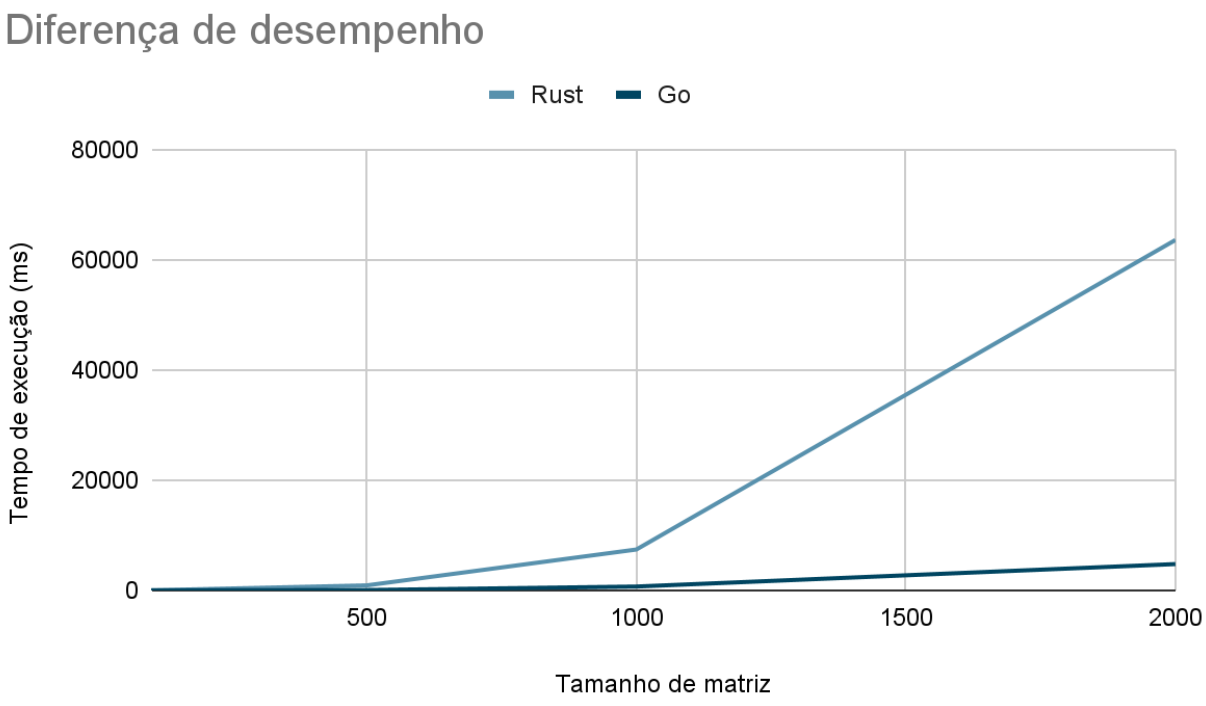
Go possui uma sintaxe muito mais simples e intuitiva que a adotada por Rust. Rust é bem mais restrito em questão de tipagens e também pareceu apresentar uma faixa de tipos mais extensa e minuciosa (um bom exemplo é o tipo usize utilizado em laços), enquanto Go por sua vez permite inferência de tipos.

Pode-se observar também que Rust apresentou menor tolerância quanto ao uso de variáveis globais, sendo estas evitadas por padrão pelo compilador, bem como impedimento da declaração de vetores muito grandes, requerendo uma alocação dinâmica para permitir este tipo de uso da memória. Go além de não apresentar resistência quanto ao uso de globais também não apresentou problemas na declaração de grandes vetores. Outro ponto interessante a se ressaltar é o fato que Rust não possui matrizes nativamente, sendo assim requer ou uma biblioteca externa ou, como foi o nosso caso, uso de um vetor sendo tratado como matriz na execução.

Da maneira que o código foi implementado, ambas as linguagens atingiram um número de linhas semelhantes, sendo Go com 124 linhas enquanto Rust em 114 linhas. Esta pequena diferença porém foi causada em grande parte devido a Go ser mais restrito no tratamento de erros (linhas 45 e 54 por exemplo) e também a maneira com que os packages foram importados (a configuração padrão da IDE utilizada impedia importação na mesma linha), de maneira que caso estes dois não existissem ambas teriam atingido 114 linhas.

Ambos os códigos foram rodados em uma máquina Windows 8.1 64bits, Intel Core i5 1.70GHz, 8GB de RAM



Foi observado que Rust possui grande atenção à segurança da memória, deixando de responsabilidade para o programador criar um código mais específico e evitar erros de execução, com sua tipagem estrita e, por padrão, diversas checagem de código para impedir os mais diversos erros, por mais pequenos que sejam.