In this project, I developed a tool to optimize matrix multiplication with multiple CPU threads.

For example, say we wanted to multiply an $N * M$ matrix with a $M * P$ matrix, resulting in a matrix with dimensions $N * P$. The user will input the number of threads $numberOfThreads$.

I decided to split the work for each thread by giving each thread a unique set of rows of the resulting product matrix to calculate. That way, there are no conflicts between the work each thread does. Thus, thread synchronization via semaphores is not required. To split up the work by rows, I knew I needed to calculate the fromRow and toRow matrix indices for each thread. I also wanted to make it work for numbers not divisible by 2400, such as 7.

Here are the formulas I came up with for figuring out the fromRow and toRow indices:

$fromRow = ((threadNumber - 1) * N)/numberOfThreads$
$toRow = ((threadNumber) * N)/numberOfThreads - 1$

As shown below, this matrix slicing algorithm works even when N is not divisible by $numberOfThreads$.

Case for $N = 2400$, $numberOfThreads = 7$ (2400 is not divisible by 7):
$Thread\,1: fromRow = 0,\ toRow = 341$
$Thread\,2: fromRow = 342,\ toRow = 684$
$Thread\,3: fromRow = 685,\ toRow = 1027$
$Thread\,4: fromRow = 1028,\ toRow = 1370$
$Thread\,5: fromRow = 1371,\ toRow = 1713$
$Thread\,6: fromRow = 1714,\ toRow = 2056$
$Thread\,7: fromRow = 2057,\ toRow = 2399$

Example calculations for threads 6 and 7:
$Thread\,6: fromRow = (5 * 2400)/7 = 1714.29 \Rightarrow 1714 \quad toRow = ((6 * 2400)/7) - 1 = 2056.14 \Rightarrow 2056$
$Thread\,7: fromRow = (6 * 2400)/7 = 2057.14 \Rightarrow 2057 \quad toRow = ((7 * 2400)/7) - 1 = 2399 \Rightarrow 2399$
I stored both fromRow and toRow into integers, so they rounded down.

Once the fromRow and toRow indices are computed for each thread, the threads can then use the basic matrix multiplication process to “fill in” their respective portions of the resulting product matrix.