Comparison of Matrix Multiplication

Ondrej Nečas
Monday 18:00
OI
necasond@fel.cvut.cz

I. Methodology

A. Warm-up

- Every benchmark is run in a single fork on the JHM framework for an arbitrary number of iterations (50 was chosen due to time constraints). Values are then plotted on a graph and the warm-up iteration count is determined by educated guess.

B. Measurements

- After the warm-up estimation, the benchmarks are run for a sufficient number of iterations (50 was chosen due to time constraints) and executions (10 was selected due to time constraints). The execution time of different forms of matrix multiplication is measured.

C. Comparison

For a single benchmark

- Compute the mean (average) of the performance measurements
- Calculate the variance within each execution of the method (execution variance)
- Calculate the variance of the mean values across executions (execution means)
- Compute sample variance (dividing by N-1) instead of population variance (dividing by N), providing an unbiased estimate
- Calculates the confidence interval for the mean performance of each method using the t-distribution
- Determine the degrees of freedom for the t-distribution, which is the number of executions minus 1
- Calculate the quantile value for the desired confidence level
- Computes the half-width of the confidence interval (h) by multiplying the quantile value with the square root of the variance of the means divided by the number of executions (sqrt(S2 / execution_count)).
- The confidence interval is expressed as the mean plus or minus the half-width: mean - h to mean + h
- This gives an overall good estimate of the 95% confidence interval of the average mean for a given benchmark

Cross benchmark comparison

- Generate all possible pairwise combinations of the methods
- For each pair of methods, calculate the ratio of means between the two methods. It divides the mean of the second method by the mean of the first method. This ratio represents the relative performance or efficiency between the methods
- Calculates the confidence interval for the ratio of means using the mean and variance values obtained in the previous step
- Confidence interval calculation for the ratio:

Speed-Up Ratios with Confidence Interval

$$\frac{\overline{Y}\cdot\overline{Y'} \mp \sqrt{\left(\overline{Y}\cdot\overline{Y'}\right)^2 - \left(\overline{Y}^2 - h^2\right)\left(\overline{Y'}^2 - h'^2\right)}}{\overline{Y}^2 - h^2}$$

$$h = \sqrt{t^2_{\frac{\alpha}{2},\nu}\frac{S_n^2}{r_n}} \qquad h' = \sqrt{t^2_{\frac{\alpha}{2},\nu}\frac{S_n'^2}{r_n}}$$

- $\overline{Y}, \overline{Y'}$ - means of the compared implementations
- h, h' - half-widths of the confidence intervals for the single implementations (you can reuse the values from prev. slide)
- S_n^2, S'^2_n - biased variance estimator of the n-th level

- The final comparison is comprised of the average mean ratio of compared methods unioned with lower and upper bounds of the ratio's confidence interval

II. *Machine specification*

- Machine specifications:

OS Name	Microsoft Windows 11 Pro
Version	10.0.22621 Build 22621
Other OS Description	Not Available
OS Manufacturer	Microsoft Corporation
System Name	BANDAR-SERI-BEG
System Manufacturer	HP
System Model	HP Pavilion Gaming Laptop 15-dk0xxx
System Type	x64-based PC
System SKU	7GS59EA#BCM
Processor	Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz, 2592 Mhz, 6 Core(s), 12 Logical P...
BIOS Version/Date	Insyde F.55, 3/28/2022
SMBIOS Version	3.0
Embedded Controller Version	42.47
BIOS Mode	UEFI
BaseBoard Manufacturer	HP
BaseBoard Product	85FB
BaseBoard Version	42.47
Platform Role	Mobile
Secure Boot State	Off
PCR7 Configuration	Elevation Required to View
Windows Directory	C:\Windows
System Directory	C:\Windows\system32
Boot Device	\Device\HarddiskVolume2
Locale	United States
Hardware Abstraction Layer	Version = "10.0.22621.1413"
User Name	BANDAR-SERI-BEG\necasond
Time Zone	Central Europe Daylight Time
Installed Physical Memory (RAM)	16.0 GB
Total Physical Memory	15.8 GB
Available Physical Memory	5.18 GB
Total Virtual Memory	18.7 GB
Available Virtual Memory	4.09 GB
Page File Space	2.88 GB

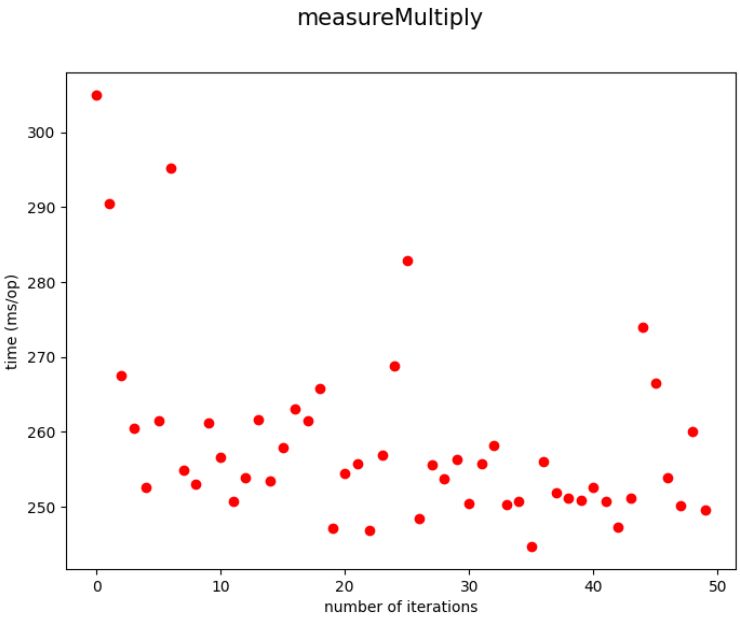
- JVM parameters:

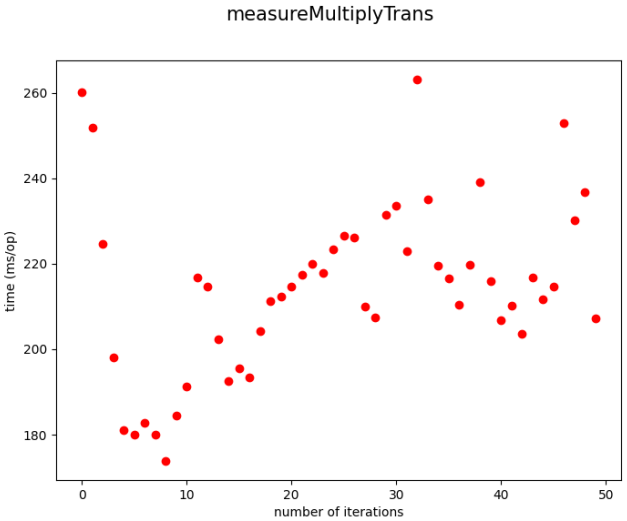
```
# JMH version: 1.34
# VM version: JDK 19.0.1, OpenJDK 64-Bit Server VM, 19.0.1+10-21
# VM invoker: C:\Program Files\Java\jdk-19.0.1\bin\java.exe
# VM options: <none>
# Blackhole mode: compiler (auto-detected, use -Djmh.blackhole.autoDetect=false to disable)
# Warmup: 10 iterations, 1 ms each
# Measurement: 50 iterations, 1 ms each
# Timeout: 10 min per iteration
# Threads: 1 thread, will synchronize iterations
# Benchmark mode: Average time, time/op
```

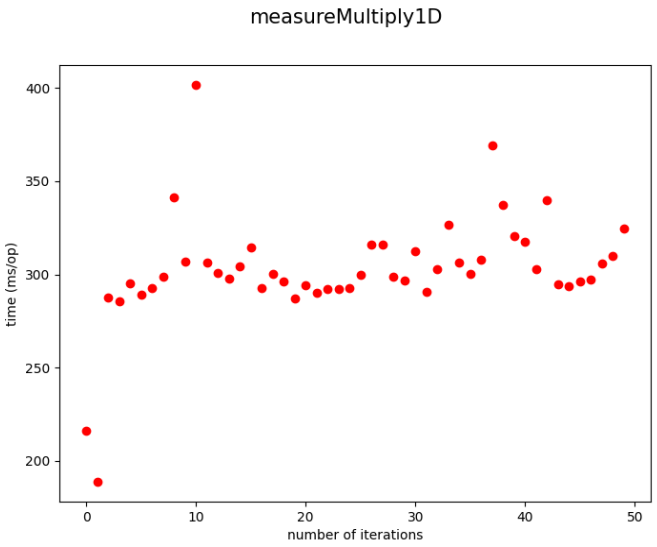
III. *RESULTS*

A. Warm-up

Initial runs are plotted:



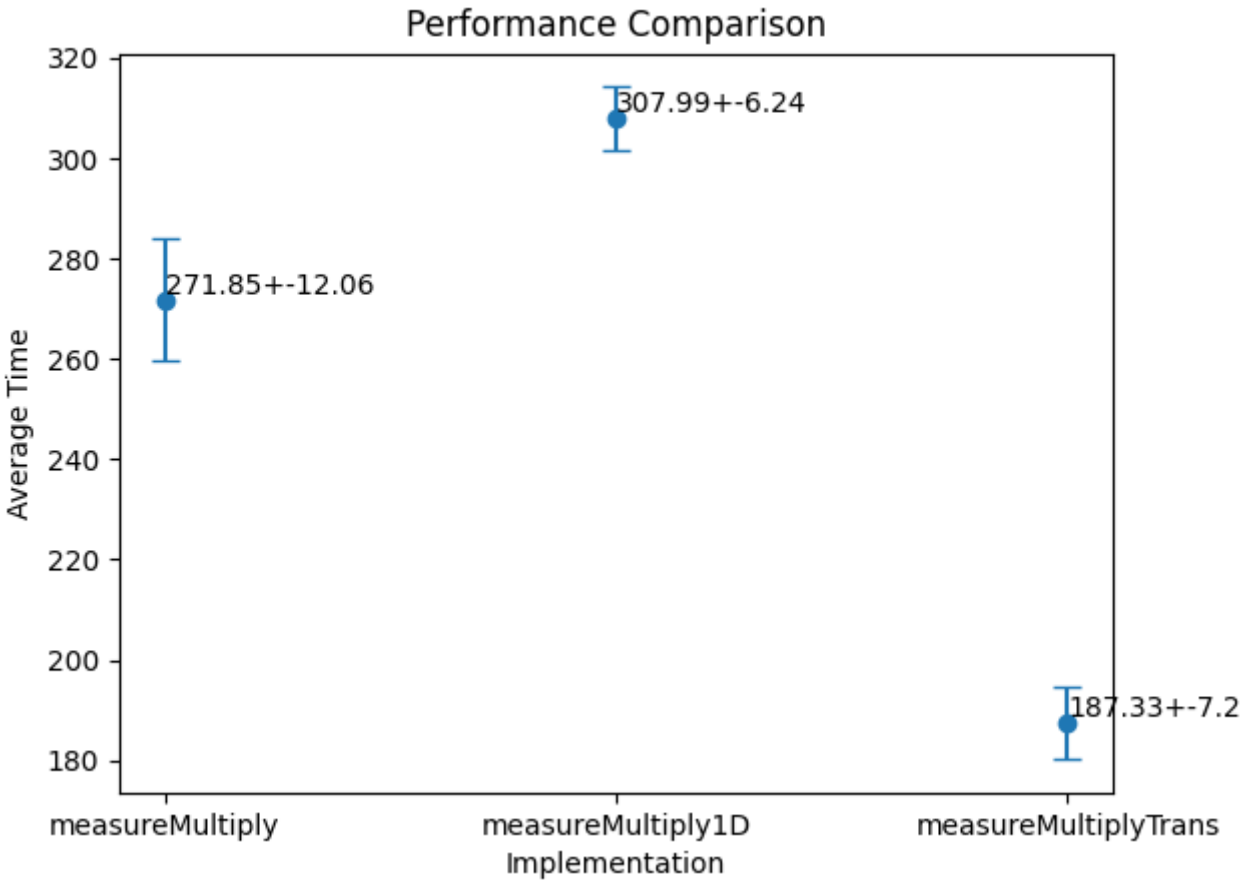




As can be observed from the plots, roughly the first 10 iterations contain the most outliers. Therefore the warm-up number of iterations was chosen to be 10.

B. Measurements visualization

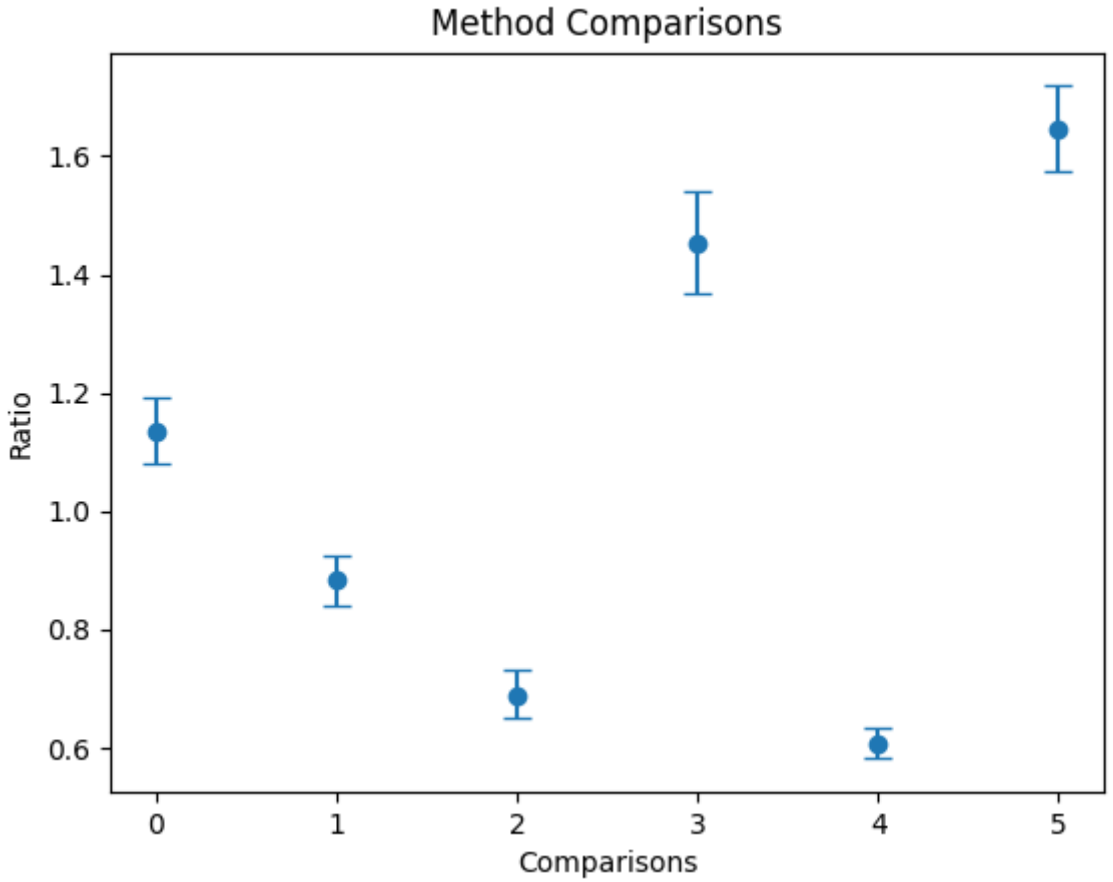
	Mean (`ms/op`)	*95% CI range +-*
Multiply	*271.85*	*12.06*
Multiply1D	*307.99*	*6.24*
MultiplyTrans	*187.33*	*7.2*



C. Comparison

- Results of the comparison of the implementations. From smallest ratio to biggest.

Methods	MultiplyTrans / Multiply1D	MultiplyTrans / Multiply	Multiply / Multiply1D	Multiply1D / Multiply	Multiply / MultiplyTrans	Multiply1D / MultiplyTrans
Ratio	0.608	0.689	0.883	1.133	1.451	1.644
Figure label	4	2	1	0	3	5



IV. Conclusion

- This benchmark test clearly shows that the most effective method for matrix multiplication is a transposition of the second matrix and multiplication by rows. This is due to the fact that row values are stored in a memory close to each other in page tables and easily cached. Therefore the access time to retrieve them is drastically lowered for the computation. The measurements show that this method is roughly 45% faster than the general method for multiplication and 64% faster than the 1D array matrix multiplication.