Vol.7 No.1 Tahun 2026
Jurnal Kecerdasan Buatan, Komputasi dan Teknologi Informasi
https://ejournal.unuja.ac.id/index.php/core
E-ISSN : 2774-7875 and P-ISSN : 2775-0124

Performance Evaluation of CPU and GPU Architectures in Google Colab-Based Parallel Computing Using Execution Time and Speedup

Didik Haryanto[1], Muhamad Nur Hoiri[2], Ahmad Sidiq[3], Amelia Azzahrah[4], Amarudin[5]
[1,2,3,4,5] Universitas Teknokrat, Lampung, Indonesia

Article Info

Article history:

Received : 05 15, 2026
Revised : 05 29, 2026
Accepted : 06 15, 2026

Keywords:

Central Processing Unit;
Graphics Processing Unit;
Google Colab;
parallel computing;

Abstract

Modern computing demands have driven the use of processing devices capable of handli(Fauziah et al., 2024)ng large workloads quickly and efficiently. The CPU plays an important role as the central controller for general-purpose instructions, whereas the GPU provides parallel-processing capabilities that are better suited to large-scale numerical operations. This study aims to evaluate performance differences between CPU and GPU architectures in Google Colab-based parallel computing. A quantitative experimental method was employed using a matrix multiplication test scenario implemented with the PyTorch library. Tests were performed on five matrix sizes: 500 x 500, 1000 x 1000, 2000 x 2000, 3000 x 3000, and 4000 x 4000. The measured parameters included CPU execution time, GPU execution time, and speedup. The results show that the GPU was not optimal for the small 500 x 500 matrix, with a speedup of 0.47 times. However, for larger matrices, the GPU delivered substantial performance gains, achieving speedups of 23.25 to 27.30 times over the CPU. These findings indicate that GPUs are more effective for large-scale parallel processing.

Corresponding Author:

Didik Haryanto,
didik_haryanto@teknokrat.ac.id

INTRODUCTION

The development of modern computing technology requires computer systems to process large volumes of data rapidly, efficiently, and reliably. This demand continues to grow alongside the use of numerical computing, image processing, artificial intelligence, machine learning, data analytics, and scientific simulation, all of which require substantial processing resources(Memeti et al., 2021). In the context of computer systems and architecture, processing capability is determined not only by the software used but also by hardware characteristics, particularly the Central Processing Unit (CPU) and Graphics Processing Unit (GPU). The CPU serves as the primary control center for a wide range of general-purpose instructions, whereas the GPU is designed to execute many operations in parallel, making it more suitable for computational workloads that can be divided into numerous small processes and executed simultaneously(Adrianta et al., 2025). In general, a CPU architecture is designed to execute instructions flexibly and efficiently across a wide variety of computing tasks. A CPU typically has fewer processing cores, but each core provides strong instruction-control capabilities. Therefore, the CPU is essential for running the operating system, controlling program flow, handling logical processes, and managing communication among devices. However, for large-scale numerical workloads such as matrix multiplication, image processing, and artificial intelligence model training, a CPU may encounter limitations because the computations require an extremely large number of repeated

mathematical operations(Owens et al., 2020).

Unlike the CPU, the GPU has an architecture that is more oriented toward parallel processing. A GPU consists of many small processing units that can work simultaneously to complete large numbers of numerical operations. This characteristic has led to the widespread use of GPUs in scientific computing, computer graphics, simulation, deep learning, and large-scale data processing(Harris & Harris, 2021). NVIDIA describes CUDA as a parallel computing platform and programming model that enables computations to be executed on a GPU to improve processing performance. These characteristics make the GPU an important component in discussions of modern computer architecture, particularly parallel processing. Matrix multiplication is one computational operation that can be used to evaluate performance differences between CPUs and GPUs. It is a fundamental operation widely used in numerical computing, machine learning, artificial neural networks, and image processing. Matrix multiplication is suitable as a testing scenario because its computational load increases with matrix size(Torres et al., 2024). For small matrices, the performance difference between a CPU and a GPU may not yet be significant because of the overhead associated with GPU execution. For large matrices, however, a GPU has the potential to perform better because it can distribute the calculations across many processing units in parallel. CPU and GPU performance testing can now be conducted more easily using cloud-based computing services, including Google Colab. Google Colab is a cloud-based Jupyter Notebook service that can be used without special installation and provides access to computing resources, including GPUs and TPUs. These facilities make Google Colab a relevant learning and experimental medium for understanding differences between CPU and GPU architectures without requiring direct ownership of GPU hardware. This is particularly important in educational settings because students can conduct practical computer architecture experiments at low cost and with easy access(Che et al., 2020).

In this study, testing was conducted using the PyTorch library because it supports tensor operations and computation on both CPUs and GPUs. PyTorch provides the torch.matmul function for matrix multiplication and supports CUDA through the torch.cuda module. The PyTorch documentation explains that CUDA tensors can be allocated to the active GPU, allowing computation to be performed on the GPU device. With this support, PyTorch can be used to compare CPU and GPU execution times under the same computational scenario. Although GPUs are known to excel in parallel processing, the performance difference between CPUs and GPUs is not identical for every workload size. For small workloads, a CPU may be faster because it does not require complex data-transfer and parallel-computing setup processes. Conversely, for large workloads, a GPU tends to be superior because a large number of operations can be executed simultaneously(Nickolls et al., 2021). Therefore, empirical evaluation is required to determine the extent to which a GPU improves performance over a CPU in the Google Colab environment. Based on this background, the present study focuses on evaluating CPU and GPU architecture performance in Google Colab-based parallel computing. The research questions are as follows: how do CPU and GPU execution times differ in matrix multiplication; what speedup does the GPU achieve over the CPU at various matrix sizes; and how do the characteristics of CPU and GPU architectures affect the test results? Accordingly, this study aims to compare CPU and GPU performance in parallel computing, measure GPU speedup relative to the CPU, and provide empirical evidence regarding the role of GPUs in supporting modern computing requirements(Ansari & Ansari, 2025).

This study is expected to contribute learning material in the field of computer systems and architecture, particularly regarding differences in CPU and GPU characteristics. The results also provide a simple example showing that computer architecture concepts are not merely theoretical but can be tested directly through computational experiments. Thus, students and beginning researchers can understand that selecting an appropriate processing device has a substantial effect on computer-system efficiency and performance, especially for

DOI: https://doi.org/10.33650/coreai.v6i2

parallel workloads.

METHOD

This study employed a quantitative experimental method. This method was selected because the study aimed to measure and compare the performance of two processing devices, namely the CPU and GPU, based on the execution time required to complete parallel computing operations. Testing was conducted by running matrix multiplication operations in Google Colab as a cloud-based computing environment(Karimi et al., 2022).

Research Type

This study is a computational experiment using a quantitative approach. The experiment was conducted to identify differences in execution time between the CPU and GPU when processing matrix multiplication. The primary data consisted of execution-time measurements in seconds. These results were then analyzed to determine the level of acceleration achieved by the GPU relative to the CPU.

Test Environment

Testing was conducted using Google Colab. Google Colab was selected because it provides a notebook-based computing environment in which Python programs can be executed without local installation(Hakim et al., 2024). It also provides CPU and GPU runtime options, making it suitable for evaluating performance differences between CPU and GPU architectures. The software and libraries used in this study were as follows:

Table 1. Test Environment

No	Component	Description
1	Google Colab	Cloud-based experimental environment
2	Python	Programming language used for testing
3	PyTorch	Library for tensor and matrix operations
4	NumPy	Supporting library for numerical data processing
5	Pandas	Library for presenting test results
6	Matplotlib	Library for visualizing test-result graphs

Test Object

The variables in this study consisted of independent and dependent variables. The independent variable was the type of processing device used, namely the CPU or GPU. The dependent variable was the execution time required to complete matrix multiplication. Matrix size was also used as a variation in computational workload to observe changes in CPU and GPU performance at different data sizes.

Table 2. Matrix Multiplication Test Scenarios

No	Matrix Size
1	500 x 500
2	1000 x 1000
3	2000 x 2000
4	3000 x 3000
5	4000 x 4000

Each matrix size was tested using two processing devices, namely the CPU and GPU. Consequently, each test scenario produced two execution-time values that were subsequently compared.

Research Variables

The variables in this study consisted of independent and dependent variables(Bororing & Gunawan, 2024). The independent variable was the type of processing device used, namely the CPU or GPU. The dependent variable was the execution time required to complete matrix multiplication. Matrix size was also used as a variation in computational workload to observe changes in CPU and GPU performance at different data sizes.

Table 3. Research Variables

No	Variable Type	Variable	Description
1	Independent variable	Processing device type	CPU and GPU
2	Dependent variable	Execution time	Processing time in seconds
3	Control variable	Operation type	Matrix multiplication
4	Control variable	Library	PyTorch
5	Treatment variable	Matrix size	500 x 500 to 4000 x 4000

Test Procedure

The test procedure consisted of several stages. First, a Google Colab notebook was prepared by enabling the GPU runtime through Runtime > Change runtime type > Hardware accelerator > GPU. The device was then checked using the nvidia-smi command and the torch.cuda.is_available() function to ensure that the GPU was available.

Second, the system generated two random matrices of equal size using PyTorch. The matrices were then multiplied using the torch.matmul function. CPU and GPU tests were conducted separately. For CPU testing, the matrices were placed on the CPU device. For GPU testing, the matrices were placed on the CUDA device so that the computation was performed by the GPU. Third, execution time was recorded using Python timing functions. For GPU testing, synchronization was performed using torch.cuda.synchronize() before and after matrix multiplication. This synchronization was necessary to ensure that the recorded execution time represented the actual GPU processing time rather than only the time required to issue the command. Fourth, each matrix size was tested three times. The execution time used in the analysis was the average of these repetitions. Repeated testing was performed to reduce potential bias caused by fluctuations in the Google Colab runtime.

Data Collection Technique

Data were collected from the execution results of Python programs in Google Colab. The recorded data included matrix size, CPU execution time, GPU execution time, and GPU speedup relative to the CPU. Each test result was stored in tabular form using Pandas and then exported to CSV format for analysis and preparation of the research results.

Data Analysis Technique

The data were analyzed using quantitative descriptive analysis. CPU and GPU execution times were compared to determine which device completed matrix multiplication more quickly. GPU speedup was then calculated using the following formula:

Speedup = CPU Time / GPU Time

The speedup value indicates the extent to which GPU performance improves over CPU performance. A speedup greater than 1 indicates that the GPU is faster than the CPU. Conversely, a speedup below 1 indicates that the CPU is faster than the GPU. The analysis results were then presented in tables and graphs to make the performance differences between the CPU and GPU more evident.

RESULTS AND ANALYSIS

Testing was conducted to identify performance differences between the CPU and GPU in completing matrix multiplication operations in Google Colab. Five matrix sizes were used: 500 x 500, 1000 x 1000, 2000 x 2000, 3000 x 3000, and 4000 x 4000. Each matrix size was tested on both the CPU and GPU, and the resulting execution times were compared to obtain the acceleration or speedup value. The test results are presented in Table 4.

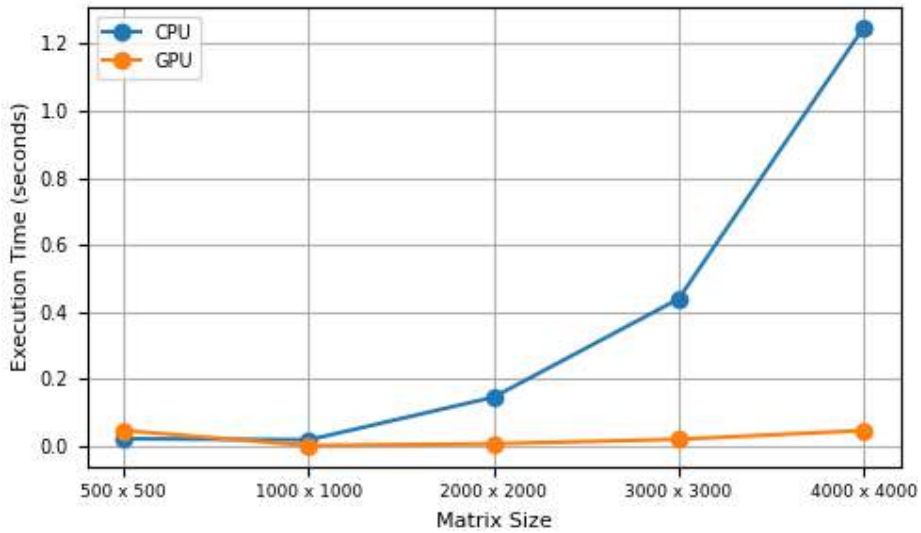
DOI: https://doi.org/10.33650/coreai.v6i2

Table 4. CPU and GPU Execution Time Test Results

No	Matrix Size	CPU Time (seconds)	GPU Time (seconds)	GPU Speedup
1	500 x 500	0.021168	0.045391	0.47 times
2	1000 x 1000	0.017665	0.000760	23.25 times
3	2000 x 2000	0.145168	0.006114	23.74 times
4	3000 x 3000	0.438398	0.019396	22.60 times
5	4000 x 4000	1.247236	0.045683	27.30 times

Table 4 shows clear performance differences between the CPU and GPU. For the 500 x 500 matrix, the CPU execution time was 0.021168 seconds, whereas the GPU required 0.045391 seconds. The GPU speedup at this size was only 0.47 times. Thus, for a small matrix, the GPU did not yet provide better performance than the CPU. This condition may occur because GPU computation requires preparation stages such as allocating data to the GPU and synchronizing the process. When the dataset is small, this preparation time may exceed the benefits provided by parallel GPU processing. A different result was observed for larger matrices. For the 1000 x 1000 matrix, the GPU completed the operation in 0.000760 seconds, whereas the CPU required 0.017665 seconds, resulting in a speedup of 23.25 times. This result indicates that as the data size increases, the GPU can make better use of its parallel-processing capabilities. A large computational workload can be divided among many small processing units, allowing the calculations to run simultaneously. For the 2000 x 2000 matrix, the CPU required 0.145168 seconds, whereas the GPU required only 0.006114 seconds, producing a speedup of 23.74 times. This result shows that the GPU remained superior as the matrix size increased. For the 3000 x 3000 matrix, the CPU required 0.438398 seconds, whereas the GPU required 0.019396 seconds, yielding a speedup of 22.60 times. Although this speedup was slightly lower than that obtained for the 2000 x 2000 matrix, the GPU still performed substantially better than the CPU.

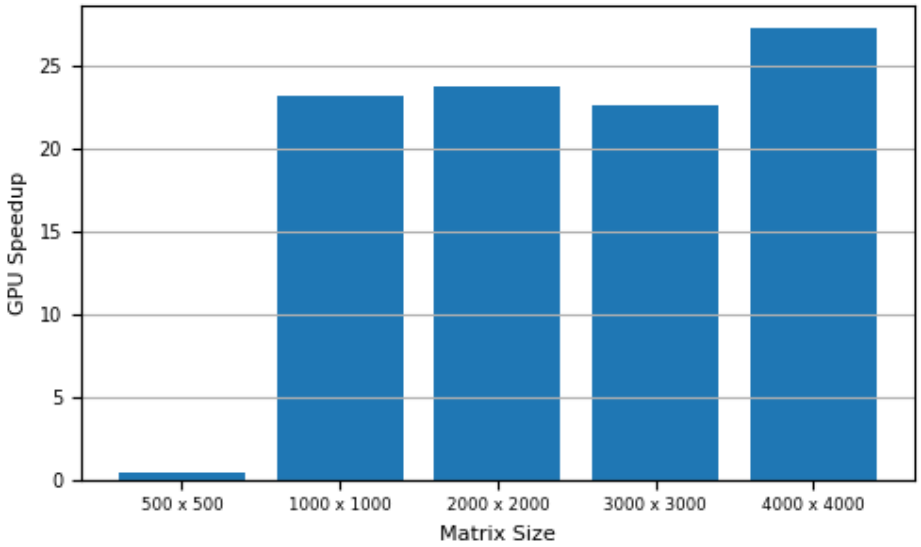
The highest result was obtained for the 4000 x 4000 matrix. At this size, the CPU required 1.247236 seconds, whereas the GPU required only 0.045683 seconds. The GPU speedup reached 27.30 times. This result demonstrates that the advantage of the GPU in parallel computing becomes increasingly apparent as matrix size increases. The GPU can



execute many mathematical operations simultaneously, resulting in a much shorter execution time than that of the CPU.

Figure 1. Comparison of CPU and GPU Execution Times

The execution-time comparison graph shows that CPU time increased sharply as the matrix size grew. For the 500 x 500 and 1000 x 1000 matrices, CPU execution time remained relatively low. However, from 2000 x 2000 to 4000 x 4000, CPU execution time increased substantially. This indicates that the CPU requires more time as the number of mathematical operations increases. Although the CPU provides strong instruction-control capabilities, its relatively smaller number of processing cores makes it less optimal for large-scale parallel workloads. In contrast, GPU execution time remained much lower for large matrices. The



GPU line in the graph stayed near the bottom and did not increase as sharply as the CPU line. This pattern indicates that the GPU is more stable in handling increases in matrix size. This advantage arises from the GPU architecture, which contains many processing units designed to execute parallel operations. Therefore, the GPU is more suitable for repetitive numerical operations such as matrix multiplication.

Figure 2. GPU Speedup over the CPU

The speedup graph shows that the GPU was not superior for the 500 x 500 matrix. A speedup of 0.47 times indicates that the CPU was faster for the small dataset. However, when the matrix size increased to 1000 x 1000, the GPU speedup rose substantially to 23.25 times. For the 2000 x 2000 and 3000 x 3000 matrices, the speedup remained above 22 times. For the 4000 x 4000 matrix, the GPU achieved the highest speedup, namely 27.30 times. These results indicate that the GPU is more effective for large workloads with parallel characteristics. Matrix multiplication is an operation that can be divided into many small calculations. Each element of the result matrix can be computed through relatively independent multiplication and addition operations. For this reason, matrix multiplication is highly suitable for GPU execution. The GPU can distribute the work across many processing units, thereby reducing execution time.

The findings also show that the processing device should be selected according to the type of workload. The CPU remains important for executing general-purpose instructions, running the operating system, handling logical processes, and managing tasks that require high flexibility. However, for large-scale numerical processes that can be executed in parallel, the GPU offers greater advantages. Thus, CPUs and GPUs should not be viewed as devices that completely replace one another; rather, they perform different functions and have distinct characteristics within modern computer systems.

The test results also exhibited minor fluctuations. For example, the CPU execution time for the 1000 x 1000 matrix was lower than that for the 500 x 500 matrix. This condition may have been influenced by several factors, including PyTorch internal optimization, Google Colab runtime conditions, server resource allocation, and caching during testing. Because Google Colab is a cloud-based computing environment, the resources allocated to a session

DOI: https://doi.org/10.33650/coreai.v6i2

may be dynamic. Consequently, execution-time results may vary slightly when tests are conducted at different times or under different accounts.Overall, the results demonstrate that the GPU outperformed the CPU in parallel computing, particularly for large matrices. For small matrices, the GPU was not necessarily faster because of processing overhead. As the computational load increased, however, the GPU provided substantial acceleration. These results support the conclusion that GPU architecture is better suited to parallel computing, whereas the CPU remains essential as the primary control center of a computer system.

CONCLUSIONS

Based on the test results, the GPU architecture performed better than the CPU architecture in parallel computing, particularly for large-scale matrix multiplication workloads. For the small 500 x 500 matrix, the GPU did not show an advantage because its speedup was only 0.47 times, meaning that the CPU was still faster. However, for matrix sizes from 1000 x 1000 to 4000 x 4000, the GPU achieved substantial performance improvements, with speedups ranging from 23.25 to 27.30 times over the CPU. These findings indicate that the GPU is more effective for numerical operations that can be processed in parallel, whereas the CPU remains important as the primary control center of a computer system. Therefore, the processing device should be selected according to workload characteristics so that the computer system can operate more efficiently. Future studies should test larger matrices, different GPU models, and other computational scenarios such as machine-learning model training, image processing, or numerical simulation. Additional parameters, including memory usage, power consumption, and runtime stability, could also be measured to provide a more comprehensive evaluation of CPU and GPU performance. Because the present tests were conducted in Google Colab, where resource allocation is dynamic, future research should also compare Google Colab results with those obtained from local hardware or other cloud-computing services.

REFERENCES

W. Stallings, Computer Organization and Architecture: Designing for Performance, 11th ed. Hoboken, NJ, USA: Pearson, 2022.

S. L. Harris and D. M. Harris, Digital Design and Computer Architecture, RISC-V Edition, 1st ed. Cambridge, MA, USA: Morgan Kaufmann, 2021. https://doi.org/10.1109/WCAE53984.2021.9707615

W.-m. W. Hwu, D. B. Kirk, and I. El Hajj, Programming Massively Parallel Processors: A Hands-on Approach, 4th ed. Cambridge, MA, USA: Morgan Kaufmann, 2022.

J. D. Owens, M. Houston, D. Luebke, S. Green, J. E. Stone, and J. C. Phillips, "GPU computing," Proceedings of the IEEE, vol. 96, no. 5, pp. 879-899, 2008, doi: 10.1109/JPROC.2008.917757.https://doi.org/10.1109/JPROC.2008.917757

J. Nickolls, I. Buck, M. Garland, and K. Skadron, "Scalable parallel programming with CUDA," ACM Queue, vol. 6, no. 2, pp. 40-53, 2008, doi: 10.1145/1365490.1365500. https://doi.org/10.1145/1365490.1365500

S. Che, M. Boyer, J. Meng, D. Tarjan, J. W. Sheaffer, S.-H. Lee, and K. Skadron, "A performance study of general-purpose applications on graphics processors using CUDA," Journal of Parallel and Distributed Computing, vol. 68, no. 10, pp. 1370-1380, 2008, https://doi.org/10.1016/j.jpdc.2008.05.014

S. Memeti, L. Li, S. Pllana, J. Kolodziej, and C. Kessler, "Benchmarking OpenCL, OpenACC, OpenMP, and CUDA: Programming productivity, performance, and

energy consumption," arXiv preprint arXiv:1704.05316, 2017. https://doi.org/10.1145/3110355.3110356

Adrianta, A. T., Ariessanti, H. D., Setiawati, P., & Ichwani, A. (2025). Rancang Bangun Sistem Pemesanan Berbasis Website Dengan QR Code Menggunakan Metode Waterfall Di Cafe Eau De Coffee. *INSERT : Information System and Emerging Technology Journal*, *6*(1), 50–62. https://doi.org/10.23887/INSERT.V6I1.90898

Ansari, M. Q., & Ansari, M. Q. (2025). *Accelerating Matrix Multiplication: A Performance Comparison Between Multi-Core CPU and GPU*. http://arxiv.org/abs/2507.19723

Bororing, G. M. G., & Gunawan, F. (2024). Sistem Pemesanan dan Pembayaran Makanan berbasis Web Terintegrasi dengan Application Programming Interface (API). *Jurnal Informatika Dan Bisnis*, *13*(1), 37–48. https://doi.org/10.46806/JIB.V13I1.1147

Che, S., Boyer, M., Meng, J., Tarjan, D., Sheaffer, J. W., & Skadron, K. (2020). A performance study of general-purpose applications on graphics processors using CUDA. *Journal of Parallel and Distributed Computing*, *68*(10), 1370–1380. https://doi.org/10.1016/J.JPDC.2008.05.014

Fauziah, L., Firmansyah, A., & Aguswin, A. (2024). Sistem Informasi Sekolah Berbasis Web Menggunakan Metode Waterfall. Studi Kasus: SMPI Al-Hudri Walibrah. *REMIK: Riset Dan E-Jurnal Manajemen Informatika Komputer*, *8*(1), 274–285.

Hakim, A. B., Hermawan, F., Muhammad, M., & Firmansyarif, R. (2024). Rancang Bangun Aplikasi Penjualan Ayam Geprek R3 Berbasis Web dengan Metode Waterfall. *Jurnal Teknologi Informatika Dan Komputer*, *10*(1), 147–159. https://doi.org/10.37012/JTIK.V10I1.2005

Harris, S. L., & Harris, D. (2021). Digital Design and RISC-V Computer Architecture Textbook. *2021 ACM/IEEE Workshop on Computer Architecture Education, WCAE 2021*. https://doi.org/10.1109/WCAE53984.2021.9707615

Karimi, K., Dickson, N. G., & Hamze, F. (2022). *A Performance Comparison of CUDA and OpenCL*. http://arxiv.org/abs/1005.2581

Memeti, S., Li, L., Pllana, S., Kołodziej, J., & Kessler, C. (2021). Benchmarking OpenCL, OpenACC, OpenMP, and CUDA: Programming Productivity, Performance, and Energy Consumption. *ARMS-CC 2021 - Proceedings of the 2021 Workshop on Adaptive Resource Management and Scheduling for Cloud Computing, Co-Located with PODC 2021*, 1–6. https://doi.org/10.1145/3110355.3110356

Owens, J. D., Houston, M., Luebke, D., Green, S., Stone, J. E., & Phillips, J. C. (2020). GPU computing. *Proceedings of the IEEE*, *96*(5), 879–899. https://doi.org/10.1109/JPROC.2008.917757

Torres, L. A., H, C. J. B., & Denneulin, Y. (2024). *Evaluation of computational and energy performance in matrix multiplication algorithms on CPU and GPU using MKL, cuBLAS and SYCL*. http://arxiv.org/abs/2405.17322