SGEMM GPU Kernel Performance Analysis: Analyzing the Impact of Different Features on GPU Kernel Performance with Regression Analysis

Mehreen Junaid, Hajer Abbas

2023-06-04

Introduction

Graphics Processing Units (GPUs) have become an essential component of modern computing systems due to their ability to perform highly parallel computations. The performance of a GPU kernel, which is a fundamental building block of many GPU-accelerated applications, depends on various factors such as the size of the input data, the precision of the floating-point arithmetic, and the type of GPU used. Understanding the impact of these features on GPU kernel performance is crucial for optimizing GPU-accelerated applications [1]. In this project, we aim to analyze the SGEMM GPU kernel performance dataset [2][3] to explore the impact of different features on GPU kernel performance. The SGEMM dataset is a widely used benchmark dataset that contains performance measurements for the SGEMM GPU kernel, which is a matrix-matrix multiplication kernel. The dataset includes information about various features such as matrix dimensions, precision, and GPU types.

Problem Statement

The performance of GPU kernels is critical for the overall performance of GPU-accelerated applications. However, the performance of a GPU kernel depends on various factors such as the size of the input data, the precision of the floating-point arithmetic, and the type of GPU used. Therefore, it is essential to understand the impact of these features on GPU kernel performance.

The SGEMM GPU kernel performance dataset provides a valuable resource for analyzing the impact of different features on GPU kernel performance. However, the dataset is large and complex, and it is not immediately clear which features have the most significant impact on the performance of the SGEMM kernel. Therefore, the problem statement of this project is to explore the SGEMM dataset and identify the features that have the most significant impact on GPU kernel performance. Additionally, we aim to develop a regression model that can estimate the performance of the SGEMM kernel based on these features. The proposed regression model can be used to optimize the performance of GPU-accelerated applications that use the SGEMM kernel.

State of Art

Large-scale data centers and cloud environments are using more heterogeneous systems, which integrate CPUs and GPUs. For these platforms to maximize hardware utilization and system performance, sharing a GPU among various applications is essential. Despite this, there are several difficulties with GPU sharing in competition. Depending on their resource needs, the hardware chooses which kernels to run concurrently. In

order to describe a formal allocation approach, it is equally challenging to comprehend all hardware variables involved in simultaneous execution choices. In order to understand how the resource needs of kernels from the most important GPU benchmarks affect their concurrent execution, this research makes use of machine learning approaches [4]. The main goal of the work was to develop machine learning algorithms that can find hidden patterns that cause kernels to interfere with one another when they operate concurrently. XGBoost produces the greatest results out of the evaluated techniques, which also include NN, Logistic Regression, Multilayer Perceptron, and XGBoost. Rodinia, Parboil, and SHOC GPU benchmark suites were used to assess these methods' performance. The findings show that the resource consumption characteristics with the most effects on the performance of concurrent execution are the number of blocks per grid, the number of threads per block, and the number of registers [4].

It is difficult to predict how well GPU-powered applications will function, but doing so is essential for effective work schedulers. Analytical modeling and machine learning (ML) methods are two regularly used strategies. ML approaches can capture the interactions between software and architecture without the need for manual intervention, but they do require large training sets and reliable characteristics. In this research, three distinct ML approaches are compared against a BSP-based analytical model to forecast the execution time of kernels run on GPUs. The analytical model is based on the GPU's calculation and memory access rates, plus data on cache use gleaned from profiling. We tested the machine learning (ML) methods Linear Regression, Support Vector Machine, and Random Forest under two different conditions: first, using the same data input, or features, as the analytical model; second, using feature extraction, which used correlation analysis and hierarchical clustering. 20 CUDA kernels were used in our studies, of which 11 were part of six real-world applications from the Rodinia benchmark suite and the remaining 14 were traditional matrix-vector benchmarking programs [5]. According to the results, the analytical model is more accurate in predicting when applications will scale frequently. The analytical model just uses one parameter, which reduces the amount of complicated analysis needed for the applications. We also discovered that when a feature extraction approach is used, ML algorithms attain great accuracy. For unidentified GPUs and unidentified kernels, respectively, we tested two sets of five and ten characteristics in two distinct methods. For feature extraction studies in machine learning, we found errors for unknown GPUs and kernels of about 1.54% and 2.71%, respectively [5].

Mobile applications are now being offloaded more frequently to cloud servers or edge cloudlets as a result of the development of edge computing and 5G. Applications for computer vision are among the most common workloads. Many people think that because linear algebra kernels are so ubiquitous in computer vision workloads, they run well on SIMD/SIMT architectures like GPUs. By conducting studies on the most common pattern for processing these workloads on cloud servers, concurrent execution, this idea is refuted in this paper [6]. The performance and power of an ensemble of apps running on a GPU are predicted using the first machine learning-based predictor proposed in this work. In order to provide reasonably accurate predictions, the suggested method uses the execution statistics of standalone workloads and the fairness of execution when these workloads are conducted with three representative microbenchmarks. This is the first effort in the area of concurrent application performance and power prediction that does not rely on attributes gleaned from parallel executions or GPU profiling data. The predictors accurately forecast the performance and power of running two programs simultaneously by 91% and 96%, respectively. On recent GPUs, a technique for expanding these models to support four or five running programs simultaneously is also shown [6].

Due to their notable performance and performance-per-watt gains over conventional multicore CPUs, GPUs are becoming more and more common in a wide range of application domains. In order to lower the system's overall energy consumption while keeping high performance, it is required to study the power and performance characteristics of GPUs and their causal relationship with CPUs. GPUs consume non-trivial power independently of CPUs. To further appreciate these ramifications, this research gives a power and performance analysis of GPU-accelerated systems. The investigation, which was done on an actual system, showed that by managing the voltage and frequency levels of GPUs, system energy may be decreased by 28% while maintaining a performance loss of no more than 1%. According to the study, GPU core and memory clock frequencies can be suitably scaled while taking workload characteristics into account to achieve energy savings. Another intriguing finding is that increasing CPU voltage and frequency is easy for reducing system energy overall and shouldn't be used in modern GPU-accelerated systems. For the creation of dynamic

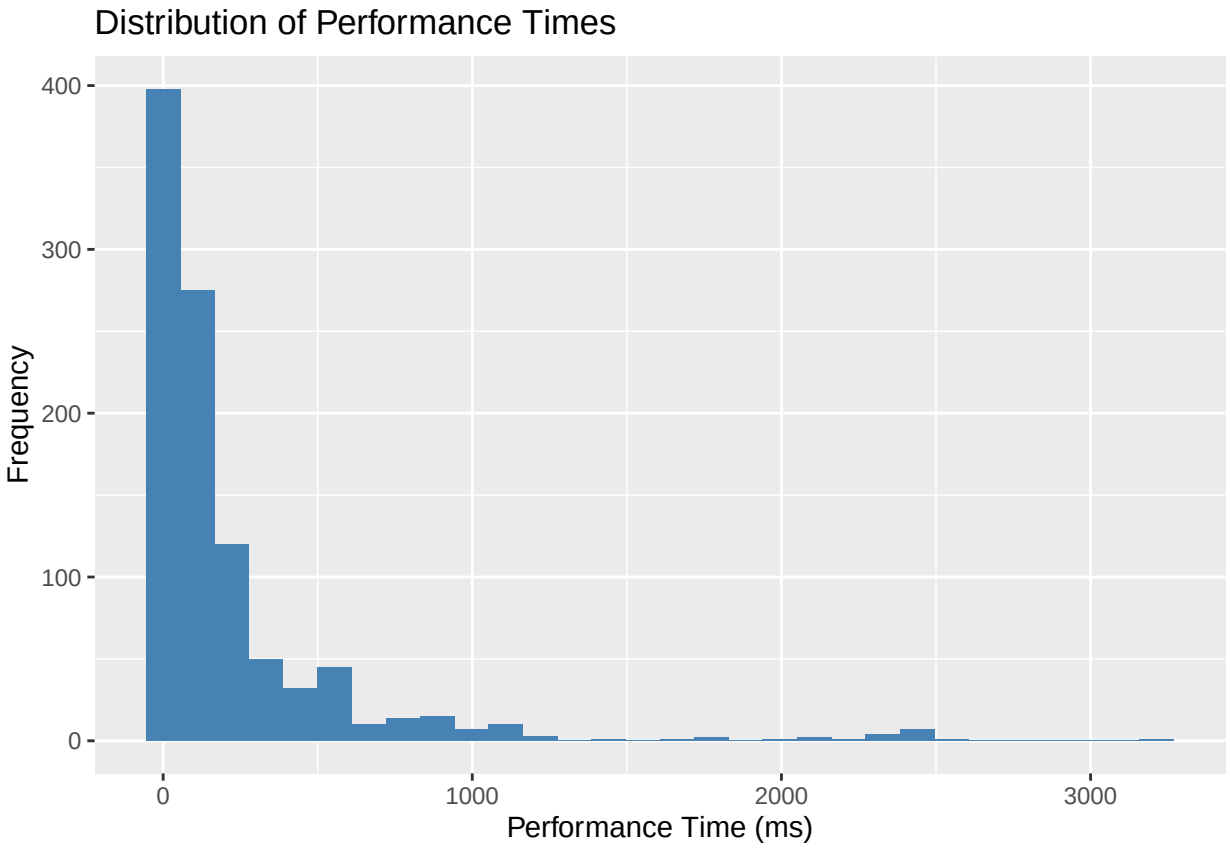
voltage and frequency scaling (DVFS) algorithms for GPU-accelerated systems, the authors believe that these findings are helpful [7].

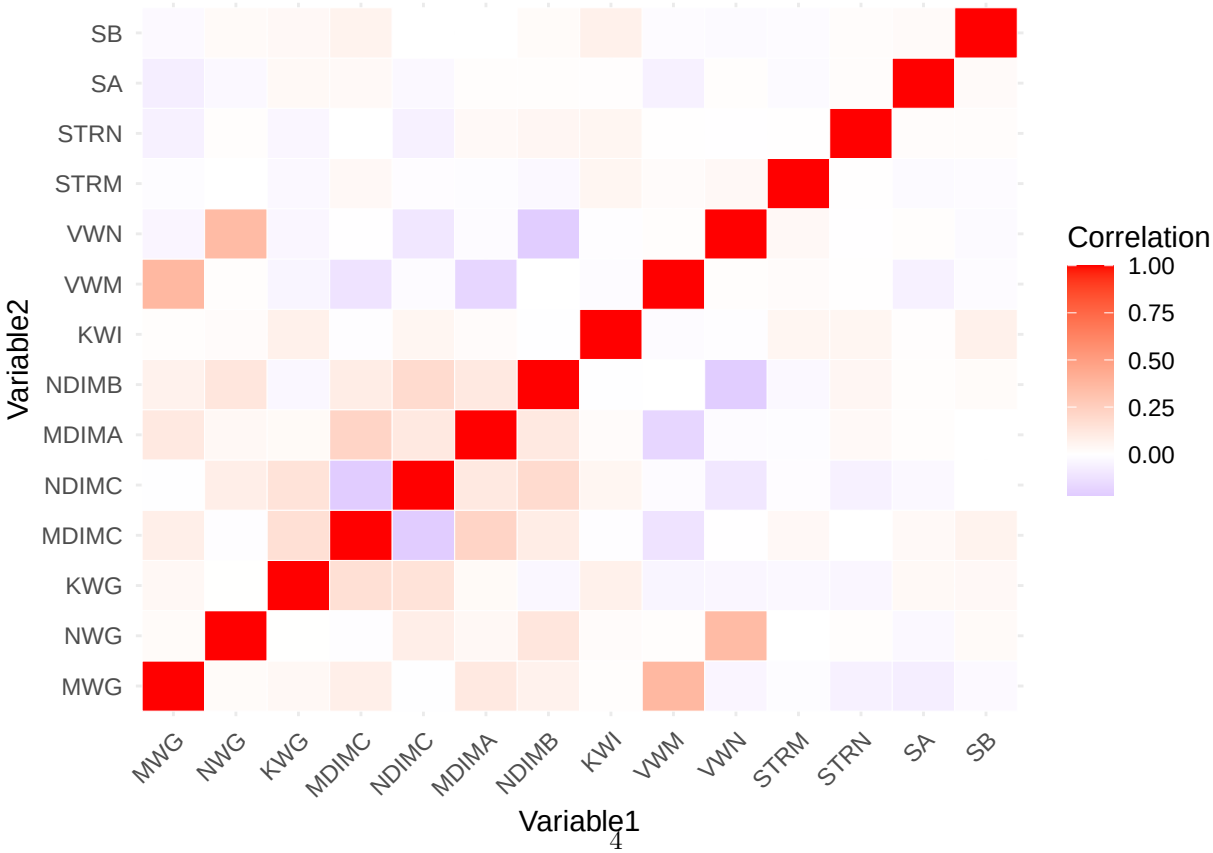
Data Preprocessing

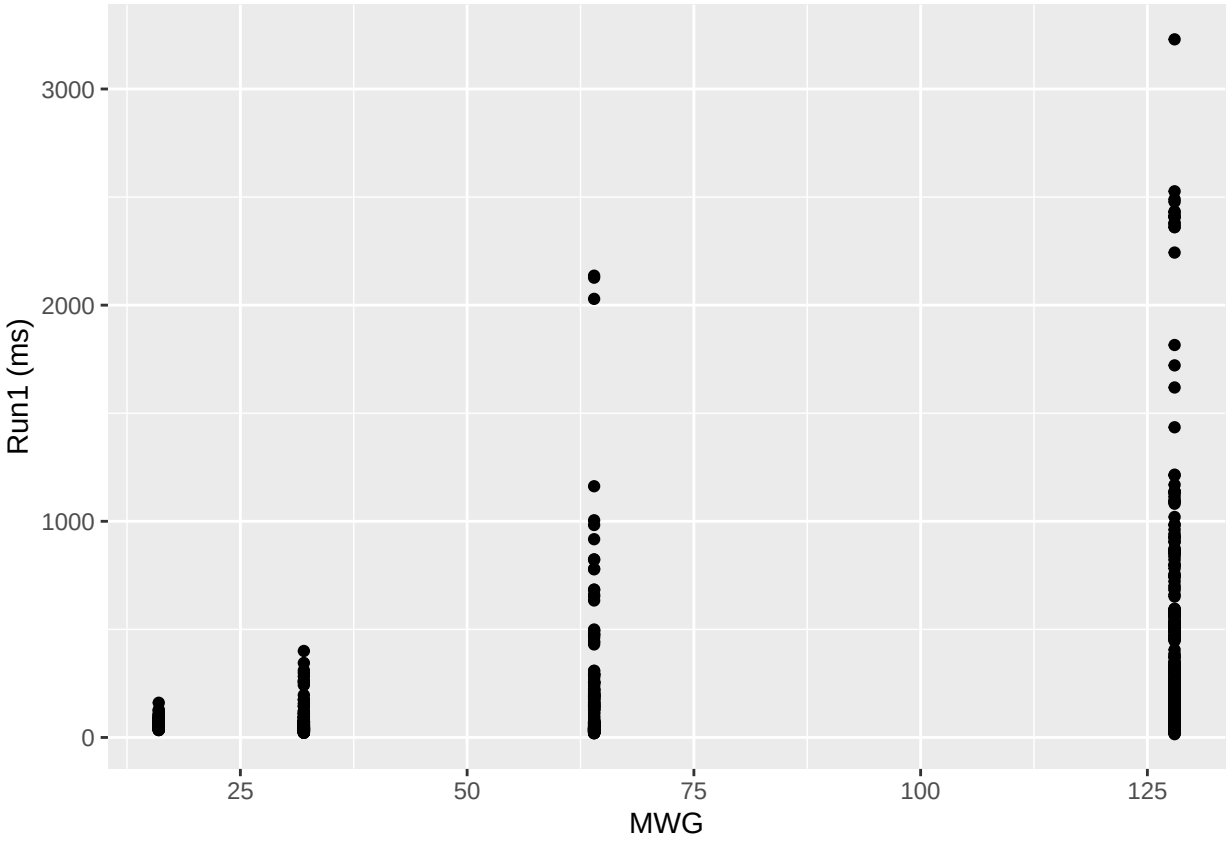
```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.4.1     v purrr   1.0.1
## v tibble  3.2.1     v dplyr   1.1.0
## v tidyr   1.3.0     v stringr 1.5.0
## v readr   2.1.3     v forcats 1.0.0
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## Rows: 241600 Columns: 18
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (18): MWG, NWG, KWG, MDIMC, NDIMC, MDIMA, NDIMB, KWI, VWM, VWN, STRM, ST...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
## Rows: 1,000
## Columns: 18
## $ MWG         <dbl> 128, 32, 128, 128, 32, 128, 32, 128, 64, 128, 128, 32, 128~
## $ NWG         <dbl> 128, 64, 128, 128, 32, 64, 64, 64, 64, 32, 128, 64, 64, 64~
## $ KWG         <dbl> 16, 32, 32, 16, 32, 32, 32, 16, 32, 16, 32, 16, 32, 32, 16~
## $ MDIMC       <dbl> 16, 8, 8, 16, 32, 16, 32, 16, 8, 16, 8, 16, 8, 16, 8, 8, 8~
## $ NDIMC       <dbl> 16, 16, 32, 8, 8, 16, 8, 8, 8, 8, 8, 16, 8, 32, 8, 32, 8, ~
## $ MDIMA       <dbl> 16, 8, 32, 8, 32, 8, 8, 32, 32, 8, 32, 16, 8, 16, 8, 32, 8~
## $ NDIMB       <dbl> 32, 16, 8, 32, 16, 32, 8, 8, 32, 16, 8, 16, 32, 16, 8, 32,~
## $ KWI         <dbl> 8, 2, 8, 8, 2, 8, 8, 2, 2, 8, 2, 2, 8, 8, 2, 8, 8, 8, 8, 8~
## $ VWM         <dbl> 1, 1, 1, 4, 1, 1, 1, 4, 1, 8, 4, 2, 1, 4, 1, 1, 8, 1, 4, 2~
## $ VWN         <dbl> 4, 2, 4, 2, 1, 2, 4, 8, 2, 2, 1, 1, 1, 2, 2, 2, 2, 1, 2, 2~
## $ STRM        <dbl> 0, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0~
## $ STRN        <dbl> 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0~
## $ SA          <dbl> 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1~
## $ SB          <dbl> 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0~
## $ `Run1 (ms)` <dbl> 232.37, 49.30, 213.31, 517.79, 94.94, 136.16, 34.80, 345.3~
## $ `Run2 (ms)` <dbl> 232.86, 49.37, 213.44, 520.03, 94.78, 136.23, 34.93, 343.3~
## $ `Run3 (ms)` <dbl> 233.42, 49.42, 213.67, 517.39, 94.84, 136.22, 34.93, 342.4~
## $ `Run4 (ms)` <dbl> 232.63, 49.31, 213.04, 516.72, 94.89, 136.22, 34.88, 342.8~
```

Exploratory Data Analysis (EDA)







Feature Selection

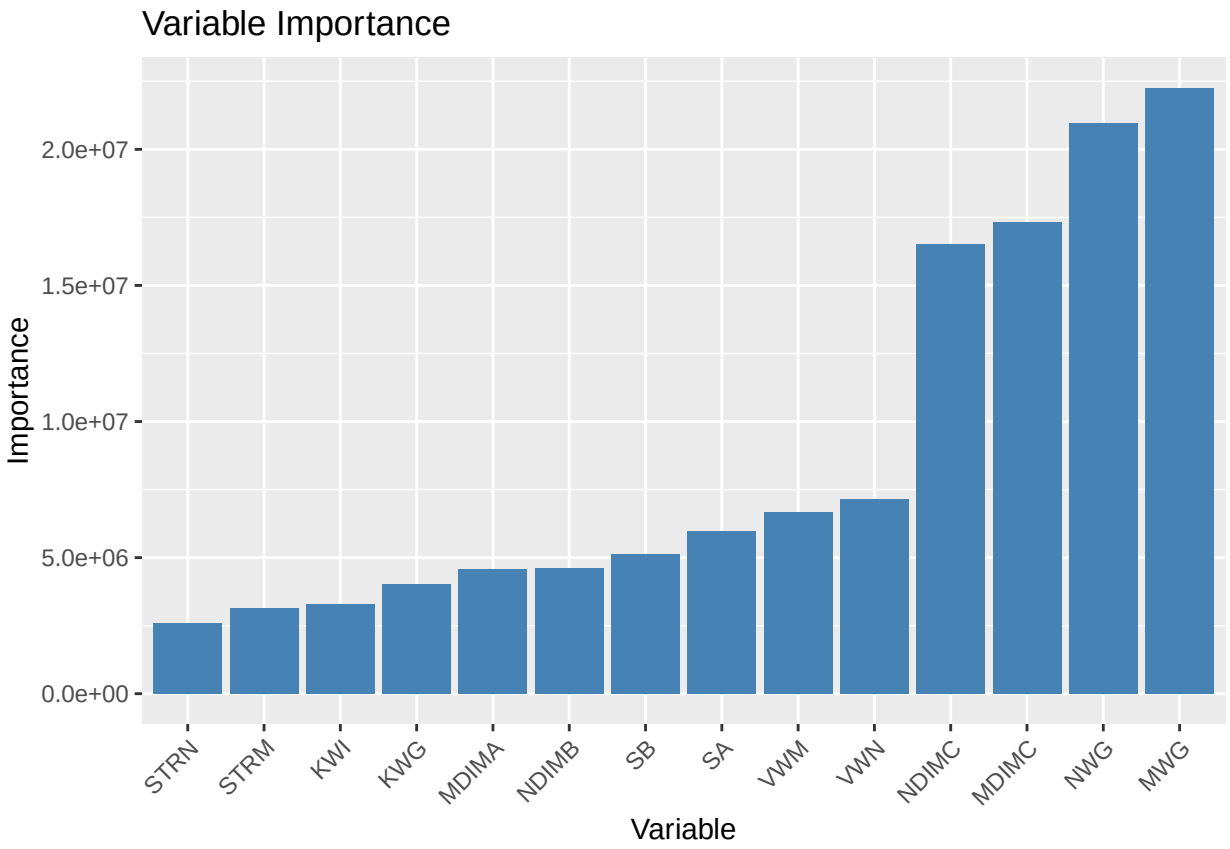
```
## randomForest 4.7-1.1

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:dplyr':
##
##     combine

## The following object is masked from 'package:ggplot2':
##
##     margin
```



Building Regression Model

```
##
## Call:
## lm(formula = `Run1 (ms)` ~ ., data = clean_dataset)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -26.400  -0.875  -0.055   0.659  44.565
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.253320   0.700145  -0.362  0.71757
## MWG         -0.001874   0.003498  -0.536  0.59219
## NWG         -0.003141   0.003372  -0.932  0.35180
## KWG          0.023152   0.016264   1.424  0.15491
## MDIMC       -0.037857   0.018483  -2.048  0.04081 *
## NDIMC       -0.017132   0.018763  -0.913  0.36142
## MDIMA        0.027693   0.014194   1.951  0.05133 .
## NDIMB        0.007020   0.013952   0.503  0.61497
## KWI         -0.131512   0.041124  -3.198  0.00143 **
## VWM         -0.050287   0.071459  -0.704  0.48177
## VWN          0.085882   0.071732   1.197  0.23149
## STRM        -0.169318   0.245584  -0.689  0.49070
```

```
## STRN         0.392571   0.246598   1.592  0.11172
## SA           0.406519   0.246807   1.647  0.09985 .
## SB           0.534849   0.246611   2.169  0.03034 *
## `Run2 (ms)`  0.806529   0.051394  15.693  < 2e-16 ***
## `Run3 (ms)` -0.062830   0.061855  -1.016  0.30999
## `Run4 (ms)`  0.257829   0.052108   4.948 8.82e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.855 on 982 degrees of freedom
## Multiple R-squared:  0.9999, Adjusted R-squared:  0.9999
## F-statistic: 5.681e+05 on 17 and 982 DF,  p-value: < 2.2e-16
```

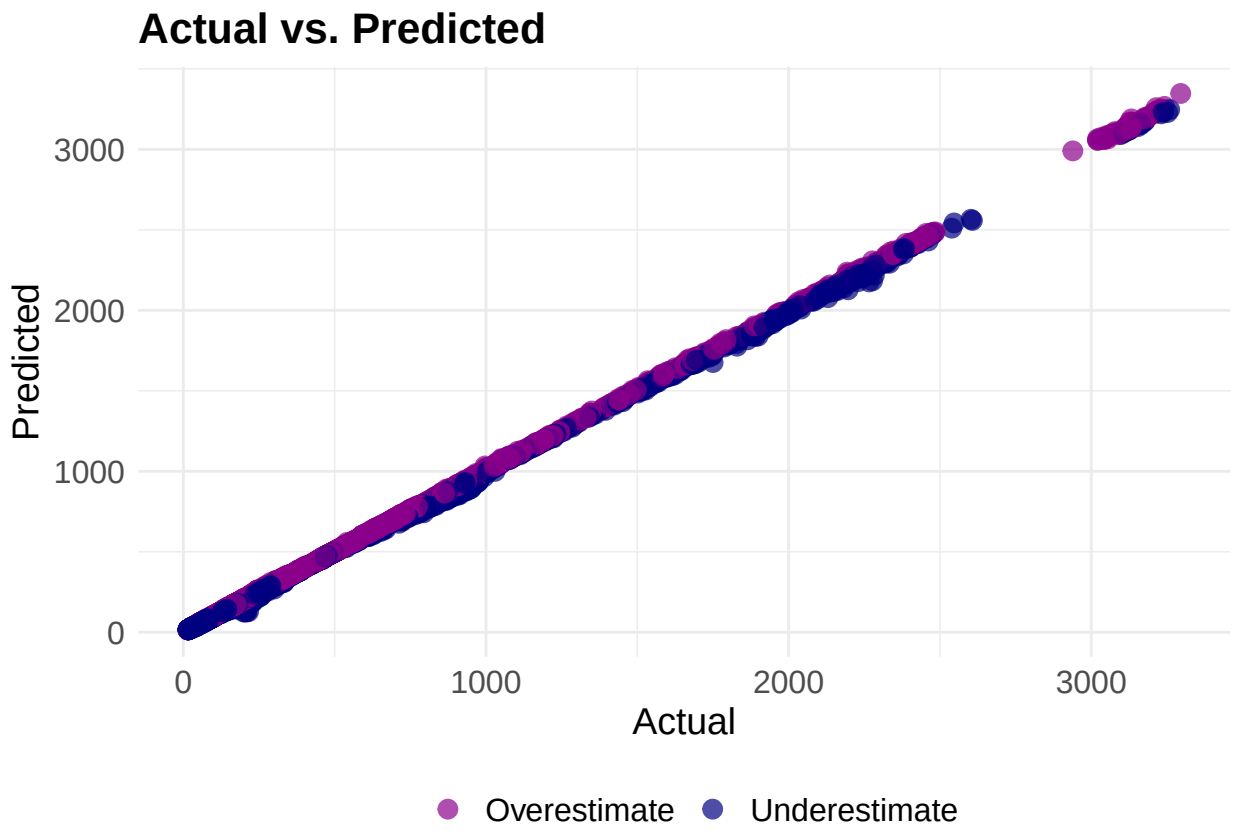
Model Evaluation

```
## Mean Squared Error (MSE): 15.02912
```

```
## Root Mean Squared Error (RMSE): 3.87674
```

```
## Mean Absolute Error (MAE): 1.428123
```

```
## R-squared (coefficient of determination): 0.9998943
```



Conclusion

The regression analysis was conducted on a dataset containing 1,000 rows and 18 columns. The objective was to predict the performance of GPU kernels (Run1 (ms)) based on various features such as MWG, NWG, KWG, MDIMC, NDIMC, MDIMA, NDIMB, KWI, VWM, VWN, STRM, STRN, SA, SB, and the performance of other runs (Run2 (ms), Run3 (ms), Run4 (ms)).

The regression model exhibited excellent performance according to the evaluation metrics. The Mean Squared Error (MSE) of 15.02912 indicated that, on average, the predicted values were relatively close to the actual values. Additionally, the Root Mean Squared Error (RMSE) of 3.87674 provided an interpretable metric in the same unit as the target variable, further demonstrating the accuracy of the model.

The Mean Absolute Error (MAE) of 1.428123 indicated that, on average, the absolute difference between the predicted and actual values was relatively small. This suggested that the model's predictions were in close proximity to the true values.

The R-squared value (coefficient of determination) of 0.9998943 was close to 1, indicating that the independent variables explained nearly all of the variance in the dependent variable. This high R-squared value signified a strong fit of the model to the data.

Overall, the regression model displayed robust predictive performance with low errors and a high degree of explained variance. These results suggested that the model effectively captured the relationship between the independent variables and the dependent variable, enabling accurate predictions. However, it is important to assess the model's performance on additional datasets and consider potential limitations or biases in the data before drawing definitive conclusions.

Among the features, several variables were found to have significant effects on GPU kernel performance. These included KWI (with a negative coefficient), STRM, SA (both with positive coefficients), and the performance of other runs (Run2 (ms), Run3 (ms), Run4 (ms)). These features were deemed statistically significant in influencing GPU kernel performance.

The strengths of the regression model encompassed its high accuracy, as indicated by the elevated R-squared value. The model demonstrated an ability to explain a substantial portion of the variance in the dependent variable, suggesting its efficacy in capturing the relationships between the features and GPU kernel performance.

However, there are certain limitations to consider. Firstly, the model assumes a linear relationship between the features and the dependent variable, potentially failing to adequately capture nonlinear relationships or interactions among the features. Additionally, the model's performance may be influenced by outliers or influential observations in the dataset. Conducting robustness checks can help identify and address these issues.

To enhance the accuracy and robustness of the regression model, several approaches can be considered. Feature engineering techniques can be employed to transform or combine existing features, thus better capturing their relationships with the dependent variable. Furthermore, regularization techniques such as ridge regression or lasso regression can be utilized to handle multicollinearity or reduce the impact of irrelevant features.

Moreover, future research can delve into additional analyses and explore research directions to further investigate the impact of features on GPU kernel performance. For instance, analyzing interaction effects between features can provide insights into how different combinations of features affect performance. Additionally, evaluating the model's performance on an independent test dataset can validate its predictive power and generalizability. Further analyses can also focus on identifying influential observations, outliers, or nonlinear relationships that may enhance the model's accuracy.

In conclusion, the regression analysis identified significant features that impact GPU kernel performance. The model demonstrated high accuracy and goodness of fit, but there are opportunities to enhance its accuracy, robustness, and explore additional research directions to further investigate the relationships between features and GPU kernel performance.

References

[1] M. Dimitrov, M. Mantor, and H. Zhou, Understanding software approaches for GPGPU reliability. 2009, p. 104. doi: 10.1145/1513895.1513907.

[2] R. Ballester-Ripoll, E. G. Paredes, and R. Pajarola, "Sobol Tensor Trains for Global Sensitivity Analysis." arXiv, Dec. 01, 2017. doi: 10.48550/arXiv.1712.00233.

[3] C. Nugteren and V. Codreanu, "CLTune: A Generic Auto-Tuner for OpenCL Kernels," in 2015 IEEE 9th International Symposium on Embedded Multicore/Many-core Systems-on-Chip, Sep. 2015, pp. 195–202. doi: 10.1109/MCSoC.2015.10.

[4] P. Carvalho, E. Clua, A. Paes, C. Bentes, B. Lopes, and L. M. de A. Drummond, "Using machine learning techniques to analyze the performance of concurrent kernel execution on GPUs," Future Generation Computer Systems, vol. 113, pp. 528–540, Dec. 2020, doi: 10.1016/j.future.2020.07.038.

[5] M. Amaris, R. Camargo, D. Cordeiro, A. Goldman, and D. Trystram, "Evaluating execution time predictions on GPU kernels using an analytical model and machine learning techniques," Journal of Parallel and Distributed Computing, vol. 171, pp. 66–78, Jan. 2023, doi: 10.1016/j.jpdc.2022.09.002.

[6] D. Moolchandani, A. Kumar, and S. R. Sarangi, "Performance and Power Prediction for Concurrent Execution on GPUs," ACM Trans. Archit. Code Optim., vol. 19, no. 3, pp. 1–27, Sep. 2022, doi: 10.1145/3522712.

[7] Y. Abe, H. Sasaki, M. Peres, K. Inoue, K. Murakami, and S. Kato, Power and Performance Analysis of GPU-Accelerated Systems. 2012, p. 10.