Optimización del Rendimiento de Multiplicación General de Matrices (GEMM) en CPU-GPU: Predicción y Análisis de Parámetros

G. Lozano
Ingeniero Informático
PUCP
Lima, Perú
glozanot@pucp.edu.pe

R. Tunque
Ingeniero Informático
PUCP
Lima, Perú
ronaldo.tunque@pucp.edu.pe

H. Sánchez
Físico
PUCP
Lima, Perú
hdsanchez@pucp.edu.pe

***Abstract*—En progreso.**
***Index Terms*—**

I. Introducción

La multiplicación general de matrices (GEMM) es una operación fundamental en muchos dominios científicos y de ingeniería, incluyendo el aprendizaje automático, la simulación física y el procesamiento de señales. GEMM implica la multiplicación de dos matrices y es conocida por ser intensiva en cómputo y memoria. La eficiencia de esta operación puede verse significativamente afectada por la parametrización y configuración de los kernels que se ejecutan en la GPU. En este contexto, un kernel es un programa que se ejecuta en la GPU y está diseñado para realizar operaciones específicas.

El rendimiento de GEMM en sistemas que combinan CPU y GPU es de gran importancia para mejorar la eficiencia de diversas aplicaciones. Sin embargo, debido a la gran cantidad de combinaciones posibles de parámetros y la complejidad inherente de las interacciones entre estos parámetros, encontrar la configuración óptima que minimice el tiempo de ejecución es un desafío significativo. Tradicionalmente, este problema se ha abordado mediante la experimentación manual y métodos heurísticos, que pueden ser ineficaces y requerir mucho tiempo.

El aprendizaje automático (ML) ofrece una solución eficiente al problema de optimización del tiempo de ejecución de GEMM a través de modelos predictivos. Estos modelos, construidos mediante técnicas de regresión supervisada como regresión lineal, polinómica, bosques aleatorios y algoritmos de incremento de gradiente, pueden predecir el tiempo de ejecución de GEMM para diversas configuraciones de parámetros.

Además, parte de estos modelos proporcionan información relevante para realizar un análisis sobre la importancia de algunos parámetros sobre otros, identificando aquellos que son críticos para optimizar el rendimiento. Al construir modelos predictivos confiables, se pueden aplicar técnicas de optimización como la optimización bayesiana para encontrar configuraciones que minimicen el tiempo de ejecución. Estos modelos también pueden generalizarse para diferentes arquitecturas de hardware y tipos de matrices, lo que permite su aplicación en diversos entornos y aplicaciones.

A. Objetivo del Estudio

El objetivo principal de este estudio es aplicar técnicas de aprendizaje automático para predecir y optimizar el rendimiento de GEMM en un entorno de computación con las características descritas en la sección III-A. Para alcanzar este propósito, se definen los siguientes sub-propósitos específicos:

1) Predicción del Tiempo de Ejecución de GEMM: Construir y entrenar modelos de regresión que puedan predecir con precisión el tiempo de ejecución de GEMM en función de una variedad de configuraciones de parámetros.

2) Identificación de Parámetros Críticos: Determinar cuáles de los 14 parámetros tienen el mayor impacto en el tiempo de ejecución usando técnicas de interpretación de modelos.

3) Optimización del Rendimiento: Aplicar técnicas de optimización para encontrar la configuración de parámetros que minimice el tiempo de ejecución de GEMM.

4) Validación y Evaluación De Resultados: Evaluar el rendimiento de los modelos predictivos con métricas como el error cuadrático medio (MSE) y el coeficiente de determinación (R^2), y realizar validaciones cruzadas para asegurar la robustez y generalización de los modelos.

5) Generación de Conocimientos para Optimización: Proporcionar recomendaciones sobre las configuraciones de parámetros más eficientes para GEMM en sistemas CPU-GPU y generar conocimientos aplicables a otras operaciones similares y en diferentes arquitecturas de hardware.

B. Organización del Artículo

Este informe está estructurado en las siguientes secciones:

1) ***Sección II: Estado del Arte****:* Presenta una síntesis de los aportes realizados por otros artículos científicos en el campo de la optimización del rendimiento de GEMM y el uso de técnicas de aprendizaje automático.

*2) **Sección III: Diseño del Experimento**:* Describe el conjunto de datos utilizado, incluyendo el número y tipo de características, así como la estadística descriptiva y visualización de los datos. Además, se detalla la metodología, cubriendo la estrategia para el manejo de datos faltantes, la selección y extracción de características, la selección y justificación de la medida de calidad, los algoritmos empleados y la estrategia para su ajuste, y la estrategia de validación utilizada para el ajuste de hiperparámetros.

II. Estado del Arte

En el campo de la predicción y optimización del rendimiento de la multiplicación general de matrices (GEMM) en GPUs, la investigación ha avanzado significativamente mediante el uso de técnicas de aprendizaje automático. Los estudios revisados han demostrado que tanto las técnicas supervisadas como las no supervisadas pueden predecir con precisión el rendimiento de los kernels SGEMM, proporcionando una base sólida para optimizar el uso de recursos en aplicaciones de cómputo intensivo. Investigaciones como las de Agrawal et al. [1] han aplicado modelos de aprendizaje automático para identificar patrones de rendimiento y guiar la optimización en sistemas GPU.

El conjunto de datos "SGEMM GPU Kernel Performance" [5], el cual también será empleado en el presente proyecto, fue aplicado en otras investigaciones donde ha permitido a los investigadores desarrollar y probar modelos que predicen el rendimiento de los kernels SGEMM con alta precisión. Agrawal et al. [1] utilizaron este conjunto de datos para aplicar técnicas de aprendizaje supervisado y no supervisado, logrando predicciones precisas que demostraron la viabilidad de usar modelos de ML para guiar la optimización del rendimiento.

Por otro lado, Carvalho et al. [2] emplearon un conjunto de datos similar para analizar la ejecución concurrente de kernels en GPUs, hallando configuraciones óptimas que mejoran significativamente la eficiencia del sistema y reducen los tiempos de ejecución. Estos estudios subrayan la importancia de estos conjunto de datos como una herramienta valiosa para la investigación en la optimización del rendimiento de GPU mediante técnicas de aprendizaje automático.

Además, otros investigadores han explorado diferentes enfoques para mejorar el rendimiento de GEMM en GPUs. Li et al. [3] desarrollaron un esquema de prefetching de grano fino con capacidades de autoajuste para kernels DGEMM, que mejoró significativamente el rendimiento al reducir los tiempos de espera de la memoria y optimizar el uso del hardware. Asimismo, Matsumoto et al. [4] y Volkov y Demmel [6] presentaron estrategias de optimización multinivel y marcos de referencia para evaluar y ajustar el rendimiento de GPUs en operaciones de álgebra lineal densa.

En resumen, el uso de técnicas de aprendizaje automático ha demostrado ser eficaz en la predicción y optimización del rendimiento de GEMM en GPUs. Se evidencia la existencia de estudios que han utilizado el conjunto de datos "SGEMM GPU Kernel Performance", o similares, para desarrollar modelos precisos y encontrar configuraciones óptimas que mejoran la eficiencia del sistema, destacando su relevancia en la investigación de optimización del rendimiento de GPU.

III. Diseño del Experimento

A. Descripción Del Conjunto De Datos

El conjunto de datos utilizado en este estudio es el "SGEMM GPU Kernel Performance" [5], que mide el tiempo de ejecución de un producto de matrices $A \times B = C$ utilizando un kernel SGEMM GPU parametrizable. Todas las matrices tienen un tamaño de 2048 x 2048. Se probaron 241,600 combinaciones de parámetros, y para cada combinación se realizaron 4 ejecuciones, cuyos tiempos se reportan en las últimas 4 columnas del conjunto de datos. Todos los tiempos se miden en milisegundos. El experimento se realizó en una computadora de escritorio con Ubuntu 16.04 Linux, equipada con un procesador Intel Core i5 (3.5GHz), 16GB de RAM y un GPU NVidia Geforce GTX 680 de 4GB GF580 GTX-1.5GB.

TABLE I: Parámetros del kernel de GPU y sus descripciones

Nombre	Descripción	Tipo	Valores Posibles
MWG	Ancho de la matriz global	Número	16, 32, 64, 128
NWG	Ancho de la matriz global	Número	16, 32, 64, 128
KWG	Ancho del kernel global	Número	16, 32
MDIMC	Tamaño de la dimensión de la matriz para el bucle más interno en M	Número	8, 16, 32
NDIMC	Tamaño de la dimensión de la matriz para el bucle más interno en N	Número	8, 16, 32
MDIMA	Dimensión de la matriz A	Número	8, 16, 32
NDIMB	Dimensión de la matriz B	Número	8, 16, 32
KWI	Factor de desenrollado del bucle del kernel	Número	2, 8
VWM	Ancho del vector para la matriz A	Número	1, 2, 4, 8
VWN	Ancho del vector para la matriz B	Número	1, 2, 4, 8
STRM	Habilitar el desplazamiento para la matriz A	Binario	0, 1
STRN	Habilitar el desplazamiento para la matriz B	Binario	0, 1
SA	Habilitar el almacenamiento en caché manual para la matriz A	Binario	0, 1
SB	Habilitar el almacenamiento en caché manual para la matriz B	Binario	0, 1

El conjunto de datos consta de 18 características en total, que comprenden 14 parámetros de entrada y 4 resultados de tiempo de ejecución. Entre los parámetros de entrada (ver la tabla I), los primeros 10 son ordinales y pueden tomar hasta 4 valores diferentes, todos ellos potencias de dos, e incluyen variables como MWG, NWG, KWG, MDIMC, NDIMC, MDIMA, NDIMB, KWI, VWM y VWN. Los últimos 4 parámetros son binarios e incluyen STRM, STRN, SA y SB. Los resultados de tiempo de ejecución están representados por las columnas Run1 (ms), Run2 (ms), Run3 (ms) y Run4 (ms). La imagen 1 muestra la distribución de los tiempos de ejecución para cada configuración de los parámetros. La tabla II presenta una descripción estadística de estos tiempos de ejecución, incluyendo la media, mediana, desviación estándar, y los valores mínimo y máximo.

TABLE II: Estadísticas descriptivas de los tiempos de ejecución

Estadística	Run1	Run2	Run3	Run4
Media	217.65	217.58	217.53	217.53
Mediana	69.82	69.93	69.79	69.82
Desviación Estándar	369.01	368.68	368.66	368.68
Min	13.29	13.25	13.36	13.37
Max	3339.63	3375.42	3397.08	3361.71

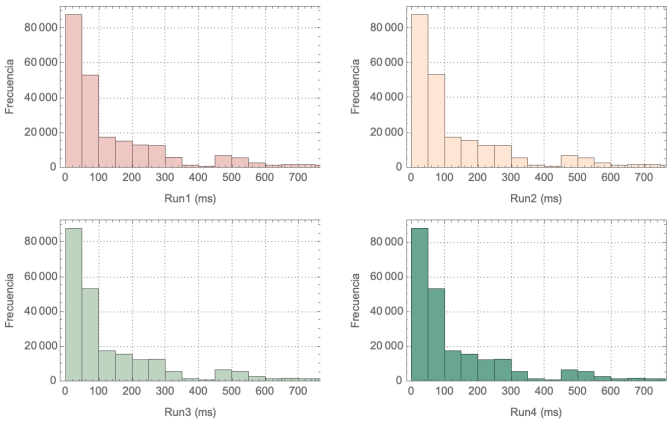

Fig. 1: Histograma de los tiempos de ejecución

El conjunto de datos contiene un total de 241,600 muestras. La práctica común en la división de conjuntos de entrenamiento y prueba es utilizar una proporción estándar, como 80/20 o 70/30. En este caso, se asume una división del 80/20. Por lo tanto, el Conjunto de Entrenamiento comprende el 80

B. Metodología

A continuación, se describe la metodología que se empleará en este proyecto:

1) ***Estrategia para el Manejo de Datos Faltantes****:* No existen datos faltantes en el dataset.

2) ***Selección y Extracción de Características****:* La selección de características se realizarán utilizando técnicas como:

- Análisis de importancia de características: Emplearemos modelos de aprendizaje automático (por ejemplo, bosques aleatorios) para determinar la importancia de cada característica.
- Métodos de filtrado y envoltura: Se evaluará la relación entre las características y la variable objetivo, como la correlación o la información mutua.
- Reducción de dimensionalidad: Se utilizarán técnicas como Análisis de Componentes Principales (PCA) o Análisis de Discriminante Lineal (LDA) para reducir la dimensionalidad de los datos.

3) ***Selección y Justificación de la Medida de Calidad****:* Para evaluar el rendimiento de los modelos, se utilizarán medidas de calidad adecuadas al tipo de problema. Dado que estamos abordando un problema de regresión para predecir el tiempo de ejecución de la multiplicación de matrices, se emplearán métricas como el error cuadrático medio (MSE) y el coeficiente de determinación (R^2).

4) ***Algoritmos que serán Empleados y Estrategia para su Ajuste****:* Se utilizarán varios algoritmos de aprendizaje automático, como regresión lineal, regresión polinómica, bosques aleatorios y algoritmos de incremento de gradiente (por ejemplo, XGBoost, LightGBM). Se ajustarán los hiperparámetros de estos modelos utilizando la búsqueda de cuadrícula (Grid Search) o la optimización bayesiana para encontrar la configuración óptima.

5) ***Estrategia de Validación a Emplear para el Ajuste de Hiperparámetros****:* Para el ajuste de hiperparámetros, se aplicará la validación cruzada, dividiendo el conjunto de datos en subconjuntos de entrenamiento y prueba. Se empleará la validación cruzada estratificada para garantizar una distribución uniforme de las etiquetas en cada pliegue. Esto ayudará a evitar el sobreajuste y proporcionará una evaluación más confiable del rendimiento del modelo.

IV. Resultados

V. Discusión

VI. Conclusiones

Agradecimientos: A Cesar Olivares por su apoyo y validación en el desarrollo del presente proyecto.

References

[1] Agrawal, S., Bansal, A., Rathor, S.: Prediction of sgemm gpu kernel performance using supervised and unsupervised machine learning techniques. In: 2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT). pp. 1–7. IEEE (2018)

[2] Carvalho, P., Clua, E., Paes, A., Bentes, C., Lopes, B., Drummond, L.M.d.A.: Using machine learning techniques to analyze the performance of concurrent kernel execution on gpus. Future Generation Computer Systems **113**, 528–540 (2020)

[3] Li, J., Ye, H., Tian, S., Li, X., Zhang, J.: A fine-grained prefetching scheme for dgemm kernels on gpu with auto-tuning compatibility. In: 2022 IEEE International Parallel and Distributed Processing Symposium (IPDPS). pp. 863–874. IEEE (2022)

[4] Matsumoto, K., Nakasato, N., Sakai, T., Yahagi, H., Sedukhin, S.G.: Multi-level optimization of matrix multiplication for gpu-equipped systems. Procedia Computer Science **4**, 342–351 (2011)

[5] Paredes, E., Ballester-Ripoll, E.I.: Sgemm gpu kernel performance. UCI Machine Learning Repository (2018)

[6] Volkov, V., Demmel, J.W.: Benchmarking gpus to tune dense linear algebra. In: SC'08: Proceedings of the 2008 ACM/IEEE conference on Supercomputing. pp. 1–11. IEEE (2008)