Análisis estadístico de un conjunto de datos sobre el rendimiento de un programa para el cálculo de multiplicación de matrices en GPU

Estadística y Métodos Matemáticos para la Investigación:
Introducción a las Técnicas Estadísticas

Estíbaliz Parcero Iglesias

24 de abril de 2023

1. Introducción

En el presente trabajo se expone el análisis estadístico de un conjunto de datos sobre el rendimiento de un programa para el cálculo de multiplicación de matrices en GPU [1][7].

Para la generación de este conjunto de datos se ha medido el tiempo de ejecución de un producto de matrices A * B = C, donde todas las matrices tienen un tamaño de 2048 x 2048, utilizando un kernel SGEMM GPU parametrizable con 241 600 posibles combinaciones de parámetros. Para cada combinación probada, se realizaron 4 ejecuciones, cuyos resultados corresponden a las últimas 4 columnas del conjunto de datos. Todos los tiempos se miden en milisegundos.

Hay 14 parámetros, los primeros 10 son ordinales y solo pueden tomar hasta 4 valores diferentes de potencias de dos, y las últimas 4 variables son binarias. A continuación se describe cada uno de los parámetros:

- MWG: dimensión M del kernel (por ejemplo, 64, 128)
- NWG: dimensión N del kernel (por ejemplo, 64, 128)
- KWG : dimensión K del kernel (por ejemplo, 8, 16)
- MDIMC: *threads* por grupo de trabajo en la dimensión M (por ejemplo, 8, 16, 32)

- NDIMC: *threads* por grupo de trabajo en la dimensión N (por ejemplo, 8, 16, 32)
- MDIMA: dimensión resultante de la matriz A: KDIMA * MDIMA
- NDIMB : dimensión resultante de la matriz B: KDIMB * NDIMB
- KWI : factor de desenrollado del bucle KWG (menor o igual que KWG)
- VWM: ancho del vector de las matrices A y C (compatible con 1, 2, 4 y 8)
- VWN: ancho del vector de la matriz B (compatible con 1, 2, 4 y 8)
- STRM : usa el acceso con *stride* dentro de un *thread* en la dimensión M (1) o no (0)
- STRN : usa acceso con *stride* dentro de un *thread* en la dimensión N (1) o no (0)
- SA: usa la memoria local/compartida para almacenar en caché la matriz A (1) o no (0)
- SB: usa la memoria local/compartida para almacenar en caché la matriz B (1) o no (0)

De un total de 1 327 104 combinaciones de parámetros, solo 241 600 son posibles debido a varias restricciones del kernel. Este conjunto de datos contiene los resultados para todas estas combinaciones factibles.

El experimento se ejecutó en una estación de trabajo de escritorio con Ubuntu 16.04 Linux con un procesador Intel Core i5 (3.5GHz), 16GB RAM y una GPU NVidia Geforce GTX 680 4GB GF580 GTX-1.5GB. Se utilizó el kernel *gemm_fast* de la librería *OpenCL CLTune* [6].

En el presente trabajo de han desarrollado diversos scripts en Python para el análisis estadístico del conjunto de datos descrito. Para ello se han utilizado técnicas habituales del estado del arte[5] mediante herramientas modernas de ciencia de datos[9].

En la sección 2 se describen los objetivos del presente trabajo. En la sección 3 se describe la metodología seguida y el entorno utilizado para la experimentación. En la sección 4.1 se realiza un análisis descriptivo de los datos, en la sección 4.2 se realiza un análisis de la correlación de los distintos parámetros de entrada de los datos y en especial con respecto al tiempo de ejecución, en la sección 4.3 se busca un modelo de ajuste a nuestros datos.

En la sección 5 se resumen los resultados. Y por último, en la sección 6 se exponen las conclusiones.

2. Objetivos

Describir los datos que se utilizarán en el análisis estadístico y proporciona información sobre su origen y calidad.

Averiguar si existe alguna correlación entre los parámetros que se analizarán y cómo se medirá esta correlación.

Describir la distribución de los datos y cómo se utilizará esta información en el análisis estadístico.

3. Metodología

Para llevar a cabo el análisis estadístico, se utilizó el entorno de Google Colab, que proporciona una plataforma en línea para ejecutar código Python en un Jupyter Notebook. Se utilizaron varias librerías especializadas en análisis estadístico, como NumPy [2], Pandas[4], SciPy[10], Scikit-learn[8] y Matplotlib[3] entre otras.

El procedimiento de análisis estadístico se dividió en varias etapas. En primer lugar, se realiza una exploración general de los datos para comprender mejor su estructura y distribución. A continuación, se llevó a cabo un análisis de correlación para determinar si existía alguna relación entre los parámetros que se analizarían. Finalmente, se generó un modelo para la representación de los datos.

4. Desarrollo

4.1. Análisis descriptivo

Se realizó un análisis descriptivo de las últimas 4 columnas del conjunto de datos, correspondientes a los tiempos de ejecución en milisegundos de 4 ejecuciones independientes utilizando los mismos parámetros. Los resultados se muestran en la Tabla 1 e indican que los tiempos de ejecución varían entre un mínimo de 13.25 y un máximo de 3397.08 milisegundos, con una media de 217.57 milisegundos y una desviación estándar de 368.75 milisegundos. Esto sugiere que hay una variabilidad significativa en los tiempos de ejecución para diferentes combinaciones de parámetros, y no se observa diferencia significativa entre las distintas ejecuciones según el orden de ejecución.

Run_i (ms)	1	2	3	4	Mean
count	241600	241600	241600	241600	241600
mean	217.65	217.58	217.53	217.53	217.57
std	369.01	368.68	368.66	368.68	368.75
min	13.29	13.25	13.36	13.37	13.32
25 %	40.66	40.71	40.66	40.64	40.67
50 %	69.82	69.93	69.79	69.82	69.79
75 %	228.53	228.31	228.32	228.32	228.39
max	3339.63	3375.42	3397.08	3361.71	3341.51

Tabla 1: Resultados del análisis descriptivo básico para las ejecuciones y la media de las ejecuciones

A continuación, se generó un histograma para visualizar la distribución de la media de los tiempos de ejecución de las 4 ejecuciones. Inicialmente, se generó un histograma utilizando los valores originales de la media, pero debido a la presencia de valores extremos, no se apreciaban claramente las diferencias en la distribución. Por lo tanto, se decidió transformar los datos aplicando el logaritmo natural a los valores de la media y generar un nuevo histograma utilizando los valores transformados.

El histograma resultante (figura 1) muestra una distribución más clara y permite apreciar mejor las diferencias en los tiempos de ejecución. Además, es posible observar fácilmente que la distribución no es una normal, lo cual puede apoyarse en la prueba de bondad de ajuste de Kolmogorov-Smirnov.

Los resultados de la prueba indican un valor del estadístico K-S de 0.14 y un p-valor de 0.00. Dado que el p-valor es menor que el nivel de significancia comúnmente utilizado de 0.05, se rechaza la hipótesis nula de que los datos siguen una distribución normal.

En el histograma mostrado se observan varios picos que sugieren que la presencia de múltiples subpoblaciones en tus datos. Cada subpoblación podría tener su propia distribución y estar asociada a diferentes combinaciones de parámetros utilizados en las ejecuciones.

4.2. Análisis de correlación

A continuación se presenta un análisis de la correlación entre el tiempo de ejecución de un producto matriz-matriz y los diferentes parámetros que afectan su rendimiento. El objetivo de este análisis es identificar los parámetros más relevantes en cuanto a correlación con el tiempo de ejecución y proporcionar información útil para optimizar el rendimiento del

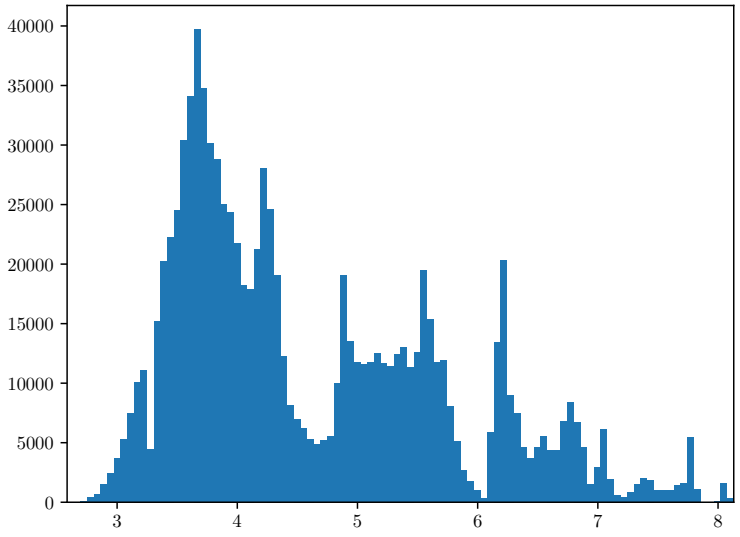

Figura 1: Histograma del logaritmo de la media de los tiempos de ejecución según las diferentes combinaciones de parámetros.

producto matriz-matriz.

Se ha calculado la matriz de correlación entre el tiempo de ejecución y los diferentes parámetros utilizando el conjunto de datos proporcionado. La matriz de correlación muestra la relación lineal entre cada par de variables. Los valores de correlación varían entre -1 y 1, donde -1 indica una correlación negativa perfecta, 0 indica ninguna correlación y 1 indica una correlación positiva perfecta. Como muestra la figura 2, los parámetros que muestran una mayor correlación con el tiempo de ejecución son MWG y NWG.

También se muestra en la tabla 2 el índice de correlación para cada parámetro en relación con el tiempo de ejecución.

Teniendo en cuenta los valores de correlación proporcionados, se puede observar que los parámetros MWG y NWG son los más relevantes en cuanto a correlación con el tiempo de ejecución, con valores de correlación de 0,351805 y 0,320455, respectivamente. Esto indica que existe una relación positiva moderada entre estos dos parámetros y el tiempo de ejecución.

Los parámetros MDIMC y NDIMC también son relevantes en cuan-

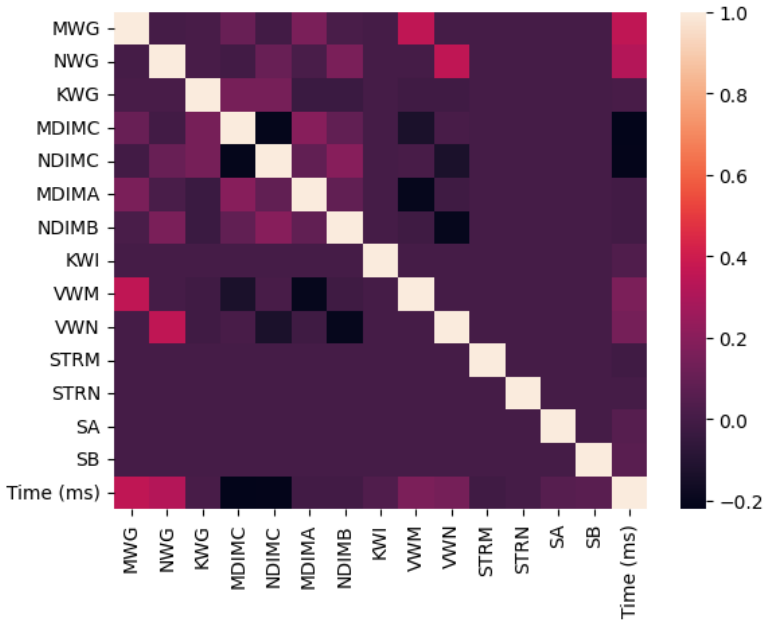

Figura 2: Matriz de correlación entre todos los parámetros y el tiempo de ejecución.

to a correlación con el tiempo de ejecución, con valores de correlación de $-0{,}221093$ y $-0{,}214592$, respectivamente. Esto indica que existe una relación negativa moderada entre estos dos parámetros y el tiempo de ejecución.

Los demás parámetros tienen valores de correlación más bajos en valor absoluto, lo que indica que su relación con el tiempo de ejecución es más débil. En resumen, los parámetros más relevantes en cuanto a correlación con el tiempo de ejecución son MWG, NWG, MDIMC y NDIMC.

Para visualizar mejor esta relación, se han creado gráficos (figura 3) que muestran el tiempo de ejecución en función del valor que toma cada uno de estos cuatro parámetros.

4.3. Análisis de la distribución

Como hemos visto anteriormente, el histograma de los logaritmos de las ejecuciones parece tener múltiples picos, podría sugerir que sus datos son una mezcla de varias distribuciones normales.

Una forma de modelar este tipo de datos es utilizar un modelo mixto,

Var	Correlation index
MWG	**0.351805**
NWG	**0.320455**
KWG	0.011229
MDIMC	**-0.221093**
NDIMC	**-0.214592**
MDIMA	-0.007035
NDIMB	-0.008706
KWI	0.032570
VWM	0.164271
VWN	0.144743
STRM	-0.012586
STRN	-0.000108
SA	0.051974
SB	0.063962
Time (ms)	*1.000000*

Tabla 2: Índice de correlación entre el tiempo de ejecución y cada uno de los parámetros de entrada. Destacados los valores más relevantes.

que supone que los datos se generan mediante una combinación de varias distribuciones subyacentes. En su caso, podría usar un modelo de mezcla gaussiana para representar sus datos como una mezcla de varias distribuciones normales.

Una vez que ajustado un modelo de mezcla gaussiana a los datos, podría usarse para varios propósitos, como:

- Clustering: El modelo puede utilizarse para agrupar los datos en diferentes grupos (clusters) en función de su similitud. Cada componente gaussiana del modelo representa un grupo diferente y los datos se asignan al grupo cuya componente gaussiana tenga la mayor probabilidad.
- Detección de valores atípicos: El modelo puede utilizarse para detectar valores atípicos en los datos. Los valores atípicos son aquellos puntos que tienen una baja probabilidad según el modelo ajustado.
- Estimación de densidad: El modelo puede utilizarse para estimar la función de densidad subyacente de los datos. La función de densidad se puede utilizar para generar nuevos datos sintéticos o para calcular probabilidades.

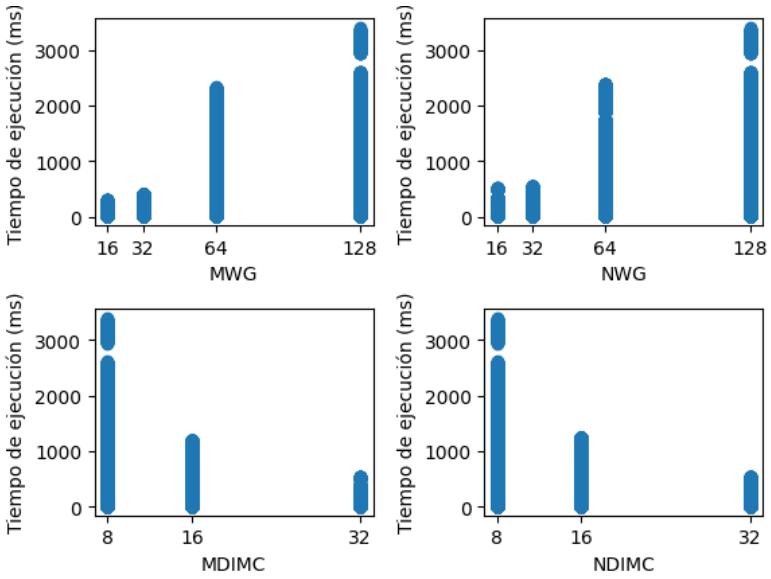

Figura 3: Relación entre el tiempo de ejecución y los parámetros MWG, NWG, MDIMC y NDIMC.

Aunque este tema está fuera del alcance del curso, se realizó una aproximación a la técnica cuyo resultado es el mostrado en la figura 4.

5. Resultados

Se analizaron los tiempos de ejecución de 4 ejecuciones independientes y se encontró una variabilidad significativa en los tiempos de ejecución para diferentes combinaciones de parámetros. Al aplicar el logaritmo natural a los valores de la media y generar un histograma, se observó que la distribución no es normal y que hay múltiples subpoblaciones en los datos. La prueba de bondad de ajuste de Kolmogorov-Smirnov confirmó que los datos no siguen una distribución normal.

Se realizó un análisis de correlación entre el tiempo de ejecución de un producto matriz-matriz y diferentes parámetros que afectan su rendimiento. Los parámetros más relevantes en cuanto a correlación con el tiempo de ejecución resultaron ser MWG y NWG, con una relación positiva moderada. Los parámetros MDIMC y NDIMC también resultaron relevantes, con una relación negativa moderada. Los demás parámetros obtuvieron una relación

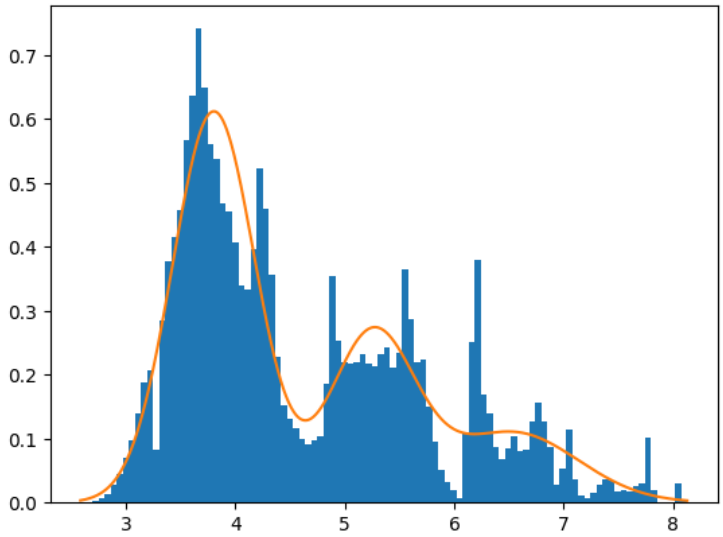

Figura 4: Histograma del logaritmo de los tiempos de ejecución y modelo de mezcla gaussiana ajustado con 3 componentes. El eje x representa el logaritmo del tiempo de ejecución en milisegundos, mientras que el eje y representa la densidad.

más débil con el tiempo de ejecución.

Se estudió la distribución de los datos. El histograma de los logaritmos de las ejecuciones sugería que los datos son una mezcla de varias distribuciones normales. Se generó un modelo de mezcla gaussiana puede ser utilizado para representar los datos y puede ser usado para clustering, detección de valores atípicos y estimación de densidad.

6. Conclusiones

En resumen, el análisis estadístico de los tiempos de ejecución y los parámetros que afectan su rendimiento ha permitido obtener información valiosa sobre la estructura de los datos y las relaciones entre las variables. Se ha encontrado una variabilidad significativa en los tiempos de ejecución para diferentes combinaciones de parámetros y se han identificado los parámetros más relevantes en cuanto a correlación con el tiempo de ejecución. Además, se ha utilizado un modelo de mezcla gaussiana para representar la distribución de los datos y se ha demostrado su utilidad para clustering, detección de

valores atípicos y estimación de densidad. Este análisis estadístico demuestra el interés y la importancia de utilizar técnicas estadísticas avanzadas para obtener información útil a partir de los datos.

Referencias

[1] R. Ballester-Ripoll, E. G. Paredes, and R. Pajarola. Sobol tensor trains for global sensitivity analysis, 2017.

[2] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, Sept. 2020.

[3] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[4] W. McKinney et al. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, volume 445, pages 51–56. Austin, TX, 2010.

[5] D. C. Montgomery. *Probabilidad y estadística aplicadas a la ingeniería*. McGraw-Hill/Interamericana, México [etc, 1996.

[6] C. Nugteren. CLTune: An Automatic OpenCL kernel tuning, 2017.

[7] 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*, pages 195–202, 2015.

[8] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[9] J. VanderPlas. *Python data science handbook : essential tools for working with data*. O'Reilly Media, Inc, Sebastopol, CA, 2016.

[10] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.