UNIVERSIDADE DE SÃO PAULO

Instituto de Ciências Matemáticas e de Computação

Ciência de dados aplicada em predições de desempenho para processamento paralelo em GPUs

Benício Ramos Magalhães

Trabalho de Conclusão de Curso - MBA em Ciência de Dados (CEMEAI)

ICMC USP SÃO CARLOS

UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

Ciência de dados aplicada em predições de desempenho para processamento paralelo em GPUs

Benicio Ramos Magalhães

USP - São Carlos

2020

BENICIO RAMOS MAGALHÃES

Ciência de dados aplicada em predições de desempenho para processamento paralelo em GPUs

Trabalho de conclusão de curso apresentado ao Centro de Ciências Matemáticas Aplicadas à Indústria do Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, como parte dos requisitos para conclusão do MBA em Ciência de Dados.

Área de concentração: Ciências de Dados

Orientador: Prof. Dr. Antonio Castelo Filho

USP - São Carlos

2020

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

M188c

Magalhães, Benicio Ramos
Ciência de dados aplicada em predições de desempenho para processamento paralelo em GPUs / Benicio Ramos Magalhães; orientador Antonio Castelo Filho. -- São Carlos, 2020.
100 p.

Trabalho de conclusão de curso (MBA em Ciência de Dados) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2020.

1. Trabalho de conclusão de curso. 2. MBA. 3. Ciência de dados. 4. Computação paralela. 5. Predição de desempenho. I. Filho, Antonio Castelo, orient. II. Título.

Bibliotecários responsáveis pela estrutura de catalogação da publicação de acordo com a AACR2:
Gláucia Maria Saia Cristianini - CRB - 8/4938
Juliana de Souza Moraes - CRB - 8/6176

ERRATA

FOLHA DE AVALIAÇÃO OU APROVAÇÃO

DEDICATÓRIA

A minha esposa pela compreensão, carinho e apoio incansável.

AGRADECIMENTOS

“Jamais considere seus estudos como uma obrigação, mas como uma oportunidade invejável para aprender a conhecer a influência libertadora da beleza do reino do espírito, para seu próprio prazer pessoal e para proveito da comunidade à qual seu futuro trabalho pertencer.”

- Albert Einsten

RESUMO

MAGALHÃES, B. R. **Ciência de dados aplicada em predições de desempenho para processamento paralelo em GPUs**. 2020. XX f. Trabalho de conclusão de curso (MBA em Ciência de Dados) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2020.

Este trabalho é um estudo de técnicas de predição de desempenho de processamento de computação paralela em GPUs utilizando ciência de dados. O objetivo é modelar o comportamento desses sistemas e estudar diversas técnicas existentes na área de ciência de dados e com isso predizer seu comportamento com relação à métricas não conhecidas. Inicialmente, realizamos uma pesquisa bibliográfica acerca dos principais trabalhos relacionados à previsão de desempenho de programas paralelos e, em seguida, obtivemos uma base de dados de métricas de desempenho de processamento paralelo de GPUs. A partir dessa base, elaboramos um modelo representativo e, por fim, realizamos predições de desempenho com relação às métricas coletadas.

Palavras-chave: Modelagem. Avaliação de desempenho. Computação paralela. Ciência de dados. Análise de predição de dados.

ABSTRACT

MAGALHAES, B. R. **Data science applied in performance predictions for parallel processing in GPUs**. 2020. XX f. Completion of course work (MBA em Ciência de Dados) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2020.

This work is a study of performance prediction techniques of parallel computing processing in GPUs using data science. The objective is to model the behavior of these systems and to study several existing techniques in the area of data science and thereby predict their behavior in relation to unknown metrics. Initially, we carried out a bibliographic search about the main works related to the performance prediction of parallel programs and, then, we obtained a database of performance metrics of parallel processing of GPUs. From this base, we elaborate a representative model and, finally, we make performance predictions regarding the collected metrics.

Keywords: Modeling. Performance evaluation. Parallel computing. Data science. Data prediction analysis.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AET	–	Application Execution Time
AI	–	Artificial Intelligence
ALU	–	Arithmetic-Logic Unit
BSP	–	Bulk Synchronous Parallel Model
CPU	–	Central Processing Unit
CUDA	–	Compute Unified Device Architecture
CRISP-DM	–	Cross Industry Standard Process for Data Mining
DL	–	Deep Learning
FPU	–	Floating Point Unit
GPU	–	Graphics Processing Unit
GDDR	–	Graphics Double Data Rate
HPC	–	High Performance Computing
IQR	–	Interquartile Range
K-NN	–	K-Nearest Neighbors
LR	–	Linear Regression
LD	–	Load
MAE	–	Mean Absolute Error
ML	–	Machine Learning
MPI	–	Message Passing Interface
NNGE	–	Non-Nested Generalized Exemplars
OpenCL	–	Open Computing Language
PAS2P	–	Parallel Application Signature for Performance Prediction
PET	–	Predicted Execution Time
PETE	–	Prediction Execution Time Error
RAE	–	Relative Absolute Error
RF	–	Random Forest
RMSE	–	Root Mean Squared Error
RRSE	–	Root Relative Squared Error
SCC	–	Spearman Correlation Coefficient
SET	–	Signature Execution Time
SFU	–	Special Function Units

SM	–	Streaming Multiprocessors
SP	–	Streaming Processors
ST	–	Store
SVM	–	Support Vector Machines
TPC	–	Texture Processing Clusters
UCI	–	University of California Irvine

LISTA DE SÍMBOLOS

SUMÁRIO

1 INTRODUÇÃO

Em um mundo tecnológico e moderno, onde os sistemas computacionais oferecem diversos benefícios à sociedade, existe uma necessidade cada vez maior de trabalhar com aplicações de alto desempenho e isso tornou-se viável por meio da utilização de sistemas distribuídos. “Um sistema distribuído é um conjunto de computadores independentes que se apresenta a seus usuários como um sistema único e coerente.” (TANENBAUM, STEEN, 2007, p.1). Com isso os desenvolvedores começaram a procurar meios para escrever aplicações distribuídas com alta eficiência e isso resultou na utilização de novas tecnologias para processamento massivo paralelo, como as GPUs (*Graphics Processing Unit*).

“Computação paralela é mais que uma estratégia para atingir um alto desempenho, ela é uma visão de como a computação pode ser escalonada para ter um poder computacional praticamente ilimitado.” (DONGARRA et al, 2003, p.3). Esse aumento de poder computacional com baixo custo tem viabilizado a resolução de problemas complexos aplicada em diversas áreas de conhecimento.

Para aferir e garantir que uma aplicação atenda aos requisitos não-funcionais de alto desempenho, exigidos principalmente por sistemas de missão crítica, existem diversas técnicas de avaliação de desempenho. Jain (1991) define uma avaliação de desempenho como uma arte, portanto, assim como uma obra de arte, toda a avaliação requer um conhecimento íntimo do que está sendo modelado e uma seleção cuidadosa da metodologia, carga de trabalho e ferramentas.

Outra área de interesse neste trabalho, envolve o estudo de modelos preditivos para estimar o comportamento dos sistemas com relação ao seu desempenho. De acordo com SAS (2020), os modelos preditivos utilizam resultados conhecidos para desenvolver (ou treinar) um modelo que pode ser usado para prever valores para dados diferentes ou novos.

Neste trabalho, vamos nos basear nas metodologias previstas na área de ciência de dados para análise de predição desempenho do resultado do tempo de processamento de um produto matricial executado em diversas configurações aplicadas em uma GPU, obtendo assim modelos variados, realizando simulações e comparando os resultados previstos com os dados reais medidos.

1.1 Objetivos

Este trabalho tem como objetivo principal estudar técnicas e metodologias de predição existentes em ciência de dados aplicados em avaliações de desempenho de sistemas computacionais. Para isso, utilizamos dados com tempos de processamento em GPUs e propomos algumas técnicas de predição de desempenho usando algoritmos de aprendizado de máquina e estatística.

Como objetivos específicos, montamos um modelo representativo para explicar as características principais de um sistema computacional com relação as suas funções e seus parâmetros. Também aplicamos simulações no modelo para predizer seu comportamento com relação à métricas não conhecidas.

1.2 Motivação

A ciência de dados é uma área que se tem mostrado promissora para resolver problemas reais de negócios, com o uso de métodos científicos e técnicas avançadas para captura, tratamento de dados, aprendizado de máquina, redes neurais e inteligência artificial.

De maneira geral, a principal motivação para avaliar o desempenho de programas computacionais é melhorar a eficiência dos algoritmos para que seja viável resolvermos problemas de grande complexidade de forma mais rápida e barata.

1.3 Justificativa

O intuito é aplicar os conhecimentos dessa área numa base de dados obtida através de testes de desempenho em GPUs para que seja possível modelar e prever o comportamento de programas sobre condições que ainda não foram testadas.

1.4 Metodologia

Primeiramente foi feito um levantamento bibliográfico acerca das principais técnicas de predição de dados aplicados a sistemas de programas paralelos. Em seguida, foi obtida uma base de dados de métricas de desempenho de um programa de processamento matricial em GPUs, com resultados obtidos através de combinações nos parâmetros de entrada. Na sequência, criamos alguns modelos representativos com as principais características do sistema e a partir desses modelos elaborados, realizamos predições de desempenho utilizando simulações e técnicas de ciência de dados.

1.5 Organização do trabalho

O próximo capítulo apresentará alguns trabalhos relacionados à predição de desempenho de programas paralelos, seguida das técnicas mais utilizadas em ciência de dados. No capítulo 3, abordaremos de forma teórica o que é computação paralela focado na utilização de GPUs. O capítulo 4 apresenta quais são e como são aplicados os modelos de predição com aprendizado de máquina. Já o capítulo 5 é a implementação dos algoritmos, geração e análise dos resultados, verificando a diferença entre os modelos, discutindo sua eficácia e comparando as predições com os dados reais medidos em testes. Finalmente, algumas conclusões decorrentes deste estudo são apresentadas no capítulo 6.

2 REVISÃO BIBLIOGRÁFICA

Fizemos um levantamento bibliográfico onde destacamos alguns trabalhos relevantes para o tema de pesquisa deste trabalho. As principais teses pesquisadas foram realizadas por González (2018) que apresenta uma predição de desempenho de aplicações executadas em diversas GPUs usando tanto um modelo analítico quanto técnicas de aprendizado de máquina. Destacamos também o trabalho realizado em Baldini et al. (2014) que mostra que é possível predizer qual GPU oferece o melhor desempenho na hora de realizar a portabilidade de um sistema de processamento paralelo executado originalmente em CPUs, utilizando apenas técnicas de aprendizado de máquina.

Considerando que temos interesse também na predição de desempenho de aplicações paralelas como um todo, destacamos aqui dois estudos que não necessariamente estão relacionados com GPUs, mas que geraram interesse devido ao fato de apresentar uma metodologia mais genérica para avaliação de sistemas desta natureza, como a PAS2P proposta por Wong, Rexachs e Luque (2015) e CRISP-DM proposta em Chapman et al [1]* (2000 *apud* QAZDAR et al, 2019, p.3580).

Na proposta do trabalho desenvolvido por González (2018) ele realiza a medição de algumas aplicações em diferentes GPUs e utiliza o método analítico através de um modelo BSP (*Bulk Synchronous Parallel Model)* e depois usa três abordagens de aprendizado de máquina (Regressão Linear, Máquinas de Vetor de Suporte e Florestas Aleatórias). Mesmo sabendo que o modelo analítico fornece previsões melhores, o intuito é mostrar que as técnicas de aprendizado de máquina podem oferecer previsões aceitáveis para todas as aplicações mesmo sem a necessidade de uma análise e um conhecimento profundo do sistema, nas quais são exigidas na modelagem analítica.

Em Baldini et al. (2014) eles lidam com as questões de esforço de portabilidade dos códigos que utilizam processamento paralelo. Realizar a portabilidade de um sistema de processamento paralelo projetado originalmente para CPUs (*Central Processing Unit*) pode gerar um esforço grande, sendo que muitas vezes é necessário reescrever todo o código para uma linguagem específica da GPU. Eles mostram que com técnicas de aprendizado de máquina é possível criar modelos que conseguem predizer o desempenho de forma precisa sem requerer

[1] *Chapman, P., Clinton, J., Kerber, R., Khabza, T., Reinartz, T., Shearer, C., & Wilrth, R. (2000). CRISP-DM 1.0 step-by-step data mining guide. The CRISP-DM consortium.

de uma análise estática do código ou mesmo a criação de um modelo analítico, conseguindo até mesmo demonstrar quais são os melhores dispositivos para o sistema.

Em Wong, Rexachs e Luque (2015) é proposta uma metodologia chamada de PAS2P (*Parallel Application Signature for Performance Prediction*) onde dados de desempenho do sistema são coletados e caracterizados por suas fases de execução (comportamento de repetição do algoritmo) e pesos (valor associado às métricas coletadas). Com essa informação é realizada uma predição do tempo de execução da aplicação paralela e depois validada através de resultados experimentais.

Por fim, a metodologia proposta por Chapman et al* (2000 *apud* QAZDAR et al, 2019, p.3580) é chamada de CRISP-DM (*Cross Industry Standard Process for Data Mining*). A metodologia consiste em alguns passos que vai desde o entendimento do negócio, passando pelo entendimento e preparação dos dados, a modelagem em si, que envolve implementações de aprendizado de máquina e a validação do modelo. Uma vez que o modelo está criado, testado e validado, a última etapa consiste na implantação, que pode ser um relatório com os resultados obtidos.

2.1 Trabalho desenvolvido por González, M. T. A.

O trabalho desenvolvido por González (2018) apresenta uma análise de desempenho de aplicações executadas em diversas GPUs dividida em duas etapas, a primeira utilizando um modelo analítico e a segunda utilizando métodos de aprendizado de máquina.

Foram utilizadas cinco aplicações algébricas (multiplicação e adição matricial, adição vetorial, produto escalar e sublista contígua de maior soma) apresentadas na tabela 2.1 e mais seis aplicações utilizando Rodinia (*Back Propagation, Gaussian Elimination, Heart Wall, Hot Spot, LU Decomposition, Needleman-Wunsch*) que são *benchmarks* do tipo *open-source* que contém códigos CUDA (*Compute Unified Device Architecture* ou arquitetura de dispositivo de computação unificada). Esses últimos apresentados na tabela 2.2. E na tabela 2.3 são apresentados nove diferentes GPUs utilizadas para nos experimentos:

Tabela 2.1 – Aplicações de álgebra linear utilizado nos experimentos de González (2018)

Application	Param	Kernel	dimGrid	dimBlock	Shared Mem
Matrix Mul	1	MMGU MMGC MMSU MMSC	(GS, GS, 1)	(BS, BS, 1)	0 0 $(BS^2 \times 2 \times 4B)$
Matrix Add	1	MAU MAC	(GS, GS, 1)	(BS, BS, 1)	0
Vector Add	1	VAdd	(GS, 1, 1)	(BS, 1, 1)	0
Dot Product	1	dotP	(GS, 1, 1)	(BS, 1, 1)	$(BS \times 4B)$
Max. Sub Array	1	MSA	(48, 1, 1)	(128, 1, 1)	$(4096 \times 4B)$

Fonte: González (2018, p.15).

Tabela 2.2 – Aplicações Rodinia utilizado nos experimentos de González (2018)

Application	Berkeley Dwarf	Domain	Param.	Kernels	Samples
Back Propagation (BCK)	Unstructured Grid	Pattern Recognition	1 - [57]	layerforward adjust-weights	57
Gaussian Elimination (GAU)	Dense Linear Algebra	Linear Algebra	1 - [32]	Fan1 Fan2	34800
Heart Wall (HWL)	Structured Grid	Medical Imaging	1 - [84]	heartWall	5270
Hot Spot (HOT)	Structured Grid	Physics Simulation	2 - [5,4]	calculate-temp	396288
LU Decomposition (LUD)	Dense Linear Algebra	Linear Algebra	1 - [32]	diagonal perimeter internal	8448 8416 8416
Needleman-Wunsch (NDL)	Dynamic Programming	Bioinformatics	2 - [16,10]	needle-1 needle-2	21760 21600

Fonte: González (2018, p.18).

Tabela 2.3 – Especificações dos Hardwares das GPUs utilizados nos experimentos de González (2018)

Model	C.C.	Memory	Bus	Bandwidth	L2	Cores/SM	Clock
GTX-680	3.0	2 GB	256-bit	192.2 GB/s	0.5 M	1536/8	1058 Mhz
Tesla-K40	3.5	12 GB	384-bit	276.5 GB/s	1.5 MB	2880/15	745 Mhz
Tesla-K20	3.5	4 GB	320-bit	200 GB/s	1 MB	2496/13	706 MHz
Titan	3.5	6 GB	384-bit	288.4 GB/s	1.5 MB	2688/14	876 Mhz
Quadro K5200	3.5	8 GB	256-bit	192.2 Gb/s	1 MB	2304/12	771 Mhz
Titan X	5.2	12 GB	384-bit	336.5 GB/s	3 MB	3072/24	1076 Mhz
GTX-970	5.2	4 GB	256-bit	224.3 GB/s	1.75 MB	1664/13	1279 Mhz
GTX-980	5.2	4 GB	256-bit	224.3 GB/s	2 MB	2048/16	1216 Mhz
Pascal-P100	6.0 GB	16 GB	4096-bit	732 GB/s	4 MB	3584/56	1328 Mhz

Fonte: González (2018, p.19).

Para o modelo analítico ele se baseia em BSP (*Bulk Synchronous Parallel Model*) que oferece uma simples abstração das arquiteturas paralelas, dividindo seus resultados em três grandes fases: processamento, comunicação e sincronização dos dados. O modelo analítico foi implementado conforme apresentado na equação 2.1, onde o tempo de execução é dividido apenas entre o custo computacional e custo de comunicação. Maiores detalhes sobre a aplicação deste modelo podem ser encontrados em (GONZÁLEZ, 2018, p.32):

$$T_K = \frac{l\,.(comp+comm_{GM}+comm_{SM})}{R\,.\,P\,.\,\lambda} \quad (2.1)$$

Fonte: González (2018, p.32).

Onde:

T_K: tempo de execução aproximado do kernel em função do número de threads.

l: número de threads

$comp$: custo computacional

$comm_{GM}$: custo de comunicação com a memória global

$comm_{SM}$: custo de comunicação com a memória compartilhada

R: taxa de clock

P: número de núcleos da GPU

λ: parâmetro estimado como taxa entre o tempo de execução previsto do kernel e o tempo de execução medido

Para avaliar a acurácia do modelo, foram realizados experimentos, onde foram comparadas as predições do modelo com os resultados medidos. A medida final da acurácia foi dada pela taxa T_K/T_m ,sendo T_K (tempo de execução do modelo) e T_m (tempo de execução medido).

Para o aprendizado de máquina ele utiliza três métodos: LR (*linear regression* ou regressão linear), SVM (*Support Vector Machines*) e RF (*random forest* ou florestas aleatórias), com duas diferentes implementações: a primeira usando todas as variáveis preditoras existentes e a segunda utilizando técnicas para otimização do uso de variáveis preditoras, realizando a extração de algumas delas. O erro nas predições foi calculado utilizando o MAPE (*Mean Absolute Percentage Error* ou erro percentual absoluto médio) apresentado na equação 2.2:

$$MAPE = \frac{100}{n}\sum_{t=1}^{n}\left|\frac{T_m - T_k}{T_m}\right| \quad (2.2)$$

Fonte: González (2018, p.47).

Onde:

T_K: tempo predito

T_m: tempo medido

n: número de pontos ajustados

Para avaliar os modelos de aprendizado de máquina foram utilizados os procedimentos de validação cruzada ou *leave-one-out cross validation*, selecionando parte dos dados como teste (tanto no contexto de GPUs quanto *Kernels*) e o restante para treinar os modelos.

A implementação com extração de variáveis preditivas foi realizada usando técnicas de correlação e clusterização de dados, no caso, o SCC (*Spearman Correlation Coefficient* ou coeficiente de correlação de Spearman). O SCC foi aplicado para todas as variáveis contra o tempo de execução dos *kernels* e aplicado um limite de 0.75 neste coeficiente, realizando assim a redução do número de variáveis no modelo.

Os resultados da primeira implementação (com todas as variáveis preditivas) foram comparadas com o modelo analítico e são apresentadas a seguir nas tabelas 2.4 e 2.5:

Tabela 2.4 – MAPE das predições em % com as aplicações de vetoriais e matriciais de González (2018)

Apps	MAPE ML Techniques			
	AM	LR	SVM	RF
MMGU	4.41±4.97	16.47±13.40	20.48±14.88	10.65±10.21
MMGC	7.35±3.49	15.53±11.28	13.93±7.49	10.02±7.44
MMSU	7.95±5.31	5.33±1.93	11.79±4.25	4.90±1.99
MMSC	8.31±6.44	14.83±10.22	17.04±8.21	8.41±2.87
MAU	9.71±3.09	69.51±51.77	14.76±9.62	36.76±22.23
MAC	11.00±5.27	12.65±2.77	17.67±8.52	10.83±3.85
dotP	6.89±5.56	3.30±1.42	15.37±8.31	4.67±1.35
vAdd	9.92±7.71	18.98±16.56	14.68±8.15	8.64±4.56
MSA	28.02±13.05	3.02±1.62	10.91±5.02	5.67±4.11

Fonte: González (2018, p.52).

Tabela 2.5 – MAPE das predições em % com as aplicações Rodinia CUDA kernels de González (2018)

Kernels/Techn.	AM	LR	SVM	RF
BCK-K1	3.92±1.00	19.89±15.57	31.00±49.90	24.49±15.90
BCK-K2	4.86±3.33	86.42±158.74	140.21±283.86	97.49±188.53
GAU-K1	54.09±5.92	65.24±116.70	14.82±3.13	35.31±53.29
GAU-K2	63.72±5.12	26.62±12.38	18.52±11.73	18.60±7.48
HTW	3.71±2.23	100.79±192.76	41.63±38.04	26.29±25.13
HOT	5.53±2.14	167.16±293.56	57.61±83.96	116.92±183.91
LUD-K1	-	27.61±41.32	10.46±8.37	27.45±41.17
LUD-K2	-	57.70±79.18	42.50±54.74	48.16±70.49
LUD-K3	-	41.22±22.08	25.94±24.01	42.13±37.12
NDL-K1	-	16.09±5.27	15.54±10.09	14.03±3.81
NDL-K2	-	15.01±5.11	10.56±4.94	13.20±3.25

Fonte: González (2018, p.53).

Pode-se observar que o modelo analítico apresenta melhores resultados e que a previsão com o aprendizado de máquina não é muito boa em alguns dos algoritmos, e essas são listadas por diversas razões, como, por exemplo, o pouco número de amostras disponíveis. Mesmo assim, alguns algoritmos apresentaram uma boa predição, como observado no caso do RF para as aplicações vetoriais e matriciais. A análise das previsões é bem extensa e não é o objetivo deste trabalho apresentar o detalhamento de todas elas, mas ela pode ser encontrada em (GONZÁLEZ, 2018, p.52-59).

De maneira geral, ele consegue constatar que os modelos de aprendizado de máquina foram capazes de trazer resultados satisfatórios uma vez que apresentam uma abordagem mais generalizada para diferentes tipos de aplicativos e GPUs, sem precisar de conhecimento prévio a respeito dos mesmos, no qual é exigido na construção do modelo analítico.

2.2 Trabalho desenvolvido por Baldini, I., Fink, S. J., e Altman, E.

A metodologia aplicada em Baldini et al. (2014) endereça o problema de identificar o benefício de migrarmos uma aplicação originalmente desenvolvida para ser executada em CPUs para um ambiente com GPUs, sem o esforço de realizar a portabilidade do mesmo para este novo dispositivo, além disso, é possível demonstrar também que alguns modelos preditivos são até mesmo capazes de escolher qual o melhor dispositivo para aquela aplicação.

A figura 2.1 mostra um diagrama em alto-nível da abordagem realizada para prever o desempenho das aplicações em GPUs:

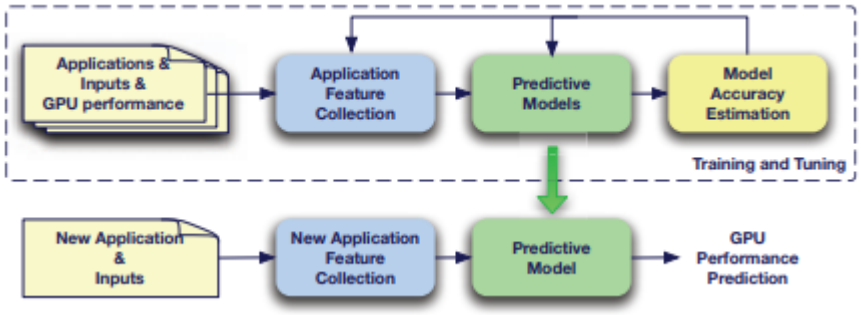

Figura 2.1 - Visão geral da metodologia para predição de desempenho de aplicações em GPUs baseado em duas fases: treino para calibração do modelo e predição, que aplica o modelo em uma nova aplicação.

Fonte: Baldini et al. (2014, p.255).

Foi realizada a escolha das variáveis preditivas pensando nas métricas mais relevantes para representar as características de uma aplicação paralela, sendo assim, as aplicações foram devidamente instrumentadas e as atenções restritas a estas variáveis. A tabela 2.6 apresenta as variáveis preditivas selecionadas em (BALDINI et al., 2014, p.256):

Tabela 2.6 – Variáveis preditivas utilizadas nas modelagens de aprendizado de máquina em Baldini et al. (2014)

Category	Feature	Mnemonic
Computation	Arithmetic and logic instructions	ALU
	SIMD-based instructions	SIMD
Memory	Memory loads	LD
	Memory stores	ST
	Memory fences	FENCE
Control flow	Conditional and unconditional branches	BR
OpenMP	Speedup of 12 threads over sequential execution	OMP
Aggregate	Total number of instructions	TOTAL
	Ratio of computation over memory	ALU-MEM
	Ratio of computation over GPU communication	ALU-COMM

Fonte: Baldini et al. (2014, p.256).

Todos os experimentos foram executados em um servidor com dois processadores de seis núcleos cada (totalizando 12 núcleos) e também equipados com duas placas GPUs. Eles utilizaram 18 *benchmarks* vindos da suíte de aplicações Parboil 2.0 e Rodinia 2.2 para a construção da sua base de dados e utilizaram procedimentos de validação cruzada ou *leave-one-out cross validation*, para selecionar parte dos dados como teste e o restante como treinamento para o desenvolvimento dos modelos.

O primeiro questionamento é se existiria uma forte correlação entre o ganho de desempenho medido na execução com CPUs (comparação utilizando 12 núcleos) contra o ganho medido nas GPUs. Chegaram à conclusão que não existia uma forte correlação e ainda que boa parte das execuções tinham desempenho pior em um ambiente com GPUs. Concluíram

que os benchmarks utilizados incluem uma boa quantidade de execuções nas quais uma aceleração vinda de GPUs não são efetivas, porém, outras poderiam apresentar benefícios.

Sendo assim, a segunda etapa é verificar se o ganho de desempenho da execução de um algoritmo em paralelo com relação à sua versão sequencial (*speedup*) justifica o esforço para realizar sua portabilidade. Neste caso, eles propõem uma solução formulando um problema de classificação binária, onde a classe identifica se a aplicação consegue um ganho em GPUs maior que um limite (*threshold*) pré-estabelecido. Os classificadores de aprendizado de máquina escolhidos foram o NNGE (*Non-Nested Generalized Exemplars*) e SVM (*Support Vector Machines*), sendo que foi obtida uma acurácia de aproximadamente 80% e o modelo treinado utilizando SVM teve uma acurácia um pouco melhor. Com isso foi possível criar um primeiro modelo de aprendizado de máquina que consegue verificar se o algoritmo é efetivo para uma aceleração vinda de GPUs.

Com este *sub set* de dados em mãos, o próximo passo é predizer qual o melhor dispositivo para realizar a portabilidade daquele algoritmo. Para isso foi utilizado o classificador NNGE com três classes, sendo elas correspondentes à uma CPU e duas GPUs distintas. Para cada execução do benchmark na base de dados, é feita uma classificação que corresponde ao dispositivo que obteve o melhor desempenho. Neste caso, o melhor desempenho foi baseado nas medições das métricas de operações de processamento e memória. Por fim, dadas as devidas considerações, obtiveram uma acurácia em torno de 91% para o preditor. A figura 2.2 apresenta os resultados obtidos para cada benchmark, sendo que cada *benchmark* teve três execuções (três grupos de barras apresentadas), o hardware com melhor desempenho em cada uma delas está pintada de azul escuro e as predições com erro foram apresentadas em vermelho escuro.

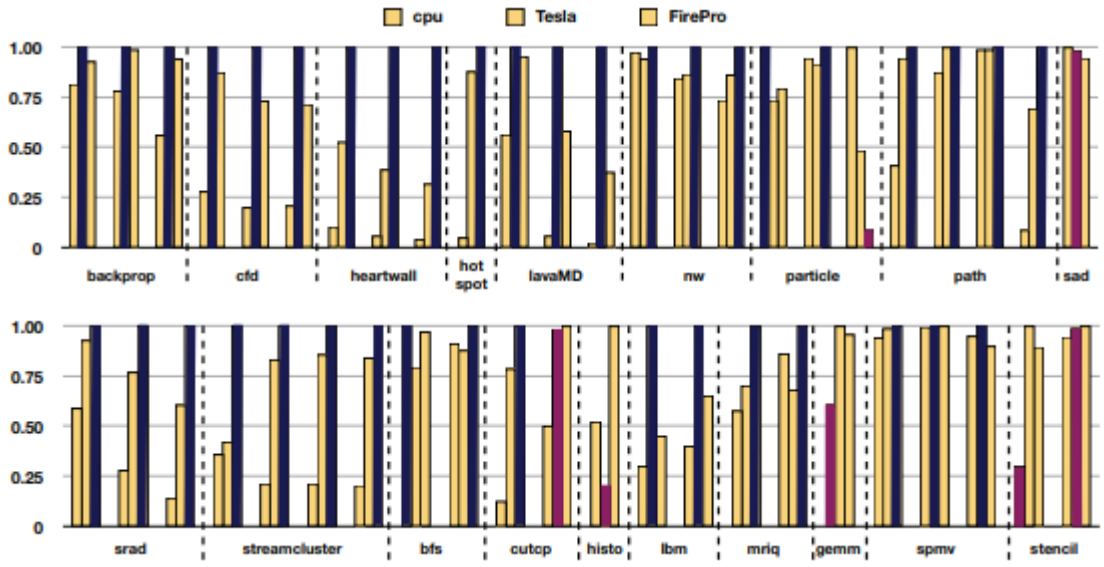

Figura 2.2 – Resultados da classificação de desempenho com três classes de dispositivos (CPU, Tesla GPU e FirePro GPU)

Fonte: Baldini et al. (2014, p.260).

2.3 PAS2P

PAS2P (*Parallel Application Signature for Performance Prediction*) proposta por Wong, Rexachs e Luque (2015) é um método de predição de desempenho empenhado em descrever uma aplicação baseada no seu comportamento. O PAS2P consiste basicamente em dois estágios:

1. Análise da aplicação e geração de sua assinatura;
2. Predição de desempenho.

A figura 2.3 apresenta uma visão geral da metodologia:

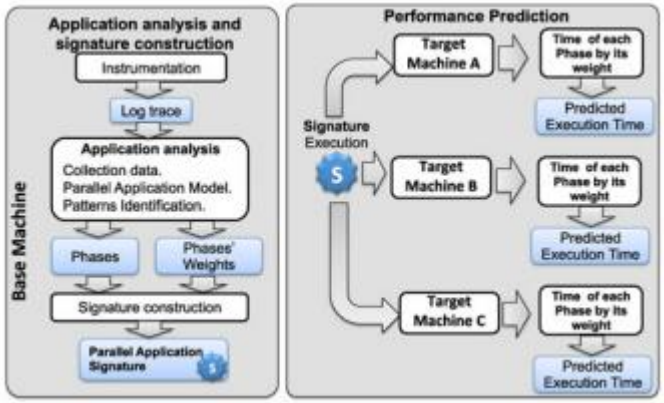

Figura 2.3 - Visão geral da metodologia PAS2P

Fonte: Wong, Rexachs, Luque (2015, p.2010).

A primeira etapa é o processo de gerar a assinatura da aplicação, que consiste em instrumentar o programa e executá-lo em uma máquina base gerando um log de rastreamento. Para modelar a aplicação, são coletadas métricas de tempo de execução dividindo as etapas de processamento em fases. Essas fases são agrupadas e é estipulado um peso para cada uma, que é definido pelo número de vezes em que elas ocorrem (repetições de um mesmo tipo de comunicação). A partir disso, as fases são marcadas (*checkpoints*) e finalmente são geradas as assinaturas, que nada mais são do que marcações do código instrumentado que sabem exatamente onde começa e termina cada fase.

A figura 2.4 exemplifica os passos para extração e determinação das fases do algoritmo:

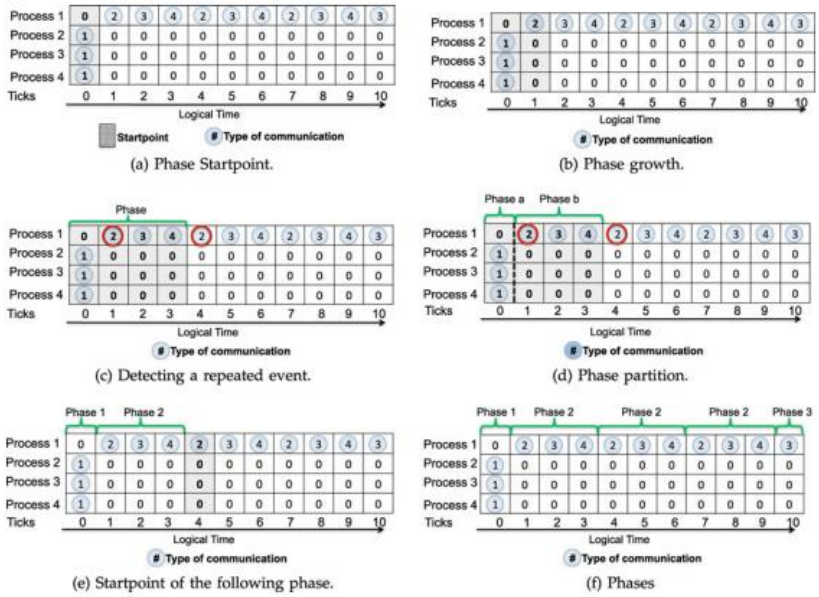

Figura 2.4 - Exemplificação do padrão do algoritmo de identificação das fases.

Fonte: Wong, Rexachs, Luque (2015, p.2013).

A figura 2.5 exemplifica uma tabela de fases que serve como base para construção da assinatura:

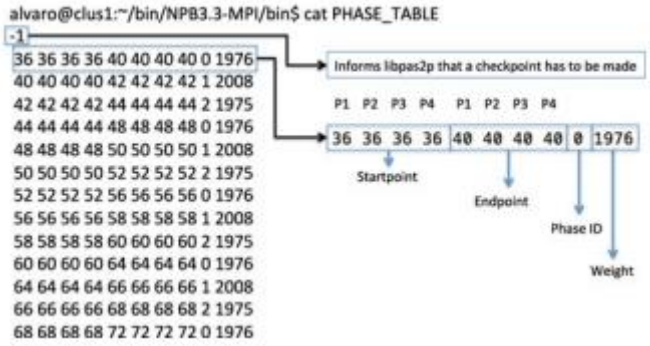

Figura 2.5 - Tabela de fases para construção da assinatura.

Fonte: Wong, Rexachs, Luque (2015, p.2014).

A segunda etapa consiste na predição de desempenho, que ocorre a partir da assinatura obtida na etapa um. Para predizer o tempo de execução PET (*Predicted Execution Time*) da aplicação é utilizada uma equação que consiste na multiplicação do tempo de execução de cada fase pelo seu peso.

Com as informações referentes à assinatura, executa-se um novo teste nas máquinas onde desejamos predizer o comportamento da aplicação. Essa execução mede o tempo de cada fase mapeada na assinatura e no final faz o cálculo da predição do tempo de execução. A principal vantagem é que a execução da assinatura da aplicação costuma ser muito mais eficiente do que a execução da aplicação em si, pois ele considera as estruturas de repetição como fases similares e os tempos preditos validados experimentalmente apresentam erros máximos de apenas 3%. Vale observar que nesta metodologia, caso a aplicação avaliada não apresente essa característica de repetição, o tempo de execução das fases será bem similar ao tempo de execução real da aplicação.

A tabela 2.7 apresenta um exemplo com os erros percentuais obtidos do tempo predito de algumas aplicações em relação ao tempo medido utilizando essa metodologia. Vale observar que o tempo de execução da assinatura SET (*Signature Execution Time*) é bem inferior ao tempo de execução da aplicação AET (*Application Execution Time*) e também que percentual de erro do tempo predito PETE (*Prediction Execution Time Error*) é bem pequeno, chegando ao máximo de 3% de variação.

Tabela 2.7 - Exemplos de resultados de predição de desempenho utilizando a metodologia PAS2P

Appl.	Cores	SET (Sec)	SET versus AET(%)	PET (Sec)	PETE(%)	AET (Sec)
CG-64	32	8.42	0.29	2793.42	1.90	2847.42
	64	4.87	0.32	1504.66	0.48	1511.91
BT-64	32	13.47	0.80	1652.65	0.9	1667.64
	64	10.19	0.77	1302.76	0.55	1309.91
SP-64	32	2.04	0.24	808.76	1.28	819.17
	64	2.08	0.51	388.367	3.05	400.55
SMG2k	32	16.75	2.63	633.23	0.38	635.61
64	64	8.37	10.15	162.87	2.32	166.74
Sweep	16	4.32	0.17	2494.36	0.06	2492.74
3d-32	32	3.01	0.22	1328.04	0.40	1322.62
POP-64	32	22.79	1.41	1608.85	0.17	1611.59
	64	18.36	1.79	1016.01	0.61	1022.28

SET: Signature Execution Time, SET versus AET: 100(SET/AET).
PET: Predicted Execution Time, AET: Application Execution Time.
PETE: Prediction Execution Time Error.

Fonte: Wong, Rexachs, Luque (2015, p.2016).

2.4 CRISP-DM

Mesmo não estando diretamente relacionada ao tema de predição de desempenho de programas paralelos, a metodologia apresentada por Chapman et al *apud* Qazdar et al chamou a atenção por conter um processo genérico para criação de modelos e predição de resultados.

A figura 2.6 apresenta uma visão geral da metodologia CRISP-DM*:*

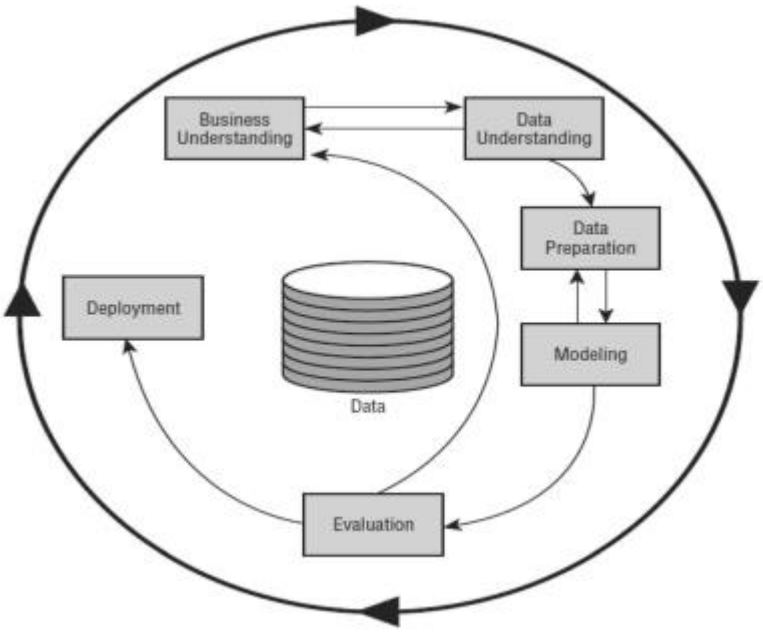

Figura 2.6 - Visão geral da metodologia CRISP-DM

Fonte: Qazdar et al. (2019, p.3581).

Para aplicarmos a metodologia CRISP-DM (*Cross Industry Standard Process for Data Mining*) são definidos os seguintes passos (QAZDAR et al, 2019, p.3580):

1. Entendimento do negócio em todos os seus aspectos, como por exemplo, os objetivos do projeto, os requisitos, regras aplicáveis, campo de aplicação, etc.;
2. Entendimento dos dados focado nas técnicas adequadas de coleta, descrição, exploração e manipulação;
3. Preparação dos dados. É a fase que cobre todas as atividades necessárias para o tratamento dos dados, obtendo assim uma base pronta para ser manipulada na fase de modelagem;
4. Modelagem. É a implementação de diferentes técnicas de aprendizado de máquina (regressão, classificação, clusterização, recomendação). A escolha desses algoritmos depende da necessidade do projeto, da base de dados e dos resultados almejados;

5. Avaliação do modelo. Aplicam-se técnicas de medidas de erro, como por exemplo, erro absoluto médio MAE (*Mean Absolute Error*), erro quadrático médio RMSE (*Root Mean Squared Error*), erro absoluto relativo RAE (*Relative Absolute Error*), erro quadrático relativo RRSE (*Root Relative Squared Error*), acurácia, entre outros. A escolha da avaliação do modelo está diretamente relacionada aos requisitos do projeto, o algoritmo usado e aos resultados desejados;
6. Entrega de resultados. Uma vez com o modelo criado, testado e avaliado, a entrega pode ser um relatório ou uma implementação do processo.

2.5 Considerações Finais

As fontes de pesquisa são de suma importância para a concepção de um novo trabalho. A ciência ganha cada vez mais credibilidade não só pelos esforços individuais de cientistas e pesquisadores, mas principalmente pelo fato de que a comunidade científica tem como ideal compartilhar conhecimento, ajudando a amadurecer idéias, firmar conceitos e principalmente evoluir. Reconhecendo a importância de estar inserido nesse contexto, seguimos para o próximo capítulo com a intenção de contribuir um pouco mais e compartilhar o conhecimento com todos aqueles que o anseiam.

3 COMPUTAÇÃO PARALELA COM GPUS

Computação paralela nada mais é do que escalar um processamento computacional por meio da utilização de diversas unidades de processamento trabalhando de forma simultânea com o objetivo de executar uma tarefa em comum. A técnica de paralelismo já existe há muitos anos e de acordo com Dongarra et. al (2003), ela é "uma visão atraente de como a computação pode ser escalonada de um único processador para um poder de computação virtualmente ilimitado".

Nos dias de hoje, este poder computacional tem sido cada vez melhor aproveitado, graças ao surgimento de aceleradores de hardware, como é o caso das GPUs (*Graphics Processing Unit)* ou unidade de processamento gráfico. Elas são basicamente processadores especializados em operações gráficas, sendo assim, foram projetadas de maneira específica para lidar com grandes conjuntos de dados de forma paralela, tornando-se mais eficientes do que as CPUs convencionais.

3.1 Introdução

A computação paralela traz uma série de desafios como, por exemplo, uma maior complexidade no desenvolvimento de softwares, problemas relacionados a comunicação e sincronização dos dados, desafios arquiteturais de memória compartilhada, paralelismo em CPUs, soluções de clusterização, entre outros. Assim sendo, todas as arquiteturas paralelas representam um peso entre o custo, complexidade, oportunidade e desempenho, porém, ela torna viável a resolução de problemas que anteriormente não conseguíamos resolver devido as limitações causadas pelo elevado tempo de processamento sequencial das informações.

A computação paralela faz uso de múltiplos núcleos de processamento, dividindo um problema em partes independentes, onde cada elemento de processamento consegue executar um algoritmo de forma simultânea. Dada a premissa de que podemos tirar um melhor proveito da computação paralela a partir de múltiplos núcleos é adequado explorarmos os conceitos básicos de funcionamento das GPUs.

GPUs inicialmente foram projetadas apenas para funções gráficas, porém, seu potencial em acelerar o processamento de dados foi logo percebido em cenários modernos de HPC (*High*

Performance Computing). O HPC ou computação de alto desempenho é o ambiente onde contextuamos o processamento de uma grande quantidade de dados para tratar problemas de ML (*Machine Learning* ou aprendizado de máquina), DL (*Deep Learning* ou aprendizado profundo) e *AI* (*Artificial Intelligence* ou inteligência artificial). Muitos outros setores também extraem benefícios com a computação de alto desempenho, pois ela ajuda a prover uma melhor acurácia e um maior detalhamento de modelos matemáticos que necessitam de um processamento massivo de dados. Como exemplo, podemos citar algumas áreas: a cosmologia, sismologia, simulação de dispositivos semicondutores, inteligência artificial, sistemas de reconhecimento de imagens, mapeando genético, etc.

3.2 Arquitetura computacional de GPUs

Como já comentado, o propósito inicial das GPUs são renderizações de imagens e é muito utilizada na execução de jogos eletrônicos de computador e consoles, sendo que as duas maiores fabricantes de GPUs nos dias de hoje são a NVIDIA e a AMD.

A figura 3.1 apresenta a arquitetura de uma GPU modelo Geforce GTX 8800, que foi uma das primeiras a suportar a tecnologia C ou CUDA:

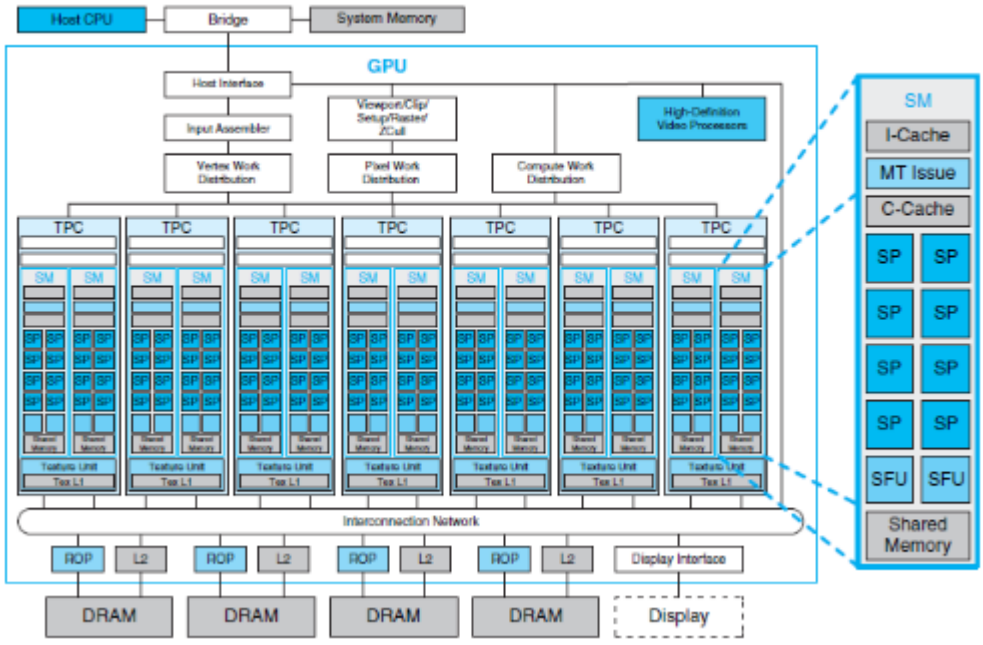

Figura 3.1 – GPU GeForce GTX 8800, primeira GPU a suportar plataforma CUDA.

Fonte: González (2018, p.10).

A seguir são explicados os principais componentes de uma GPU:

- **TPC (*Texture Processing Clusters*)**: Cada GPU consiste em múltiplas TPCs que são chips que agrupam múltiplos SMs (*Streaming Multiprocessors*). Cada TPC também contém um motor raster, responsável pela rasterização, que é o processo de converter uma imagem vetorial em uma imagem raster (formada por *pixels* ou pontos). De forma mais simplificada, este chip é arquitetado de forma a encapsular todos os principais componentes de processamento gráfico juntos.
- **SM (*Streaming Multiprocessors*)**: São processadores projetados para executar milhares de *threads* simultaneamente. Para gerenciar uma quantidade tão grande de *threads*, cada SM consiste em um barramento de cache L1 dedicado com seus respectivos núcleos associados e uma memória compartilhada (*shared memory*) cache L2, que armazenam os processamentos antes de extrair os dados de uma DRAM global GDDR (*Graphics Double Data Rate*), que são memórias específicas para gráficos. Temos também dentro de cada SM um conjunto de unidades de hardware responsáveis por executar instruções específicas de processamento, como SP (*Streaming Processors* ou *CUDA cores*), SFUs (*Special Function Units*) e LD/ST (*Load/Store*).
- **SP (*Streaming Processors*)**: Os SP ou *CUDA cores* são processadores dedicados para executar cálculos de lógica aritmética de 32-bits (ALU ou *Arithmetic-Logic Unit*) e cálculos de ponto flutuante de 64-bits (FPU ou *Floating Point Unit*).
- **SFU (*Special Function Units*)**: Os SFU são unidades de hardware responsáveis por executar instruções transcendentais, como, por exemplo, seno, cosseno e raiz quadrada.
- **LD/ST (*Load/Store*)**: LD/ST são unidades responsáveis em ler e escrever nas unidades de memória global.

Dado essa estrutura básica, temos que as GPUs andam avançando sua tecnologia e aumentando cada vez mais seu poder computacional, como o número de transistores, SPs, largura de banda, quantidade de memória, entre outras características. Não iremos entrar nos detalhes de cada arquitetura existente, porém, podemos citar algumas delas: Tesla, Fermi, Kepler, Maxwell, Pascal, Volta e Turing.

3.3 Modelos de programação paralela com GPUs

A complexidade arquitetural de uma GPU implica também em uma maior complexidade no desenvolvimento de aplicações. É fácil perceber que sem um modelo de programação projetado especificadamente para abstrair esse conceito de paralelismo torna inviável a implementação de soluções eficientes e com alto desempenho.

GPUs em sua natureza arquitetural são processadores *stream* (grande conjunto ordenado de dados), ou seja, podem operar em paralelo, executando um kernel em muitos registros em um fluxo de uma só vez (GPGU, 2020). Existem diversas linguagens de programação em GPUs, porém, as duas principais são CUDA (*Compute Unified Device Architecture*) e OpenCL (*Open Computing Language*).

Tomando como exemplo o CUDA temos que, de acordo com Resios (2011), o conceito principal é particionar o problema em subproblemas que podem ser resolvidos de forma independente através de blocos de threads paralelas e, portanto, cada subproblema dividido em partes ainda menores que podem ser resolvidos em paralelo pelas threads desses blocos. Sendo esses blocos independentes um dos outros, conseguimos obter boa escalabilidade.

Podemos citar algumas áreas que se beneficiam dos modelos de programação paralela, como: clusters HPC utilizando MPI (*Message Passing Interface*), processamento de imagem digital, computação geométrica, computação científica utilizada para previsão do tempo, modelagem molecular, astrofísica e também áreas de estudo mais recentes como redes neurais artificiais, aprendizado de máquina, aprendizado profundo e inteligência artificial.

3.4 Considerações finais

É imprescindível entender que o maior salto tecnológico que temos até o momento depende e muito das tecnologias envolvidas no processamento de computação paralela utilizando GPUs. Dessa forma, este capítulo nos ajuda a compreender um pouco mais sobre o seu funcionamento e dá uma base de conhecimento para avançarmos para os próximos tópicos, que é um estudo de modelos de predição de desempenho com base de dados de execuções de testes em GPUs.

4 MODELOS DE PREDIÇÃO EM CIÊNCIA DE DADOS

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

4.1 Introdução

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

4.2 Técnicas de predição utilizando modelos estatísticos

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

4.3 Algoritmos de aprendizado de máquina

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

4.4 Considerações finais

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

5 IMPLEMENTAÇÃO, RESULTADOS E DISCUSSÃO

Uma vez descrito toda a parte conceitual necessária para a compreensão prática deste trabalho, iremos agora apresentar o problema levantado, as bases de estudo, as técnicas de ciências de dados aplicadas, os resultados obtidos, assim como as devidas discussões e aprendizados adquiridos.

5.1 Introdução

A concepção inicial do tema de estudo está relacionada diretamente à área de interesse profissional do autor, que lida diariamente com avalições de desempenho de sistemas de missão crítica. Dado a complexidade em encontrar bases consistentes para estudo, onde seja possível aplicar conceitos de aprendizado de máquina e aprendizado profundo, conseguimos obter um conjunto de dados referente as medições de desempenho com o tempo de execução de um kernel GPU SGEMM (*Single precision GEneral Matrix Multiply*) para multiplicação de matrizes. O repositório oficial da base de dados pode ser encontrado em (DUA, GRAFF, 2019).

A proposta de contribuição deste trabalho é pode aplicar abordagens de aprendizado de máquina para conseguir obter um modelo para classificar e predizer o desempenho do sistema de multiplicação de matrizes sem a necessidade de realizarmos uma análise profunda ou obter um conhecimento específico do sistema em questão. Neste caso, queremos mostrar que é possível obter uma boa acurácia sem requerer de uma análise estática do código ou mesmo de criar um modelo analítico.

5.2 Descrição do problema

Implementar modelos de aprendizado de máquina para prever o tempo de processamento de uma GPU aplicada a um sistema de multiplicação de matrizes SGEMM.

Para tal problema iremos abordar técnicas de classificação como, por exemplo, SVM (*Support Vector Machines*), K-NN (*K-Nearest Neighbors* ou k-vizinhos mais próximos), RF (*Random Forest*), redes neurais artificiais, entre outros. Também iremos verificar a possibilidade de trabalharmos com métodos de predição utilizando regressões, como, por exemplo, regressão logística, regressão linear e assim por diante.

Dessa forma, pretendemos realizar uma série de experimentos, como por exemplo, variar os parâmetros das regressões e verificar quais deles apresentam melhores resultados, fazer a seleção de atributos de forma a otimizar os modelos gerados e por fim apurar e comparar os modelos, verificando se de fato conseguimos obter bons resultados de predição.

5.3 Descrição das atividades realizadas

Iremos agora abordar todos os passos implementados para a resolução do problema proposto, desde a concepção da base, passando pela preparação e tratamento dos dados, experimentos, as metodologias, análises realizadas e por fim os resultados obtidos.

5.3.1 Coleta e descrição dos dados

Esta base de dados foi obtida diretamente do repositório oficial de aprendizado de máquina da UCI (*University of California Irvine*) disponível em (DUA, GRAFF, 2019).

Conforme descrito em Dua e Graff (2019), este conjunto de dados mede o tempo de execução de um sistema de multiplicação de matrizes A * B = C, onde todas as matrizes tem tamanho 2048 X 2048 usando um *kernel* de GPU SGEMM parametrizável com 241.600 combinações de possíveis parâmetros. Para cada combinação testada, quatro medições de desempenho foram realizadas e seus resultados são relatados nas 4 últimas colunas. Todos os tempos estão medidos em milissegundos. Existem 14 parâmetros, onde os 10 primeiros são ordinais e as quatro últimas são variáveis binárias. Das 1.327.104 combinações de parâmetros totais, apenas 241.600 são viáveis, devido a várias restrições do *kernel*, e este conjunto contém o resultado de todas essas combinações viáveis. O experimento foi realizado em uma estação de trabalho rodando Linux Ubuntu v.16.04 com processador Intel Core i5 (3.5 GHz), 16 GB de memória RAM, uma GPU NVIDIA GeForce GTX 680 4GB e uma GeForce GTX 580 1.5GB. *Kernel* utilizado foi o ‘gemm_fast’ da biblioteca OpenCL ‘CLTune’.

A tabela 5.1 apresenta uma pequena amostra da base de dados, com seus parâmetros, número de linhas e número de colunas:

Tabela 5.1 – Amostragem da base de dados com medições de tempo de uma GPU kernel SGEMM

Número de linhas e colunas: (241600, 18)

	MWG	NWG	KWG	MDIMC	NDIMC	MDIMA	NDIMB	KWI	VWM	VWN	STRM	STRN	SA	SB	Run1 (ms)	Run2 (ms)	Run3 (ms)	Run4 (ms)
0	16	16	16	8	8	8	8	2	1	1	0	0	0	0	115.26	115.87	118.55	115.80
1	16	16	16	8	8	8	8	2	1	1	0	0	0	1	78.13	78.25	79.25	79.19
2	16	16	16	8	8	8	8	2	1	1	0	0	1	0	79.84	80.69	80.76	80.97
3	16	16	16	8	8	8	8	2	1	1	0	0	1	1	84.32	89.90	86.75	85.58
4	16	16	16	8	8	8	8	2	1	1	0	1	0	0	115.13	121.98	122.73	114.81

Fonte: tabela elaborada pelo autor.

A figura 5.1 apresenta os parâmetros relacionadas aos atributos preditivos do conjunto de dados da multiplicação de matrizes:

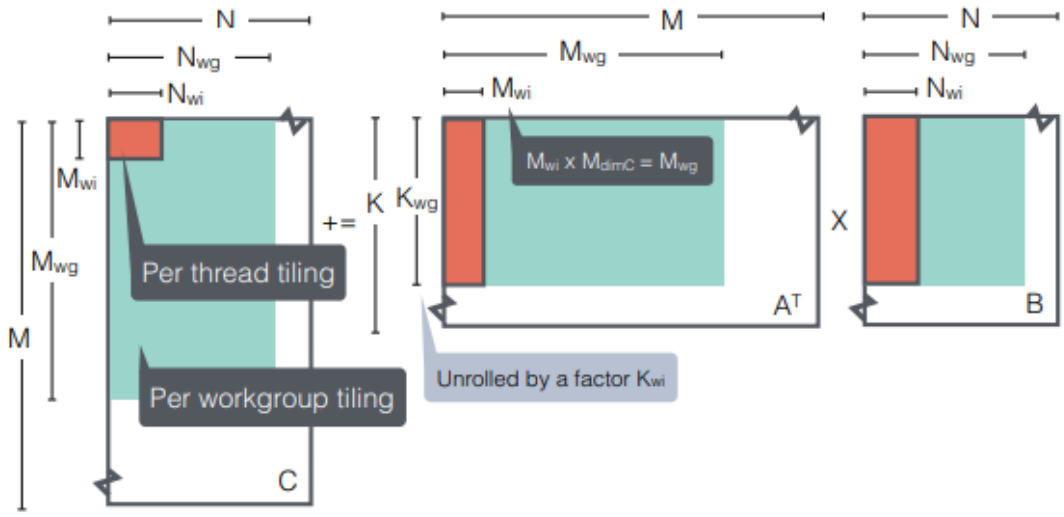

Figura 5.1 – Parâmetros da base de dados de multiplicação de matrizes

Fonte: Nugteren, Codreanu (2015, p.201).

Descrição dos parâmetros:

- **MWG, NWG e KWG**: mosaico 2D aplicado em um *workgroup* de *threads*. Esses parâmetros correspondem as dimensões das matrizes M, N e K respectivamente.
- **MDIMC, NDIMC**: parâmetros ajustáveis que definem o tamanho dos *workgroups* de *threads*. A unidade desses *workgroups* está definida na figura 5.1 como MWI e NWI, que são basicamente o tamanho do mosaico por *thread*. Eles são definidos pela seguinte relação: MWI = MWG/MDIMC e NWI = NWG/NDIMC. A sigla WI é uma abreviação de *work item*, ou seja, corresponde a um *thread*, considerando que estamos implementando um mosaico 2D à nível de *threads*.

- **SA, SB:** esses parâmetros binários são usados para armazenar o *workgroup* em memória. Quando eles estão setados para 1 = sim, a memória local é habilitada.
- **MDIMA, NDIMB**: parâmetros que são usados quando a memória local está ativada. Sendo assim elas podem ser ajustadas da seguinte forma: MDIMC x NDIMC = MDIMA x KDIMA = KDIMB x NDIMB. Portanto, MDIMA e NDIMB são parâmetros extras de *tunning* e o KDIMA e KDIMB são calculados de acordo com a equação apresentada.
- **STRM, STRN:** esses parâmetros binários são usados quando queremos habilitar ou desabilitar o acesso de um *thread* a memória externa (*off-chip memory*) do *workgroup* de *threads* definido. STRM é usada para a matriz A e C e STRN para a matriz B. STR vem do termo *stride* (passo). Se estiver habilitado o *stride* é ajustado para MDIMA e NDIMB, caso contrário ele é ajustado para 1 (*no stride*).
- **VWM, VWN:** a largura do vetor (VW: *vector width*) para carregamento e armazenamento de dados podem ser configuradas por estes dois parâmetros. VWM para as matrizes A e C e VWN para a matriz B.
- **KWI:** um fator onde é possível controlar o *kernel-loop* do KWG, fazendo um *loop-unrolling*. O *loop-unrolling* é basicamente uma técnica de transformação de um *loop* que tenta otimizar a velocidade de um programa em relação ao seu tamanho binário, esta é uma abordagem conhecida como compensação espaço-tempo. (LOOP UNROLLING, 2020)

A tabela 5.2 apresenta uma descrição estatística da base de dados para todos os seus parâmetros:

Tabela 5.2 – Estatística descritiva da base de dados

	MWG	NWG	KWG	MDIMC	NDIMC	MDIMA	NDIMB	KWI	VWM
count	241600.000000	241600.000000	241600.000000	241600.000000	241600.000000	241600.000000	241600.000000	241600.000000	241600.000000
mean	80.415364	80.415364	25.513113	13.935894	13.935894	17.371126	17.371126	5.000000	2.448609
std	42.469220	42.469220	7.855619	7.873662	7.873662	9.389418	9.389418	3.000006	1.953759
min	16.000000	16.000000	16.000000	8.000000	8.000000	8.000000	8.000000	2.000000	1.000000
25%	32.000000	32.000000	16.000000	8.000000	8.000000	8.000000	8.000000	2.000000	1.000000
50%	64.000000	64.000000	32.000000	8.000000	8.000000	16.000000	16.000000	5.000000	2.000000
75%	128.000000	128.000000	32.000000	16.000000	16.000000	32.000000	32.000000	8.000000	4.000000
max	128.000000	128.000000	32.000000	32.000000	32.000000	32.000000	32.000000	8.000000	8.000000

VWN	STRM	STRN	SA	SB	Run1 (ms)	Run2 (ms)	Run3 (ms)	Run4 (ms)
241600.000000	241600.000000	241600.000000	241600.000000	241600.000000	241600.000000	241600.000000	241600.000000	241600.000000
2.448609	0.500000	0.500000	0.500000	0.500000	217.647852	217.579536	217.532756	217.527669
1.953759	0.500001	0.500001	0.500001	0.500001	369.012422	368.677309	368.655118	368.677413
1.000000	0.000000	0.000000	0.000000	0.000000	13.290000	13.250000	13.360000	13.370000
1.000000	0.000000	0.000000	0.000000	0.000000	40.660000	40.710000	40.660000	40.640000
2.000000	0.500000	0.500000	0.500000	0.500000	69.825000	69.930000	69.790000	69.820000
4.000000	1.000000	1.000000	1.000000	1.000000	228.530000	228.310000	228.320000	228.320000
8.000000	1.000000	1.000000	1.000000	1.000000	3339.630000	3375.420000	3397.080000	3361.710000

Fonte: tabela elaborada pelo autor.

5.3.2 Preparação dos dados

Nesta seção vamos descrever toda a metodologia aplicada na preparação dos dados, realizando os devidos tratamentos para aplicar os algoritmos de aprendizado de máquina.

Primeira validação foi verificar dados duplicados e dados faltantes, porém, nada foi identificado. A seguir, vamos verificar a ocorrência de *outliers* nas variáveis alvo de tempo de execução. A figura 5.2 apresenta o gráfico *boxplot* com os outliers identificados:

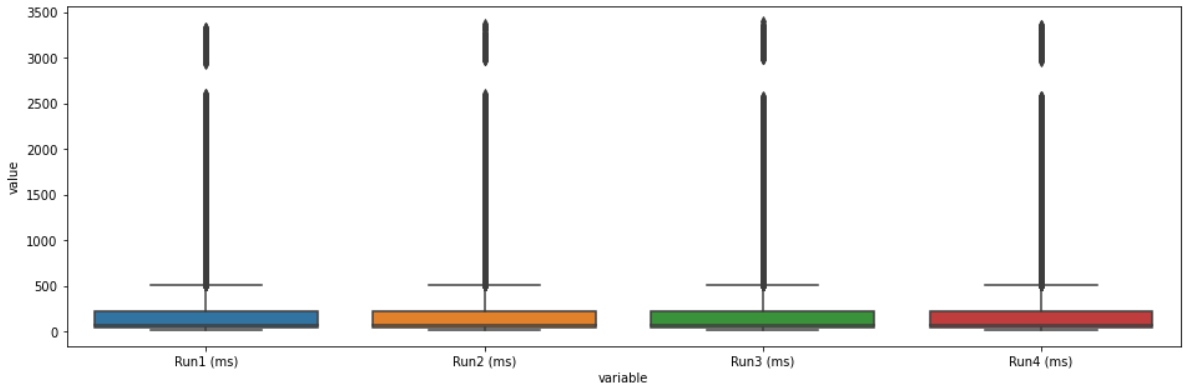

Figura 5.2 – Gráfico de boxplot com os outliers para os tempos de execução

Fonte: figura elaborada pelo autor.

Neste caso, vamos aplicar a metodologia do IQR (*InterQuartile Range* ou intervalo interquartil) para a identificação dos *outliers* e optamos pela estratégia de remoção desses dados. Após aplicar esta metodologia, tivemos uma redução na base de dados de 241.600 para 153.253 linhas. A figura 5.3 ilustra os gráficos de boxplot após remoção de *outliers*:

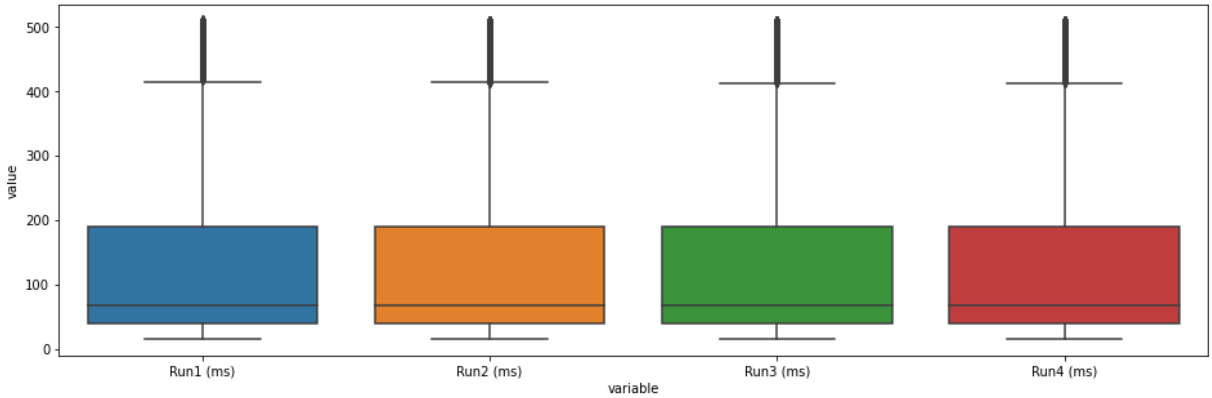

Figura 5.3 – Gráfico de boxplot com os outliers removidos para os tempos de execução

Fonte: figura elaborada pelo autor.

Dando sequência vamos verificar a correlação existentes entre as variáveis e por fim verificar a distribuição das variáveis alvo de tempo de execução.

A figura 5.4 apresenta a matriz de correlação entre todos os atributos da base:

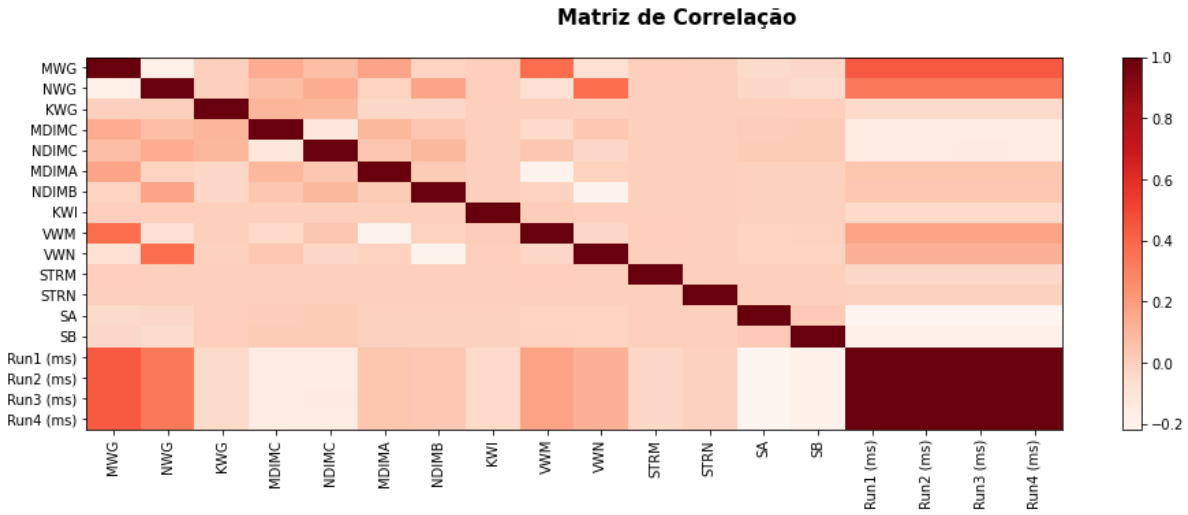

Figura 5.4 – Matriz de correlação entre todos os atributos da base de dados

Fonte: figura elaborada pelo autor.

Neste caso, aplicamos um critério para apresentar os atributos que tenham uma correlação mínima de 30% e na figura 5.5 apresentamos um gráfico de dispersão e histograma apresentando esses resultados:

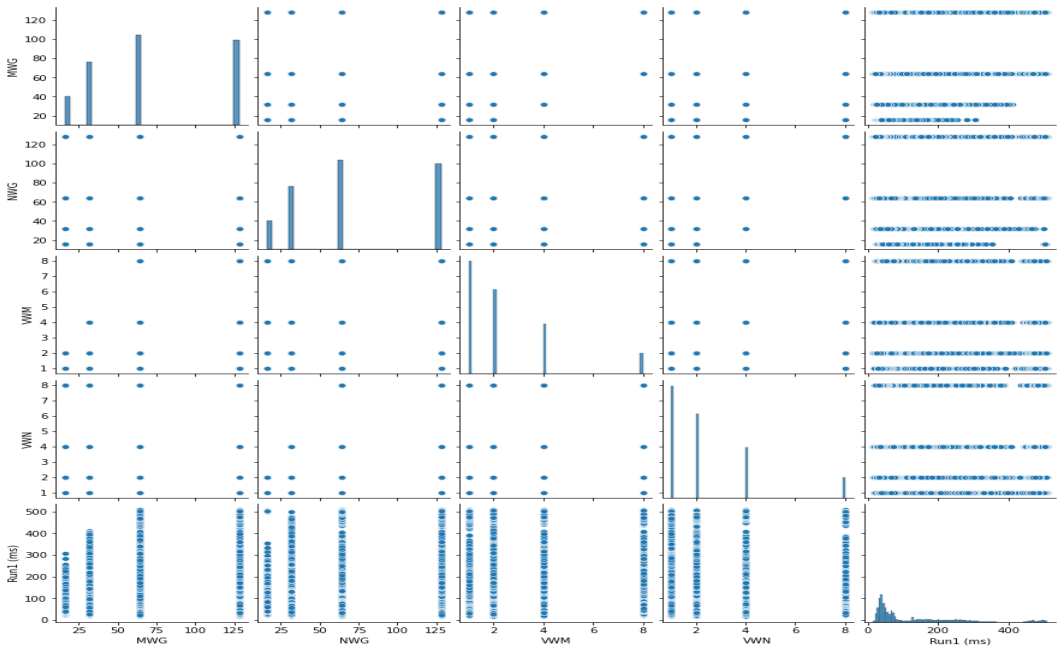

Figura 5.5 – Gráfico de dispersão e histograma entre variáveis com maior correlação

Fonte: figura elaborada pelo autor.

Até o momento de escrita deste trabalho, a última análise realizada, trata-se da verificação da distribuição dos dados das variáveis alvo. A figura 5.6 apresenta o histograma com a distribuição dos dados:

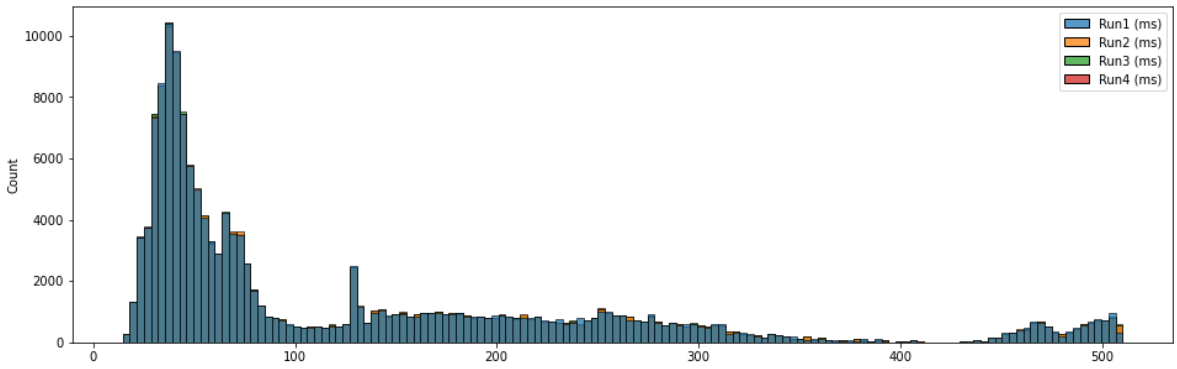

Figura 5.6 – Histograma com a distribuição dos dados de tempo de execução

Fonte: figura elaborada pelo autor.

Como próximos passos iremos começar a desenvolver os modelos de predição baseados na evolução realizada na base de dados com o objetivo de predição de desempenho.

5.3.3 Experimentos

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

5.3.4 Análises

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

5.4 Resultados obtidos

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

5.5 Considerações finais

Os principais resultados já estão expostos e o capítulo a seguir apresenta opiniões pessoais, considerações do autor sobre todo o conteúdo apresentado e também possíveis trabalhos futuros relacionados ao tema.

6 CONCLUSÕES E TRABALHOS FUTUROS

6.1 Conclusões

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum

6.2 Considerações finais

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

REFERÊNCIAS

BALDINI I., FINK S. J., and ALTMAN E., **Predicting gpu performance from cpu runs using machine learning**. In Computer Architecture and High PerformanceComputing (SBAC-PAD), 2014 IEEE 26th International Symposium on, pages 254–261. doi:10.1109/SBAC-PAD.2014.30.

BALLESTER-RIPOLL, R., PAREDES, E. G., PAIROLA, R., **Sobol Tensor Trains for Global Sensitivity Analysis**. In arXiv Computer Science / Numerical Analysis e-prints, 2017

DONGARRA, J., FOSTER, I., FOX, G., GROPP, W., KENNEDY, K., TORCZON, L., WHITE, A., **Sourcebook Of Parallel Computing**. San Francisco: Morgan Kaufmann Publishers, 2003.

DUA, D., GRAFF, C., (2019). **UCI Machine Learning Repository** [http://archive.ics.uci.edu/ml/datasets/SGEMM+GPU+kernel+performance]. Irvine, CA: University of California, School of Information and Computer Science. Acesso em: 05 out. 2020.

FALCH, T. L., ELSTER, A. C., **Machine Learning Based Auto-Tuning for Enhanced OpenCL Performance Portability**. 2015 Ieee International Parallel And Distributed Processing Symposium Workshop, [S.L.], p. 1231-1240, maio 2015. IEEE. http://dx.doi.org/10.1109/ipdpsw.2015.85.

GONZÁLEZ, M. T. A., **Performance Prediction Of Applications Executed on GPUs using a Simple Analytical Model and Machine Learning Techniques**. 2018. 109 f. Tese (Doutorado) - Curso de Mathematics And Statistics, Institute Of Mathematics And Statistics Of University Of São Paulo, Universidade de São Paulo, São Paulo, 2018.

GPGPU, 2020. Disponível em: https://pt.wikipedia.org/wiki/GPGPU. Acesso em: 03 out. 2020.

JAIN, R., **The Art of Computer Systems Performance Analysis:** techiniques for experimental desing, measurement, simulation and modeling. New York: John Wiley & Sons, 1991.

LOOP UNROLLING, 2020. Disponível em: https://en.wikipedia.org/wiki/Loop_unrolling. Acesso em: 05 out. 2020.

NUGTEREN, C., CODREANU, V., **CLTune**: A Generic Auto-Tuner for OpenCL Kernels. In: MCSoC: 9th International Symposium on Embedded Multicore/Many-core Systems-on-Chip. IEEE, 2015

QAZDAR, A., ER-RAHA, B., CHERKAOUI, C., MAMMASS, D., **A machine learning algorithm framework for predicting students performance**: A case study of baccalaureate students in Marocco. Springer Science+Business Media, LLC, part of Springer Nature, 2019.

RESIOS, A., **GPU Performance Prediction using Parametrized Models**. 2011. 66 f. Tese (Doutorado) - Computer Science, Utrecht University, Utrecht, 2011.

SAS, **Análises Preditivas: o que são e qual sua importância?** SAS. Disponível em: https://www.sas.com/pt_br/insights/analytics/analises-preditivas.html#:~:text=An%C3%A1lises%20preditivas%20usam%20dados%2C%20algoritmos,que%20poder%C3%A1%20acontecer%20no%20futuro Acesso em: 26 Jun. 2020.

TANENBAUM, A., STEEN, M. V., **Sistemas distribuídos:** princípios e paradigmas. 2.ed. Tradução de Arlete Simille Marques. São Paulo: Pearson Prentice Hall, 2007.

WONG, A., REXACHS, D., LUQUE, E., **Parallel Application Signature for Performance Analysis and Prediction**, in IEEE Transactions on Parallel and Distributed Systems, v.26, n.7, p. 2009-2019, 1 July 2015, Doi: 10.1109/TPDS.2014.2329688

GLOSSÁRIO

Apêndice A – Digitar o título do apêndice A

Apêndice B – Digitar o título do apêndice B

ANEXO A – Digitar o título do anexo A

ÍNDICE