# Carátula



**UNIVERSIDAD TÉCNICA PARTICULAR DE LOJA**

*La Universidad Católica de Loja*

**ÁREA TÉCNICA**

**INGENIERO EN SISTEMAS INFORMÁTICOS Y COMPUTACIÓN**

[Implementación](http://dspace.utpl.edu.ec/handle/20.500.11962/23888) de un Ambiente de Ciencia de Datos a través de la librería Open Source Rapids

**Autor**: Guarnizo Romero, José Alberto

**Director**: Mgtr. Elizalde Solano, René Rolando

LOJA - ECUADOR

2020

# Aprobación del director del trabajo de titulación

Loja, día, de mes, de año

Magister.

Fernanda Maricela Soto Guerrero

**Coordinadora de carrera**

Loja.

De mi consideración:

El presente trabajo de titulación denominado: Implementación de un Ambiente de Ciencia de Datos a través de la librería Open Source Rapids realizado por José Alberto Guarnizo Romero, ha sido orientado y revisado durante su ejecución, por cuanto se aprueba la presentación del mismo. Así mismo, doy fe que dicho trabajo de titulación ha sido revisado por la herramienta antiplagio institucional.

Particular que comunico para los fines pertinentes.

Atentamente,

Firma del Director del Trabajo de Titulación

René Rolando Elizalde Solano.

C.I:

# Declaración de **autoría** y cesión de derechos

“Yo, José Alberto Guarnizo Romero, declaro y acepto en forma expresa lo siguiente:

* Ser autor del Trabajo de Titulación denominado: Implementación de un Ambiente de Ciencia de Datos a través de la librería Open Source Rapids, de la Titulación Sistemas Informáticos y Computación, específicamente de los contenidos comprendidos en: Capítulo 1. Preparación Librería Rapids, Capítulo 2. Marco teórico, Capítulo 3. Trabajos Relacionados, Capítulo 4. Implementación de la librería Rapids, Conclusiones y Recomendaciones, siendo René Rolando Elizalde Solano, director del presente trabajo; y, en tal virtud, eximo expresamente a la Universidad Técnica Particular de Loja y a sus representantes legales de posibles reclamos o acciones judiciales o administrativas, en relación a la propiedad intelectual. Además, ratifico que las ideas, conceptos, procedimientos y resultados vertidos en el presente trabajo investigativo son de mi exclusiva responsabilidad.
* Que mi obra, producto de mis actividades académicas y de investigación, forma parte del patrimonio de la Universidad Técnica Particular de Loja, de conformidad con el artículo 20, literal j), de la Ley Orgánica de Educación Superior; y, artículo 91 del Estatuto Orgánico de la UTPL, que establece: “Forman parte del patrimonio de la Universidad la propiedad intelectual de investigaciones, trabajos científicos o técnicos y tesis de grado que se realicen a través, o con el apoyo financiero, académico o institucional (operativo) de la Universidad”.
* Autorizo a la Universidad Técnica Particular de Loja para que pueda hacer uso de mi obra con fines netamente académicos, ya sea de forma impresa, digital y/o electrónica o por cualquier medio conocido o por conocerse, sirviendo el presente instrumento como la fe de mi completo consentimiento; y, para que sea ingresada al Sistema Nacional de Información de la Educación Superior del Ecuador para su difusión pública, en cumplimiento del artículo 144 de la Ley Orgánica de Educación Superior.

Firma: ..............................................................

Autor: José Alberto Guarnizo Romero

C.I.: ..............................................................

# Dedicatoria

# Agradecimiento

# Índice de contenidos

Contenido

[Carátula I](#_Toc43999744)

[Aprobación del director del trabajo de titulación II](#_Toc43999745)

[Declaración de autoría y cesión de derechos III](#_Toc43999746)

[Dedicatoria V](#_Toc43999747)

[Agradecimiento VI](#_Toc43999748)

[Índice de contenidos VII](#_Toc43999749)

[Resumen 1](#_Toc43999750)

[Abstract 2](#_Toc43999751)

Introducción............................................................................................................................................3

[Capítulo uno 9](#_Toc43999753)

Problema de Investigación [4](#_Toc43999754)

[1.1 **Problemática** 4](#_Toc43999755)

[1.2 **Justificación** 4](#_Toc43999755)

[1.3 **Objetivos** 4](#_Toc43999755)

[*1.3.1* *General* 8](#_Toc43999761)

[*1.3.2* *Específicos* 8](#_Toc43999761)

[1.4 **Estrategia o Metodología de Desarrollo** 4](#_Toc43999755)

[1.5 **Estructura del Documento** 4](#_Toc43999755)

[Capítulo dos 9](#_Toc43999753)

Marco Teórico Librería Rapids [4](#_Toc43999754)

[2.1 **Ciencia de Datos** 4](#_Toc43999755)

[*2.1.1* *Big Data* 8](#_Toc43999761)

[2.1.1.1 Datos Estructurados..…..………………….………………………………………...10](#_Toc43999761)

[2.1.1.2 Datos Semiestructurados….………………………………………………………...11](#_Toc43999761)

[2.1.1.3 Datos no Estructurados…….………………………………………………...……..12](#_Toc43999761)

[2.1.1.4 Características…….…………………….………………….………………………...12](#_Toc43999761)

[2.1.1.5 Ventajas y Desventajas.…….……………………………………………………….14](#_Toc43999761)

[*2.1.2* *Procesamiento de Datos* 15](#_Toc43999761)

[2.1.2.1 Evolución del Procesamiento de Datos para llegar a Rapids…………...……....16](#_Toc43999761)

[*2.1.2.1.1 Distributed* *Storage*.…..………………………………………………...17](#_Toc43999761)

[*2.1.2.2.2 Spark In-Memory Processing……..…………………………………...20*](#_Toc43999761)

[*2.1.2.2.3 Gpu-Accelerated Computer…..……...………………………………..21*](#_Toc43999761)

[*2.1.2.2.4 Rapids………….………………….……………………………………..22*](#_Toc43999761)

[**2.2** **Gpu** 23](#_Toc43999760)

[*2.2.1* *Gpu vs Cpu* 25](#_Toc43999761)

[*2.2.2* *Arquitecturas Gpu* 27](#_Toc43999761)

[2.2.2.1 Arquitectura Nvidia GeForce ………………………………………………..……...29](#_Toc43999761)

[2.2.2.2 Arquitectura Unificada…………………...…….…………………………………….30](#_Toc43999761)

[2.2.2.3 Arquitectura orientadas a computación de propósito general sobre Gpu…......31](#_Toc43999761)

[2.2.2.4 Arquitectura Nvidia………...…………………………….…………………………..32](#_Toc43999761)

[*2.2.3* *Funcionamiento Gpu* 35](#_Toc43999761)

[2.2.3.1 Cuda ……………………………………………..……….…………………………...35](#_Toc43999761)

[2.2.3.2 Arquitectura Cuda….…………………………………….…………………………..33](#_Toc43999761)

[2.2.3.3 Ventajas y Desventajas Cuda ………………...……….…………………………...36](#_Toc43999761)

[**2.3** **Librería** **Open Source Rapids** 37](#_Toc43999760)

[*2.3.1* *Ambiente de Trabajo Rapids* 38](#_Toc43999761)

[2.3.1.1 Preparación de la Data ………….…………………………...………………….….39](#_Toc43999761)

[2.3.1.2 Machine Learning ………….…………..…….………….…………………………..40](#_Toc43999761)

[*2.3.1.2.1 Tipos de Algoritmos Machine Learning*.…………………..……….…41](#_Toc43999761)

[2.3.1.3 Visualización ……….………….…………..…………………………………………42](#_Toc43999761)

[*2.3.2* *Gpu Memory Apache Arrow* 43](#_Toc43999761)

[2.3.2.1 Funcionamiento de Apache Arrow …..……………………………….…….……...43](#_Toc43999761)

[2.3.2.2 Arquitectura de Apache Arrow….…………………………………………………..45](#_Toc43999761)

[*2.3.3* *Biblioteca Cudf* 46](#_Toc43999761)

[2.3.3.1 Características de Cudf …………………...………………………………………..46](#_Toc43999761)

[2.3.3.2 Funcionamiento Cudf ……..……………...…………………………….………..….47](#_Toc43999761)

[*2.3.4* *Biblioteca Cuml* 48](#_Toc43999761)

[2.3.4.1 Algoritmos soportados por Cuml ………………..…………….…………………...48](#_Toc43999761)

[*2.3.5* *Biblioteca Cugraph* 51](#_Toc43999761)

[2.3.5.1 Algoritmos soportados por Cugraph …………..…………………………………..52](#_Toc43999761)

[*2.3.6* *Biblioteca Cuxfilter* 53](#_Toc43999761)

[2.3.6.1 Arquitectura Cuxfilter ………………….……………..……………………………...54](#_Toc43999761)

[*2.3.7* *Dask* 55](#_Toc43999761)

[2.3.7.2 Dask y Rapids ………………….……………..………………………………….….56](#_Toc43999761)

[*2.3.8* *Integración con Bibliotecas de Deep Learning* 56](#_Toc43999761)

[2.3.8.1 Chainer ……………………………………………………….…..…………………..57](#_Toc43999761)

[*2.3.8.1.1 Características de Chainer*………….……………......………………..58](#_Toc43999761)

[2.3.8.2 Mxnet ..……………………………………………………….…………..…………...58](#_Toc43999761)

[*2.3.8.2.1 Características de Mxnet*…..……….…………………..……………...59](#_Toc43999761)

[2.3.8.3 Pytorch ……………………………………………..…….…………………………...59](#_Toc43999761)

[*2.3.8.3.1 Características de Pytorch*………….…………………..………….….60](#_Toc43999761)

[2.3.8.4 Numba ……………………………….……………..…….…………………………..60](#_Toc43999761)

[*2.3.8.4.1 Python y Numba*………………………………………………………...61](#_Toc43999761)

[*2.3.9* *Análisis e Implementación de la Arquitectura y características de Rapids*  62](#_Toc43999761)

[Capítulo tres 66](#_Toc43999762)

[Trabajos Relacionados 66](#_Toc43999763)

[**3.1** **Fuentes de Información** 66](#_Toc43999765)

[**3.2** **Cadenas de Busqueda** 66](#_Toc43999765)

[**3.3** **Criterios de Inclusión y Exclusión** 66](#_Toc43999765)

[**3.4** **Análisis de Trabajos Relacionados** 67](#_Toc43999765)

[*3.4.1* *Accelerated Simulation of Air Pollution Using Nvidia Rapids* 67](#_Toc43999761)

[*3.4.2* *Gpu Acceleration of PySpark using Rapids AI* 68](#_Toc43999761)

[*3.4.3* *Accelerating recommender system training 15x with Rapids* 69](#_Toc43999761)

[*3.4.4* *Security Alert Analysis Using Gpus* 69](#_Toc43999761)

[*3.4.5* *Accelerating Random Forest up to 45x using cuml* 70](#_Toc43999761)

[*3.4.6* *Combining Speed & Scale to Accelerate K-Means in Rapids cuml* 71](#_Toc43999761)

[*3.4.7* *Gpu Accelerated Cyber Log Parsing with Rapids* 71](#_Toc43999761)

[Capítulo cuatro 73](#_Toc43999762)

[Implementación de la Librería Rapids 73](#_Toc43999763)

[**4.1** **Instalación de la Librería Rapids** 73](#_Toc43999765)

[*4.1.1* *Instalación en Máquina Personal* 73](#_Toc43999761)

[4.1.1.1 Instalación con Docker………………...……………………………….…….……...76](#_Toc43999761)

[4.1.1.2 Instalación con Conda………………....……………………………….…….……..77](#_Toc43999761)

[*4.1.2* *Plataforma* *Google Colabority* 7](#_Toc43999761)9

[4.1.2.1 Entorno de Ejecución……..…………...……………………………….…….……...80](#_Toc43999761)

[4.1.2.2 Instalación Rapids…..…….…………...……………………………….…….……...81](#_Toc43999761)

[*4.1.2* *Plataforma* *BlazingSQL* 7](#_Toc43999761)

[**4.2** **Proceso de Extracción, Transformación y Carga** 8](#_Toc43999765)

[*4.2.1* *Extracción con Cudf* 7](#_Toc43999761)

[*4.2.2* *Transformación con Cudf* 7](#_Toc43999761)

[*4.2.3* *Carga con Cudf* 7](#_Toc43999761)

[**4.3** **Proceso de Machine Learning** 8](#_Toc43999765)

[*4.3.1* *Dividir Dataset en Train y Test con Cuml* 7](#_Toc43999761)

[*4.3.2* *Preprocesamiento con Cuml* 7](#_Toc43999761)

[*4.3.3* *Aplicación de Algoritmos Machine Learning con Cuml* 7](#_Toc43999761)

[4.3.3.1 Algoritmo Regresión Lineal ……..…………...…………………….….…….……...80](#_Toc43999761)

[4.3.3.2 Algoritmo Regresión Logística ……..…………...………………………….……...80](#_Toc43999761)

[**4.4** **Visualización de Gráficos** 8](#_Toc43999765)

[*4.3.1* *Gráficos Estadísticos de Barras con Cuxfilter* 7](#_Toc43999761)

[*4.3.2* *Gráficos Estadísticos de LÍneas con Cuxfilter* 7](#_Toc43999761)

[**4.5** Implementación de contraparte con Librerías Tradicionales 8](#_Toc43999765)

[Capítulo cinco 73](#_Toc43999762)

[Pruebas y Resultados 66](#_Toc43999763)

[**5.1** **Aceleración en el Proceso Extracción, Transformación y Carga** 73](#_Toc43999765)

[**5.2** **Aceleración en el Proceso Machine Learning** 73](#_Toc43999765)

[**5.3** **Aceleración en la Visualización de Gráficos** 73](#_Toc43999765)

[Conclusiones 9](#_Toc43999767)

[Recomendaciones 10](#_Toc43999768)

[Referencias 11](#_Toc43999769)

[Apéndice 12](#_Toc43999770)

**Índice de tablas**

[Tabla 1. Núcleos de procesamiento de un dispositivo Cuda](#_Toc40382068)

[Tabla 2. Algoritmos soportados por Cuml](#_Toc40382068)

[Tabla 3. Algoritmos soportados por Cugraph](#_Toc40382068)

[Tabla 4. Modulos Cuxfilter](#_Toc40382068)

[Tabla 5. Fast Chainer](#_Toc40382068)

[Tabla 6. Intuitive Chainer](#_Toc40382068)

[Tabla 7. Componentes Pytorch](#_Toc40382068)

[Tabla 8. Cadenas de Busqueda](#_Toc40382068)

[Tabla 9. Criterios de Inclusión y Exclusión](#_Toc40382068)

[Tabla 10. Prerrequisitos de Instalación Rapids](#_Toc40382068)

[Tabla 11. Especificaciones computador portátil](#_Toc40382068)

[Tabla 12. Tipos de extracción y lectura con cudf](#_Toc40382068)

[Tabla 13. Características de Transformación a los datos con cudf](#_Toc40382068)

[Tabla 14. Características de carga a los datos con cudf](#_Toc40382068)

[Tabla 15. Características de tipos de transformación a datos categorizados](#_Toc40382068)

[Tabla 16. Desarrollo y configuración para utilizar el algoritmo Regresión Lineal](#_Toc40382068)

[Tabla 17. Caracaterísticas del algoritmo Regresión Lineal de cuml](#_Toc40382068)

[Tabla 18. Desarrollo y configuración para utilizar el algoritmo Regresión Logística](#_Toc40382068)

[Tabla 19. Características del algoritmo Regresión Logística de cuml](#_Toc40382068)

[Tabla 20. Características de cuxfilter para la creación del gráfico de barras](#_Toc40382068)

[Tabla 21. Características de cuxfilter para la creación del gráfico de líneas](#_Toc40382068)

[Tabla .](#_Toc40382068)

[Tabla .](#_Toc40382068)

[Tabla .](#_Toc40382068)

[Tabla .](#_Toc40382068)

[Tabla .](#_Toc40382068)

[Tabla .](#_Toc40382068)

[Tabla .](#_Toc40382068)

[Tabla .](#_Toc40382068)

[Tabla .](#_Toc40382068)

[Tabla .](#_Toc40382068)

[Tabla .](#_Toc40382068)

[Tabla .](#_Toc40382068)

[Tabla .](#_Toc40382068)

**Índice de figuras**

[Figura 1: Ciencia de Datos](#_Toc40381998)

[Figura 2: Representación de Datos Estructurados](#_Toc40381998)

[Figura 3: Representación de Datos Semiestructurados](#_Toc40381998)

[Figura 4: Representación de Datos no Estructurados](#_Toc40381998)

[Figura 5: Procesamiento Hadoop, leyendo desde el disco](#_Toc40381998)

[Figura 6: Spark procesamiento en memoria](#_Toc40381998)

[Figura 7: Procesamiento tradicional GPU](#_Toc40381998)

[Figura 8: Rapids](#_Toc40381998)

[Figura 9: Modelo de computación heterogéneo CPU + GPU](#_Toc40381998)

[Figura 10: Comparativa de memoria lógica de control para CPU y GPU](#_Toc40381998)

[Figura 11: Interconexión entre CPU y GPU mediante PCLe](#_Toc40381998)

[Figura 12: Arquitectura básica de GPU](#_Toc40381998)

[Figura 13: Esquema de la Arquitectura de la GPU GeForce de Nvidia](#_Toc40381998)

[Figura 14: Comparativa de asignación de procesadores de una GPU en el procesamiento de vértices y de fragmentos en arquitecturas unificadas y no unificadas](#_Toc40381998)

[Figura 15: Esquema de arquitecturas G80 de Nvidia](#_Toc40381998)

[Figura 16: Arquitectura Cuda](#_Toc40381998)

[Figura 17: Ciencia de Datos acelerados de GPU de extremo a extremo con Rapids](#_Toc40381998)

[Figura 18: Fases de Preparación de Datos](#_Toc40381998)

[Figura 19: Sin Apache Arrow](#_Toc40381998)

[Figura 20: Con Apache Arrow](#_Toc40381998)

[Figura 21: Arquitectura Apache Arrow](#_Toc40381998)

[Figura 22: Estructura de biblioteca Cudf](#_Toc40381998)

[Figura 23: Arquitectura Cuxfilter](#_Toc40381998)

[Figura 24: Proceso de Chainer Define-by-Run](#_Toc40381998)

[Figura 25: Proceso de Trabajo Python-Numpy](#_Toc40381998)

[Figura 26: Arquitectura Rapids](#_Toc40381998)

[Figura 27: Evaluación comparativa de XGBoost con CPU, GPU y XGBoost cudf+Gpu](#_Toc40381998)

[Figura 28: PySpark + Rapids y PySpark nativo](#_Toc40381998)

[Figura 29: Análisis comparativo de alertas con una sola GPU](#_Toc40381998)

[Figura 30: Evaluación comparativa de múltiples nodos, múltiples GPUs cuml y Dask-Cuml](#_Toc40381998)

Figura 31: Tiempo de procesamiento a los datos de registro de eventos Windows…..

[Figura 32: Nvidia Activa](#_Toc40381998)

[Figura 33: Drivers Disponibles](#_Toc40381998)

[Figura 34: Gpu Nvidia Activa](#_Toc40381998)

[Figura 35: Instalación de la librería Rapids al contenedor creado en la máquina personal](#_Toc40381998)

[Figura 36: Iniciar los cuadernos de JupyterLab con Docker + Rapids](#_Toc40381998)

[Figura 37: Crear entorno en la plataforma de Conda e Instalación de Rapids](#_Toc40381998)

[Figura 38: Plataforma de desarrollo Jupyter Notebook con Conda + Rapids](#_Toc40381998)

[Figura 39: Entorno de ejecución GPU Google Colabority](#_Toc40381998)

[Figura 40: Tipos de GPU Tesla en Google Colabority e Instalación de Rapids](#_Toc40381998)

[Figura 41: Plataforma BlazingSQL + Rapids](#_Toc40381998)

[Figura 42: Importar bibliotecas de Rapids conjuntamente con otras librerías a utilizar](#_Toc40381998)

[Figura 43: Extracción y Lectura de la Información con cudf](#_Toc40381998)

[Figura 44: Fase de Transformación de la Data proceso 1](#_Toc40381998)

[Figura 45: Fase de Transformación de la Data proceso 2](#_Toc40381998)

[Figura 46: Fase de Transformación de la Data proceso 3](#_Toc40381998)

[Figura 47: Fase de Transformación de la Data proceso 4](#_Toc40381998)

[Figura 48: Exploración y visualización de la Data Transformada](#_Toc40381998)

[Figura 49: Carga de los datos](#_Toc40381998)

[Figura 50: Dividir la data en Train y Test](#_Toc40381998)

[Figura 51: Transformar a datos categóricos](#_Toc40381998)

[Figura 52: Utilizando el algoritmo de cuml Regresión Lineal](#_Toc40381998)

[Figura 53: Utilizando el algoritmo de cuml Regresión Logística](#_Toc40381998)

[Figura 54: Creación del gráfico estadístico de Barras](#_Toc40381998)

[Figura 55: Gráfico estadístico de Barras](#_Toc40381998)

[Figura 56: Creación del gráfico estadístico de Líneas](#_Toc40381998)

[Figura 57: Gráfico estadístico de Líneas](#_Toc40381998)

# Resumen

La Librería Open Source Rapids ejecuta canalizaciones de datos y análisis de extremo a extemo utilizando las principales características del poder de la *GPU* (Unidad de procesamiento gráfico) de Nvidia, dichas GPUS se encuentran instaladas en cualquier máquina personal en donde Rapids puede ser utilizado, la librería maneja grandes volúmenes de información en tiempo real y trabaja con computación acelerada, su principal enfoque es realizar una analítica de la información o datos, utilizando el proceso de extracción, transformación y carga e implementar todo un conjunto de algoritmos *Machine* *Learning*; se da realce a sus principales funcionalidades, características y todo su conjunto de bibliotecas que tiene la librería, además, se utilizó la Plataforma Cloud BlazingSQL para la implementación de la librería Rapids, donde se resalta el proceso de construcción de un Ambiente de Ciencia de Datos para los procesos de ETL, Machine Learning y Visualización; se implemento una contraparte utilizando librerías que aún no tiene soporte para GPU como *Pandas* para el proceso de ETL, para el proceso de Machine Learning con *Scikit-learn* y para la visualización con *Matplotlib*, con el fin de presentar resultados de aceleración o rapidez, es decir, resaltar la aceleración que tiene Rapids cuando se está trabajando con grandes volúmenes de datos.

*Palabras claves***:** Ciencia de Datos, GPU, Machine Learning.

# Abstract

Rapids

*Keywords***:** Data Science, GPU, Machine Learning.

# Introducción

La cantidad de datos producidos en la actualidad son de gran volumen, es creciente en instituciones públicas y privadas, producen millones de datos al día, empresas como: bancos, entidades del estado, negocios independientes, entre otros, aportan considerables cantidades de información que se encuentra almacenadas en Base de Datos, información almacenada en la nube (internet) y otros medios con la finalidad de almacenarlas y respaldarlas, lo que causa que no tenga un adecuado tratamiento en cuanto a su procesamiento, implementación y utilización para general valor y rapidez.

La implementación de un ambiente de ciencia de datos a través de la Librería Open Source Rapids, se enfoca en analizar un conjunto de datos abiertos utilizando las principales características de la librería.

Mediante un análisis completo de la librería Rapids permitirá manejar grandes volúmenes de datos de información en tiempo real, además, poder determinar y hacer uso de las características más importantes de la librería que permiten: ocupar el máximo de recursos disponibles, tiempos mínimos de cada proceso y capacidad para el análisis, procesamiento e implementación de algoritmos Machine Learning.

El trabajo de titulación se realizó con el interés de conocer, analizar, ampliar y utilizar la librería Rapids, para determinar un resultado de este análisis e implementación, tanto para investigadores, docentes y estudiantes puedan ser uso de esta librería que permitirá ayudar al crecimiento de una empresa, negocio o una institución, analizando un gran volumen de cantidad de datos y que los tiempos de cada proceso como extracción, transformación, carga, implementación de algoritmo Machine Learning y la visualización de forma gráfica sea rápida y de forma acelerada para que los datos generan valor, sean entendibles y generen ingresos.

# Capítulo uno

# Problema de Investigación

## **Problemática**

La generación de datos crece minuto a minuto, grupos de investigación, instituciones públicas capturan datos de sus procesos internos a gran escala. Dicha información necesita ser proceda e interpretada para un posterior uso específico de acuerdo con las necesidades de la organización que lo necesite.

Para esto empresas como Nvidia, a través de su equipo de investigación han creado la librería Rapids, que permiten realizar un tratamiento eficaz y rápido de forma acelerada para obtener resultados que sirven a las empresas en la toma acertadas de decisiones. Permitiendo además que utilicen sus recursos de forma optimizada para cubrir las verdaderas necesidades de sus usuarios y su adaptación a diversos escenarios.

## **Justificación**

En el desarrollo de la presente investigación se pretende realizar un análisis de la librería Rapids que permitan realizar un análisis exploratorio a un conjunto de datos abiertos, procesamiento a los datos en tiempo real, la implementación y ejecución de Rapids. Cuyo principal objetivo es la implementación de un Ambiente de Ciencia de Datos a través de la Librería Open Source Rapids, debido a que actualmente las empresas e instituciones están realizando el análisis de información para generar valor tanto monetario como estratégico que ayudan a su crecimiento lo que resulta de gran interés y curiosidad.

En al ámbito educativo el trabajo de titulación de investigación servirá como referencia y base para estudiantes que realicen posteriores trabajos sobre implementar Ambientes de Ciencia de Datos utilizando la librería. Así que los resultados obtenidos de las pruebas de funcionalidad y rendimiento de esta librería puedan servir para determinar la eficacia en el procesamiento, entrenamiento Machine Learning, análisis de grandes cantidades de datos y mostrar la integridad y flexibilidad para relacionarse con diferentes herramientas, dando a conocer con estos resultados lo importante del la librería Rapids.

## **Objetivos**

### *General*

Implementar un ambiente de ciencia de datos a través de la Librería Open Source Rapids.

### *Específicos*

* Elaborar una investigación documentada sobre la Librería Open Source Rapids*.*
* Realizar analítica de grandes volúmenes de datos a través del uso de las características de la Librería Open Source Rapids.
* Implementar algoritmos de Machine Learning haciendo uso de Librería Open Source Rapids.

## **Estrategia o Metodología de Desarrollo**

Para el desarrollo del presente trabajo de titulación se ha considerado los siguientes puntos:

* Recolección de información de fuentes académicas confiables.
* Revisión de casos de éxito relacionados a la Librería Open Source Rapids.
* Experimentación y puesta en funcionamiento de la Librería Open Source Rapids.
* Plan de pruebas y resultados con la Librería Open Source Rapids.

## **Estructura del Documento**

En el Capítulo 1 se hace la presentación formal de la problemática que da raíz al tema, la relación entre el objetivo general, los objetivos específicos y la metodología utilizada en el Trabajo de Titulación.

En el Capítulo 2 constituye el encuadre del problema que se va a tratar dentro de limitantes teóricas y constituyen el punto de partida para orientar el desarrollo de la investigación. Se incluyen estudios relacionados, problemas relacionados y situación actual de la Librería Open Source Rapids.

En el Capítulo 3 se enfoca en un análisis de todos los trabajos relacionados, en donde se há utilizado e implementado la Librería Open Source Rapids aplicando su conjunto de bibliotecas como tema de éxito.

En el Capítulo 4 se realiza la implementación de la Librería Open Source Rapids, utilizando sus características, funcionalidades, aplicación de algoritmos Machine Learning y desarrollo de gráficos estadísticos utilizando la librería, además, se implemento un Ambiente de Ciencia de Datos con librerías tradicionales como una contraparte.

En el Capítulo 5 consiste en la presentación de pruebas y resultados de aceleración de la Librería Open Source Rapids, en donde será comparada con otras librerías tradicionales.

# Capítulo dos

# Marco Teórico

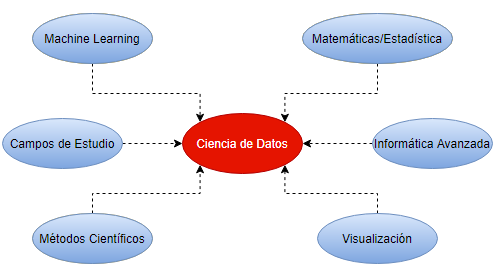


## **Ciencia de Datos**

La ciencia de datos es parte muy fundamental de todas las ciencias de aprendizaje automático y profundo, la inteligencia artificial y de negocios, involucra métodos científicos, procesos y sistemas para extraer conocimiento para un mejor entendimiento de los datos, con el fin de que esos datos sean útiles a nivel económico, social, educativo y empresarial.

**Figura 1**

Ciencia de Datos



*Nota.* Ciencia de Datos [Gráfico], por Autor, 2020.

La figura 1 demuestra de manera global lo que involucra y compreende la ciencia de datos, como: conocimientos en el campo de estúdio (geología, finanzas, medicina, etc.), se toman en cuenta aspectos computacionales, además, involucra matemáticas y estadística, punto fuerte es el aprendizaje automático o machine Learning[[1]](#footnote-1), visualización de protótipos de software.

Además, la ciencia de datos entiende y permite analizar los datos desde un punto de vista comercial, el principal objetivo es proporcionar la predicción más precisa, que permite impedir que una empresa o un empresario pierda en el futuro.

Cabe destacar que la misma ha tenido un sin número de definiciones y se lo relaciona muy a menudo con la Big Data[[2]](#footnote-2).

### *Big Data*

Big Data supone la confluencia de una multitud de tendencias tecnológicas que se han ido consolidando durante los años 2011 y 2013, además han explosionado e irrumpido con gran fuerza en distintas organizaciones, instituciones y empresas.

Joyones Aguilar (2016) desde su punto de vista científico describe la Big Data de la siguiente manera:

Los grandes datos o volúmenes de datos han ido creciendo de modo ascendente, los Big Data están brotando por todas partes utilizándolos adecuadamente, una gran ventaja competitiva a las organizaciones tanto internacionales como locales. En general existen diferentes aspectos donde casi todas las decisiones están de acuerdo y con conceptos consistentes para capturar la esencia de lo que es Big Data: Se interpreta como crecimiento exponencial de la creación de grandes volúmenes de datos, origen o fuentes de datos y la necesidad de su captura, almacenamiento y análisis para conseguir el mayor beneficio para organizaciones y empresas junto con las oportunidades que ofrecen y los riesgos de su no adopción.

Cabe recalcar que la Big Data tiene muchas definiciones que han sido implementadas por grandes científicos tecnológicos, por medio de ello se abarcara tres definiciones que se puedan entender de manera conceptual.

Para Joyones Aguilar (2016) la primera definición corresponde a Mickinsey Global Institute da a conocer:

Big Data se refiere a los conjuntos de datos cuyo tamaño estas más allá de las capacidades de las herramientas típicas de software de bases de datos para capturar, gestionar y analizar. Mickinsey destaca que la definición puede variar por cada sector, dependiendo de cuales sean los tipos de herramientas de los conjuntos de datos en ese sector o indústria.

Para Joyones Aguilar (2016) la segunda definición que aporta la empresa multinacional de auditoría *Deloitte* es que: “Big Data aplica un conjunto de datos cuyo volumen supera la capacidad de las herramientas informáticas de su uso común, para capturar, gestionar y procesar datos en un lapso de tiempo razonable”

Para Joyones Aguilar (2016) la tercera y última definición que aporta sobre la Big Data es que: “Big Data son los grandes conjuntos de datos que tiene tres características principales: volumen, velocidad y variedad”

En definitiva la Big Data puede variar de distintas formas, según las características que tiene cada empresa o una organización en general, es decir, para algunas organizaciones lo primero que importa es el volumen, capturar la información la cual va a hacer procesada, guardada y actualizada para después incorporarla en procesos de negocio que pueden ser beneficiosos para las organizaciones, para otros les interesa que la información sea guardada con una velocidad impresionante pero en tiempo real, como enfoque cada organización busca el beneficio enfocado en la Big Data.

Existen algunos conceptos relacionados como *Data Warehouse*, *Data Mining* y *Cloud Computing.*

**Data Warehouse**

El Data Warehouse es una evolución de los sistemas de bases de datos relacionales, es un proceso, no un producto. En 1988 los investigadores de *IBM[[3]](#footnote-3) Barry Devlin* y *Paul Murphy* inventaron el término *Warehouse* de información, aunque el considerado padre de los Data Warehouse es *William Harvey Inmon*.

La definición de *William Harvey Inmon* dice: “Una colección de datos que sirve de apoyo a la toma de decisiones, organizados por temas, integrados, no volátiles y en los que el concepto de tiempo varía respecto a los sistemas tradicionales” (López García, 2012).

**Data Mining**

La minería de datos es una herramienta que permite extraer conocimiento de los datos que tenemos almacenados para tratarlos y convertirlos en información útil y objetiva que ayudará al empresario o toda una organización a tomar las decisiones más adecuadas.

López García (2012) describe a la Minería de Datos segun sus conceptos y definiciones de la siguiente manera:

Según el portal *Daedalus* la Minería de Datos se define como la extracción no ligera de información implícita, previamente desconocida y potencialmente útil. En la actual sociedad de la información, la minería de datos es una herramienta fundamental para analizar y explotar de forma eficaz los objetivos de cualquier organización.

**Cloud Computing**

Es una tecnología joven al igual que Big Data brinda la posibilidad de brindar disntintos servicios a través de internet.

Para López García (2012) esta tecnología busca tener todos nuestros archivos e información almacenados en todo el internet sin preocuparnos de tener la capacidad suficiente para almacenar dicha información: “Cloud Computing coge fuerza cuando la provisión de hardware se convierte en un problema, además de costes monetarios los tiene de espacio, escalabilidad es aquí donde Cloud Computing es una gran alternativa”

#### **Datos Estructurados**

Joyones Aguilar (2016) pronunica que los Datos Estructurados se los considera como datos tradicionales, los cuales son datos con un formato y un esquema fijo, posee campos fijos. En estos casos son los datos de las bases de datos relacionales, las hojas de cálculo y los archivos, en fin es información entendible para los usuarios.

Los datos estructurales se componen por distintas piezas de información, también vienen en un formato ya especificado, además, estos formatos son conocidos como: fecha, nacionalidad, cedula, etc.

**Figura 2**

Representación de Datos Estructurados



*Nota.* Representación de Datos Estructurados [Gráfico], por Autor, 2020.

La figura 2 representa de manera gráfica los tipos de datos estructurados que pueden ser almacenados dentro de una base de datos relacional, estos datos pueden ser: estadísticos, información de usuarios (nombres, apelidos, cédula, etc), cadenas de texto, contenido de registros, por lo que es una cuestión fácil de usar.

#### **Datos Semiestructurados**

Los Datos Semiestructurados tienen un flujo que pueden ser definidos, pero no es fácil de entender para el usuário, es decir que no tienen formatos fijos, así como los datos estructurados.

Joyones Aguilar (2016) describio que la lectura de los datos requiere el uso de reglas complejas que determinan como proceder después de la lectura de cada pieza de información, como ejemplo para dar a conocer los datos semiestructurados son los web logs. Un web log este compuesto por diferentes piezas de información, cada una sirven para un propósito en específico, ejemplo son las etiquetas de *HTML[[4]](#footnote-4)* y *XML[[5]](#footnote-5)*.

**Figura 3**

Representación de Datos Semiestructurados



*Nota.* Representación de Datos Semiestructurados [Gráfico], por Autor, 2020.

La figura 3 representa de manera gráfica los tipos de datos semiestructurados, se entiende que este tipo de datos es difícil de entender, pero en si son una mezcla de datos estructurados, además, representan una organización definida en sus metadatos donde se describen los objetos y sus relaciones.

#### **Datos no Estructurados**

Los Datos no estructurados son datos sin tipos predefinidos. Estos se almacenan como documentos u objetos sin estructura uniforme, y se tiene poco o ningún control sobre ellos, como ejemplo de estos datos son: texto, video, audio, fotografía, etc. Al menos el 80% de la información de las organizaciones contiene este tipo de datos.

**Figura 4**

Representación de Datos no Estructurados



*Nota.* Representación de datos no estructurados [Gráfico], por Autor, 2020.

La figura 4 representa de manera gráfica los tipos de datos no estructurados, estos datos no tienen un formato en específico, pueden estar almacenados en múltiples formatos como texto, áudio, pdf, word, correos electrónicos, etc.

#### **Características**

Los datos pueden proceder de redes sociales, logs, registros de servidores web, sensores, imágenes de satélites, audios, radio, etc. Dentro de ellos se destacarán las características principales que contiene la Big Data*:*

La primera característica que se destaca es volumenJoyones Aguilar (2016) pronuncia que distintas empresas, indústrias, organización contiene y guarda grandes volúmenes cantidades de información, desde el año 2000 se han almacenado en el mundo 800.000 petabytes, se espera que en el año 2020 se alcance los 35 *zettabytes*:

Las organizaciones que se enfrentan a volúmenes masivos de datos desconocen cómo pueden gestionar esta información, es por eso que se han creado librerías que puedan analizar y satisfacer las necesidades del mismo, es por eso que más adelante se hablara de Rapids, permite gestionar la información en grandes volúmenes de datos. IBM plantea que el volumen de datos disponible en las organizaciones hoy en día está en ascenso mientras que el porcentaje de los datos que se analiza está en disminución.

La segunda caractertistica es la velocidad de los datos es de sumamente importancia para el creciente flujo de información en tiempo real que almacena cada organización. Requiere que el procesamiento y posterior análisis debe hacerse en tiempo real para la mejora de toma de decisiones y el tiempo de cada proceso sea más rapido. La importancia de la velocidad de los datos se une a las características de volumen y variedad, de modo que la idea de velocidad no sea asociada a la tarea de crecimiento de los depósitos o almacenes de datos, sino que se aplica a la definición al concepto de los datos en movimento.

La tercera característica corresponde a la variedad de los datos Joyones Aguilar (2016) pronunica lo siguiente:

Una organización puede almacenar distintos tipos de datos, pueden ser datos estructurados o no estructurados y cuando estos dos se analizan juntos se requieren de nuevas técnicas. Variedad representa todos los tipos de datos, y supone un desplazamiento fundamental en el análisis de requisitos desde los datos estructurados tradicionales hasta la inclusión de los datos en bruto, semiestructurados y no estructurados como parte del proceso fundamental. Sin embargo, el éxito de una organización dependerá de su capacidad para resaltar el conocimiento de los diferentes tipos de datos disponibles en ella, que incluirá tanto los datos tradicionales como los no tradicionales.

La cuarta característica corresponde a la veracidade de los datos Joyones Aguilar (2016) pronuncia lo siguiente**:**

Según IBM, en su definición de Big Data*,* al comentar la característica de veracidad proporciona un dato estremecedor: “Uno de cada tres líderes de negocio no se fija de las informaciones que utilizan para tomar decisiones”. El establecimiento de la veracidad o fiabilidad de *Big Data* supone un gran reto a las fuentes de datos que crecen(Joyones Aguilar, 2016).

La quinta y ultima característica corresponde al valor que tiene los datos, cada organización se enfoca en obtener información de los grandes datos de manera rentable y factible, el cual les ayudara a crecer exponencialmente, aquí es donde las tecnologías de código abierto permiten realizar el análisis de los datos obtienen una fuente de valor muy determinante que les permita a las organizaciones saber que datos son más factibles o que datos no lo son.

#### **Ventajas y Desventajas**

Big Data produce un gran avance, permite automatizar el proceso en gran medida, permitiendo obtener una respuesta de los datos muy rápido, existen ventajas y desventajas que serán detalladas de la siguiente manera:

Ventajas:

* Implementación de mejoras tecnologías o librerías que permitan la posibilidad de analizar, procesar datos en tiempo real y descubrir necesidades de puntos de mejora para una organización.
* El análisis de los datos puede mejorar sustancialmente dentro de la organización, permitirá reducir numerosos riesgos que puedan existir.
* Facilitar que las empresas puedan evaluar productos mediante el análisis de datos, obteniendo información muy valiosa, permitirá crear nuevos productos y ofrecer mejores servicios.
* Mejorar la accesibilidad y fluidez de la información dentro de la empresa y organización.

Desventajas:

* Estrada et al. (2018) describio que las empresas que han evolucionado y han decidido que su empresa siga a delante para conseguir metas más ambiciosas, suelen minimizar el riesgo, es decir lo más importante son los resultados que la inversión genera mas no se escatima ni los recursos necesarios para afrontar un proyecto de Big Data, así como también los costos associados.
* Las acciones lentas permiten obtener un duro fracaso para una organización, esperan que otros lo usen para ver si funciona o no. A demás de esto, aunque las empresas sean del mismo sector, siempre la información va a hacer diferente y el análisis no va a hacer el mismo.

### *Procesamiento de Datos*

Los datos pueden ser cualquiera dato como número o carácter que son representados por distintos valores, pueden ser manipulados en otras formas para que sean útiles y comprensibles, convirtiendo los datos en información.

El procesamiento de datos es utilizando principalmente por organizaciones, permiten tener sistemas de grandes transacciones de un sin número de datos. Además, el procesamiento es en tiempo real, se debe tener en cuenta que el tipo de procesamiento es instantáneo, esto quiere decir que el dato o información llega se procesa y se da una salida automática, permite tener una respuesta oportuna y confiable para el usuario o sistema que necesita de esta solución.

Cabe recalcar que el tiempo de respuesta para el procesamiento de un flujo de datos es casi en tiempo real Shilpi & Saurabh (2001) afirma:

Una de las verdades más grandes sobre el análisis en tiempo real es que nada es en realidad en tiempo real; es un mito. En realidad, está cerca del tiempo real. Dependiendo del rendimiento y la capacidad de una solución y la reducción de las latencias operativas, los análisis podrían estar cerca del tiempo real, pero, mientras que día a día estamos cerrando la brecha entre el tiempo real y el tiempo casi real, es prácticamente imposible eliminar la brecha debido a las latencias computacionales, operacionales y de red.

Para que los datos puedan ser procesados deben pasar por algunas fases, se denominan “Fases de procesamiento de datos”, son las siguientes:

**Entrada de datos:** Los datos sin procesar comienzan a tomar forma de información utilizable, la mayoría de las tareas de entrada de datos consumen mucho tiempo, sin embargo, la entrada de datos se considera una tarea básica y necesaria para la mayoría de las organizaciones.

**Preparación de datos:** La preparación de los datos permite limpiar y organizar los datos para que los mismos sea información entendible y generen valor. Los datos sin procesar se verifican diligentemente para detectar cualquier error que pueda surgir, el propósito es eliminar datos que estén incorrectos o que no estén acordes con la información (datos redundantes, incorrectos e incompletos) y comenzar a crear datos de alta calidad.

**Proceso:** Se ejecutan las operaciones precisas para cambiar los datos en información importante, se realiza la operación de salida que emite varios métodos de presentación, se tomará como base para la toma de decisiones de una organización.

**Salida:** En el procesamiento de datos se proyecta como actividad extra, la administración de resultados de salida, es decir se describe como los procesos obligatorios para que la información importante llegue al usuario.

#### **Evolución del Procesamiento de Datos para llegar a Rapids**

La invención de las computadoras es una clara necesidad de información y procesamiento de datos, los informáticos tuvieron que escribir programas personalizados para procesar datos y estos probablemente se almacenaban en una tarjeta perforadora.

A través de los últimos años, han existido grandes avances tecnológicos que han llevado a un aumento dramático en el tráfico de información. Por ejemplo, las redes móviles han aumentado cobertura, rendimiento de datos y los teléfonos fijos se están actualizando lentamente de cobre a fibra óptica.

El gran volumen de datos ha introducido nuevos desafíos en la captura, almacenamiento, análisis, búsqueda, uso compartido, transferencia, visualización, consulta, actualización y privacidad de toda la información que es almacenada como información aérea, imágenes, sonido, datos meteorológicos, etc.

##### **Distributed Storage**

Los sistemas de almacenamiento distribuido proporcionan acceso confiable a los datos a través de la redundancia distribuida en nodos que son confiables individualmente.

El almacenamiento distribuido se centra en dos aspectos importantes *Hadoop* y *MapReduce,* permiten el almacenamiento distribuido y el procesamiento en múltiples máquinas.

Hadoop:Fue creado por *Doug Cuttin*, creador de *Apache Lucene[[6]](#footnote-6)*, la biblioteca de búsqueda de texto ampliamente utilizada:

Hadoop tiene su principal origen en Apache Nutch, este es un motor de búsqueda web de código abierto, que forma parte del proyecto Lucene. El creador del proyecto Doug, explica cómo surgió el nombre Hadoop, se detalla de la siguiente manera.

El nombre que mi hijo le dio a un elefante amarillo de peluche. Corto, relativamente fácil de deletrear y pronuncia, sin sentido, y no se usa en otro lugar: esos son mis criterios de denominación. Los niños son buenos para generar tales nombres (White, 2015).

Hadoop está siendo utilizada por bastantes compañías como *Yahoo!, Last.fm, Facebook* y el *New York Times.* En abril del 2008 Hadoop rompió un récord mundial al convertirse en el sistema más rápido para clasificar terabyte completo de datos Turner (2011) afirma lo siguiente:

La tecnología subyacente fue inventada por Google en sus primeros días para que pudiera indexar últimamente toda la rica información textural y estructural que estaban recopilando y luego presentar resultados provechosos, significativos y procesables a los usuarios. No había nada en el mercado que pudiera dejar hacer eso, así construyeron su propia plataforma, las innovaciones de Google se incorporaron a *Nutch[[7]](#footnote-7)*, un proyecto de código abierto y Hadoop luego se separó de esto. Hadoop fue diseñada para resolver problemas en los que tiene muchos datos, tal vez una mezcla de datos complejos y estructurados y no encaja bien en las tablas, para estas situaciones donde quieres que se ejecute un respectivo análisis, el cual debe ser profundo y computacionalmente extensivos, como la agrupación y la orientación.

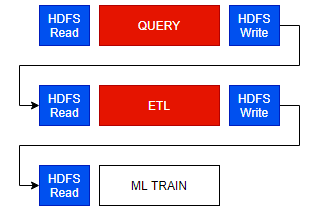
Hadoop está diseñado para trabajar y ejecutarse en una gran cantidad de máquinas que no comparten memoria ni discos. Significa que pueden comprar un montón de servidores básicos, colocarlos en un rack (Los armarios rack permiten, en un espacio reducido, contar con todo conectado: computador, *UPS[[8]](#footnote-8)*, teléfono o cualquier otra herramienta que se necesita. Para Turner (2011) esto se han convertido en una parte imprescindible en empresas ya que son lugares donde se requiere tener unas instalaciones seguras y de prestaciones adecuadas.

Turner (2011) pronuncia lo siguiente: “Ejecutar Hadoop en cada uno, lo que realiza es romper los datos en pedazos a través de los diferentes servidores, también realiza un respectivo seguimiento de donde residen los datos, además, los datos se alamacenan en un servidor que se desconecta o mueren pueden replicarse automáticamente desde una copia valida conocida”

Hadoop puede ejecutar su trabajo de indexación enviando su código a cada uno de los servidores y cada servidor opera en su propia pequeña porción de datos. Dentro de los resultados esperados, lo que hace Hadoop es devolver todos los resultados en un todo unificado (general).

**Figura 5**

Procesamiento Hadoop, leyendo desde el disco



*Nota.* Adaptado de *Procesamiento Hadoop, leyendo desde el disco* [Gráfico], por Srinath & Kraus, 2019.

La figura 5 representa la interpretación de Hadoop, *HDFS Read[[9]](#footnote-9)* son archivos de lectura, pasan a *Query[[10]](#footnote-10)* interpretadas como una cadena de consulta en las respectivas bases de datos, siguiendo con el esquema va a *HDFS Write[[11]](#footnote-11)* archivos de escritura*;* HDFS Read nuevamente archivos que necesitan ser leídos, esos archivos de lectura pasan a *ETL* (Extracción, transformación y carga) realiza un proceso de reformateo, limpiarlos y cargarlos en otra base de datos para después analizarlos y nuevamente tener archivos de escritura; HDFS Read pasan a *ML Train*, consiste en un proceso de entrenamiento de un modelo, implica utilizar algoritmos de aprendizaje con los respectivos datos para obtener predicciones sobre nuevos datos.

Para Howard et al. (2010) MapReducese ha convertido en una de las plataformas de computación paralela para procesar datos en escala de *terabytes* y *petabytes*. Utilizando diariamente en empresas como Yahoo!, Google, Amazon y Facebook y adaptado por varias universidades, permite una paralelización fácil de cálculos intensivos de datos en muchas maquinas. Una principal característica que tiene MapReduce es que intercala computación secuencial y paralela.

MapReduce es un modelo de programación para el procesamiento de datos. Los programas MapReducetrabajan en paralelo, por lo que ponen análisis de datos a gran escala en manos de cualquier persona con suficientes maquinas a su disposición. El funcionamiento de MapReduce divide el procesamiento en dos fases, la fase de mapa y la fase de reducción.

##### **Spark In-Memory Processing**

Sparkes el primer paradigma de computación distribuido y de propósito general, debido a su velocidad y adaptabilidad. Spark logra velocidad a través de un modelo de datos llamado *RDD[[12]](#footnote-12)* se almacenan en la memoria mientras se calculan eliminando así costosas escrituras intermedias en el disco. Además, consta con un motor de ejecución de *DAG[[13]](#footnote-13)*, puede optimizar la computación, particularmente la computación iterativa, es esencial para las tareas teóricas de datos como la optimización y el aprendizaje automático.

En *Apache Spark* el cálculo en memoria define un lugar de almacenar datos en algunas unidades de disco lento, los datos se guardan en la memoria de acceso aleatorio (RAM). Los datos que han sido almacenados se procesan en paralelo, al utilizar el procesamiento en memoria, se destaca un patrón, analizar datos de gran tamaño y también reduce el coste de la memoria, está conformado por dos principales cálculos en memoria los cuales son: Almacenamiento en RAM y Procesamiento distribuido en Paralelo.

Apache Spark es una plataforma de computación en *clúster* que, proporcionada una API para programación distribuida similar a MapReduce, pero está diseñado para ser veloz realizar consultas y algoritmos interactivos. Cabe recalcar queSparkfue diseñado desde cero para admitir aplicaciones de Big Data y ciencia de datos en particular, evitando la recarga de datos entre iteraciones.

**Figura 6**

Spark procesamiento en memoria



*Nota.* Adaptado de *Spark procesamiento en memoria* [Gráfico], por Srinath & Kraus, 2019.

La figura 6 representa el proceso de trabajo de Spark, HDFS Readlee el archivo, pasa a Query interpretado como una cadena de consulta en las respectivas BD , pasan por ETL es un proceso de reformateo, limpieza y carga, para después analizarlos y guardarlos en memoria cache, terminado el proceso de ETL se enfoca en ML, utilizar algoritmos de aprendizaje con el fin de obtener predicciones futuras, mediante Spark procesamiento en memoria se obtiene menos código, lenguaje flexible y principalmente en memoria.

##### **Gpu-Accelerated Computer**

La computación acelerada por GPU es el empleo de una unidad de procesamiento gráfico, juntamente con una unidad de procesamiento de computadora, para facilitar las respectivas operaciones de procesamiento intensivo como: aprendizaje profundo, análisis y aplicaciones de ingeniería, la GPU hace que los programas o aplicaciones sean más rápidas.

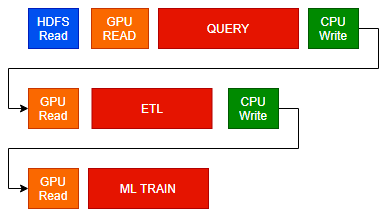
Las tecnologías de computación acelerada como GPU reciben una atención considerable en la *HPC* (Computación de alto rendimiento) moderna. En comparación con los aceleradores clásicos y las CPU tradicionales, la computación acelerada exhibe una mayor densidad de cómputo, sino que también poseen un ancho de banda de memoria significativo y capacidades similares a las de los vectores para transmitir datos.

Por supuesto la aceleración a través de hardware especializado o procesadores no es nueva: varias áreas de aplicación como gráficos, multimedia, redes, criptografía, etc., generalmente se acelera en *PC[[14]](#footnote-14)* y dispositivos integrados (Matsuoka et al., 2009).

Matsuoka et al.(2009) afirma: “Existen tres desafíos clave en el uso de GPU para aplicaciones de uso intensivo de datos, el cual los clasifica de la siguiente manera: Ejecución, Tubería de Transmisión, Coalescencia de Memoria.”

**Figura 7**

Procesamiento tradicional GPU



*Nota.* Adaptado de *Procesamiento tradicional GPU* [Gráfico], por Srinath & Kraus, 2019.

La figura 7 representa el proceso de trabajo de Procesamiento tradicional de GPU, HDFS Read se lee el archivo pasan a GPU *Read[[15]](#footnote-15)*, la GPU realiza múltiples tareas en paralelo, pasan por Query que son interpretadas como una cadena de consulta en las respectivas BD, además, *CPU* *Write[[16]](#footnote-16)* se encarga de realizar los cálculos secuenciales, es aquí cuando trabajan conjuntamente la GPU y CPU; nuevamente se aplica GPU Read para realizar los cálculos complicados y pasa al proceso de ETL realiza el proceso de reformateo, CPU Write realiza el cálculo secuencial, lee la GPU Read y pasan al proceso de ML que implica utilizar algoritmos de aprendizaje a los datos. En la GPU existe más código, un lenguaje rígido y sustancialmente debe realizarse las principales tareas en GPU.

Cabe mencionar que en la sección 2.2 se enfocara en el funcionamiento, diferencias entre GPU y CPU y las principales arquitecturas.

##### **Rapids**

Rapids es el punto fuerte de esta temática, se enfoca en escribir en memoria del sistema, donde se mantendrá cosas en la GPU siempre como sea posible, tiene formatos comunes que permite obtener un ecosistema de computación acelerado.

**Figura 8**

Rapids



*Nota.* Adaptado de *Rapids* [Gráfico], por Srinath & Kraus, 2019.

En la figura 8 representa el proceso de trabajo de *rapids*, obtiene resultados beneficiosos, permite una simplificación significativa y obtiene algunas características principales como: mismo código, lenguaje flexible y principalmente en GPU; comienza con *Arrow* *Read*, después pasa por Queryque son interpretadas como una cadena de consulta en las respectivas bases de datos y pasan a ETL (Extracción, transformación y carga), realiza un proceso de reformateo, limpiarlos y cargarlos, por último, realizanMLTrain*,* enfocado en realizar un proceso de entrenamiento de un modelo, implica utilizar algoritmos de aprendizaje con los respectivos datos para obtener predicciones sobre nuevos datos. La librería rapids ya simplifica algunos pasos que han sido mencionados anteriormente, rapids es una librería de computación acelerada.

## **Gpu**

Se produjo la evolución en la informática de masas, en el mundo entero de la computación profesional la empresa *Sillicon Graphics* dedico muchos esfuerzos durante la década de 1980 a desarrollar soluciones orientadas a gráficos tridimensionales, con el fin de obtener una excelente visualización en cuanto a entornos gráficos*. Sillicon Graphics* popularizó el uso de tecnologías para *3D[[17]](#footnote-17)* en diferentes sectores como el gubernamental, defensa, visualización científica y técnica, además, de proporcionar herramientas para crear efectos cinematográficos nunca vistos.

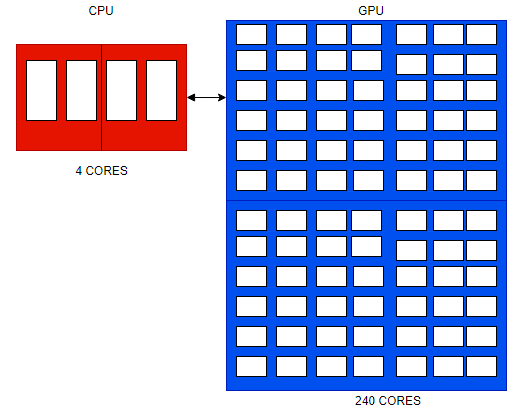
Guim & Rodero (2019) afirma que a mediados de la década de 1990, la demanda de gráficos 3D por parte de los usuarios aumento vertiginosamente a partir de la aparición de juegos inversivos en primera persona, como *Doom, Duke Nukem 3D*, acercaban al sector de los videojuegos para ordenadores personales para entornos 3D. Al mismo tiempo, empresas como *Nvidia*, *ATI Technologies* y *3dfx* *Interactive[[18]](#footnote-18)* empezaron a comercializar aceleradores gráficos que era suficientemente económicos para los mercados de gran consumo.

GPU se ha convertido en una parte integral de los principales sistemas informáticos actuales (Owens et al., 2008) afirma lo siguiente:

En los últimos seis años, ha habido un mercado en aumento para el rendimiento y las capacidades de las GPU. La GPU moderna no es solo un potente motor de gráficos (ver figura 9), es un procesador programable altamente paralelo con un ancho de banda de memoria y aritmética que supera a su contraparte de CPU. Además, es un procesador de propósito específico, permite optimizar grandes cantidades de datos y realizar las mismas operaciones una y otra vez.

**Figura 9**

Modelo de computación heterogéneo CPU + GPU



*Nota.* Adaptado de *Modelo de computación heterogéneo CPU + GPU* [Gráfico], Pérez Represa et al., 2016.

La figura 9 demuestra que la GPU contiene tarjetas gráficas y realiza cálculos científicos de propósito general, el modelo de computación sobre tarjetas gráficas es usar conjuntamente una CPU y una GPU de manera que formen un modelo de computación heterogéneo Owens et al. (2008) afirma lo siguiente com respecto al trabajo de la GPU:

Requisitos computacionales son grandes: El renderizado en tiempo real requiere de miles de millones de pixeles por segundo y cada pixel requiere cientos o más operaciones. La GPU debe entregar una enorme cantidad de rendimiento informático para satisfacer la demanda de aplicaciones complejas en tiempo real. Paralelismo Sustancial: La canalización de gráficos es adecuada para el paralelismo. Operaciones en los vértices y fragmentos se adaptan bien a unidad de cómputo programables paralelas de grano fino y estrechamente acopladas, que a su vez son aplicables a muchos otros dominios computacionales. Rendimiento más importante que la latencia: Las implementaciones de GPU de la tubería de gráficos priorizaron el rendimiento sobre la latencia. El sistema visual humano opera en escalas de tiempo de milisegundos, mientras que las operaciones dentro de un procesador moderno toman nanosegundos.

### *Gpu vs Cpu*

La CPU está diseñada para el procesamiento en serie, se compone de pocos núcleos muy complejos que pueden ejecutar unos pocos programas al mismo tiempo; GPU tiene cientos o miles de núcleos sencillos que pueden ejecutar cientos de miles de programas específicos a la vez.

La GPU requieren un alto grado de paralelismo tradicionalmente una instrucción y múltiples datos, la GPU se vuelve más apta en el trabajo de procesamiento en paralelo, es por esto que se diseño para satisfacer los procesos informáticos, que necesitan de un procesamiento ligeramente rápido.

Guim & Rodero (2019) pronunica distintas diferencias entre el rendimiento que tiene la CPU contra el rendimiento que tiene la GPU:

Las diferencias en CPU y GPU se deben principalmente a una cuestión de filosofía de diseño. Mientras que las GPU están pensadas para explotar el paralelismo a nivel de datos con el paralelismo masivo y una lógica bastante simple, el diseño de una CPU esta optimizada para la ejecución eficiente de código secuencial. Las CPU utilizan lógica de control que permite un paralelismo a nivel de instrucción y utilizan memorias cache bastante grandes para reducir el tiempo de acceso a los datos. También hay otras cuestiones, como el consumo eléctrico o el ancho de banda de acceso a la memoria. Las GPU actuales tienen anchos de banda a memoria en torno a diez veces superiores a los de la CPU, entre otras cosas porque las CPU deben satisfacer requisitos heredados de los sistemas operativos, de las aplicaciones o dispositivos de entrada y salida.

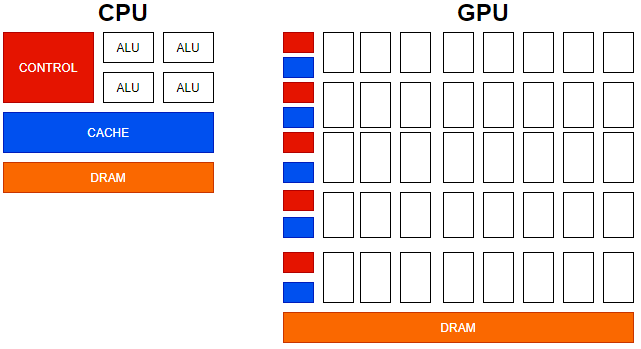
Por medio de la GPU ha existido una evolución inmensa y muy rápida desde el punto de vista de la programación de las GPU, han cambiado el propósito de los dispositivos; desde el año 2000 las GPUs utilizaban unidades aritméticas programables para devolver el color de cada píxel de a pantalla.

Guim & Rodero (2019) afirma un concepto importante com respectoa a configuraci :

“Las apariciones aritméticas que se aplican a los colores de entrada y texturas las podía controlar completamente el programador, los investigadores observaron que los colores de entrada podían ser cualquier tipo de dato. Los datos de entrada eran datos numéricos que tenían algún significado más allá de un color, los programadores podían ejecutar cualquiera de los cálculos que necesitaban sobre esos datos de entrada.”

**Figura 10**

Comparativa de memoria y lógica de control para CPU y GPU

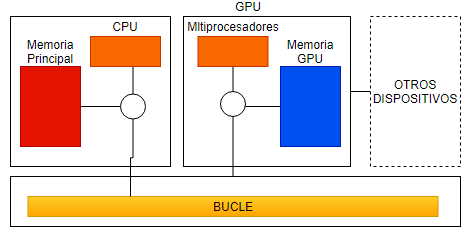


*Nota.* Adaptado de *Comparativa de memoria y lógica de control para CPU y GPU* [Gráfico], por Guim & Rodero, 2019.

En la figura 10 se demuestra la comparativa de CPU y GPU, la GPU requiere de menos lógica de control para cada flujo de ejecución. Al mismo tiempo dispone de una pequeña memoria cache, permite que los flujos compartan memoria y tengan el respectivo ancho de banda.

**Figura 11**

Interconexión entre CPU y GPU mediante PCle



*Nota.* Adaptado de *Interconexión entre CPU y GPU mediante PCle* [Gráfico], por Guim & Rodero, 2019.

Guim & Rodero (2019) afirma: “La figura 11 es la comunicación entre CPU y GPU se lleva a cabo por medio de un puerto dedicado. En la actualidad el *PCI Exprres* o *PCIe* (*peripheral component interconnect express*) es el estándar para ejecutar esta comunicación”

Hay que tener en cuenta que una GPU no puede acceder a la memoria principal, es decir directamente, en cambio la CPU no puede acceder directamente a la memoria de una *Gpu,* por lo tanto, hay que copiar los datos entre GPU y CPU de manera explícita, es decir en ambas direcciones.

### *Arquitecturas Gpu*

La arquitectura GPU durante las primeras generaciones tenían una cantidad de núcleos bastante reducida, ámbitos de consultas y dedicación por parte de los desarrolladores rápidamente se incrementó hasta hoy en día, cuando hablamos de dispositivos de tipo *many-core[[19]](#footnote-19)* con centenares de núcleos en un único chip.

Las arquitecturas GPU es una de las tendencias en computación paralela con más crecimiento en los últimos años y que goza de gran popularidad, entre otras cuestiones, debido a la buena relación entre las prestaciones que ofrece y su coste.

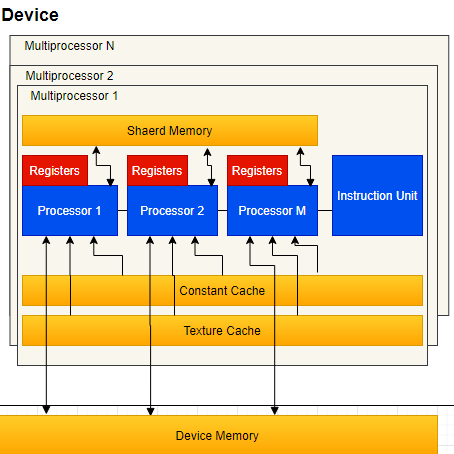
La GPU siempre ha sido un procesador con amplios recursos computacionales, ha evolucionado desde una función fija, a un procesador de propósito especial en un procesador programable paralelo con funcionalidad adicional y función de propósito especial.

En general, una arquitectura GPU está hecha de un número escalable de multiprocesadores de transmisión y contiene varios núcleos de procesamiento en paralelo (procesadores de flujo, *SP*), planificadores de deformación, unidades de despacho, unidades de funciones especiales (*SFU[[20]](#footnote-20)*), memoria local, memoria compartida, memoria de textura, L1, *Caches* L2 y cache constante.

Una placa GPU tiene un ancho de banda alto, memoria global sin chip, conocida como *DDR[[21]](#footnote-21)* gráfica, hasta varios *gigabytes* de tamaño. Se anuncia que la GPU incluye una memoria global de 12 GB. Una memoria compartida en el chip proporciona alta velocidad de acceso a la memoria tan rápido como un registro y una parte de la memoria compartida se puede configurar manualmente en un Cache L1. La memoria constante como la memoria de textura es colocada en una memoria global, pero se almacenan en una cache (Thi Yen et al., 2017).

**Figura 12**

Arquitectura básica de GPU



*Nota.* Adaptado de *Arquitectura básica de GPU* [Gráfico], por Huang et al., 2013.

En la figura 12 representa la arquitectura GPU, tiene su propia memoria llamada memoria global. A excepción de la memoria global, la GPU tiene caches especiales como cache constante, cache textura y memoria compartida. Dentro de la GPU existen varios multiprocesadores y dentro del multiprocesador hay varios procesadores de flujo. El programa ejecutado por GPU se divide en dos niveles de paralelismo. El primer es el paralelismo grueso con respecto a los bloques controlados por multiprocesadores. El segundo es el paralelismo final con respecto a los subprocesos ejecutados en procesadores de flujo por cada bloque, se puede asignar una memoria especial llamada memoria compartida, es rápida y limitada (Huang et al., 2013).

A continuación, en las secciones siguientes se detallará las arquitecturas que utilizan GPU, las cuales desde un punto de vista científico y experimental de desarrollo son muy primordiales:

#### **Arquitectura Nvidia GeForce**

La Arquitectura *Nvidia* *GeForce* brinda diferentes productos, se dividen en tres fases principales:

* GeForce: Enfocada a un mercado de consumo multimedia como videojuegos, edición de video, fotografía digital.
* Cuadro: Enfocada para soluciones profesionales las cuales requieren modelos *3D*, como por ejemplo en el ambiente de ingeniería y arquitectura.
* Tesla: Enfocada para la computación de altas prestaciones, como por ejemplo el procesamiento de información sísmica, simulaciones bioquímicas, modelos meteorológicos y cambio climático, computación financiera y análisis de datos.

Este apartado se enfoca en la serie Nvidia GeForce, es de uso de GPU pensada para tratamiento de gráficos. A pesar de que no es la arquitectura muy actual nos permitirá interpretar y conocer de mejor manera cómo funciona (Guim & Rodero, 2019).

**Figura 13**

Esquema de la Arquitectura de la GPU GeForce de Nvidia



*Nota.* Adaptado de *Esquema de la Arquitectura de la GPU GeForce de Nvidia* [Gráfico], por Guim & Rodero, 2019.

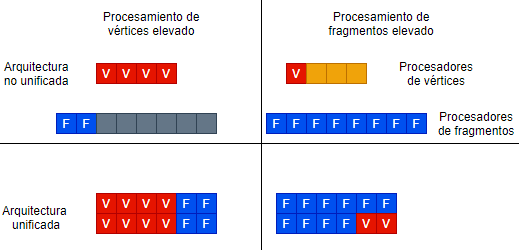
En la figura 13 se envía tres tipos de datos, estos son: instrucciones, texturas y vértices. Los vértices se encargan de aplicar un programa, tiene que ser específico porque ejecuta transformaciones sobre cada uno de los vértices de entrada. GeForce es el primer programa ejecutado en el procesador de los vértices.

#### **Arquitectura Unificada**

En las arquitecturas unificadas no existe la división a nivel de hardware entre procesadores de vértices y procesadores de fragmentos, son capaces de trabajar a nivel de vértice como a nivel de fragmento, sin estar especializada en un tipo concreto. Con arquitecturas unificadas, no hay partes específicas del chip asociadas a una etapa concreta del pipeline, sino un único tipo de unidad de ejecutar todas las operaciones, sea cual sea su naturaleza. También ofrecen un mayor potencial para hacer computación de propósito general (Guim & Rodero, 2019).

**Figura 14**

Comparativa de asignación de procesadores de una GPU en el procesamiento de vértices y de fragmentos en arquitecturas unificadas y no unificadas



*Nota.* Adaptado de *Comparativa de asignación de processadores de una GPU en el procesamiento de vértices y de fragmentos en arquitecturas unificadas y no unificadas* [Gráfico], por Guim & Rodero, 2019.

La figura 14 muestra la respectiva ejecución de dos aplicaciones no balanceadas y como la arquitectura unificada puede ofrecer una solución más eficiente y factible.

#### **Arquitectura orientadas a computación de propósito general sobre Gpu**

Los procesadores gráficos pueden ser utilizados para ejecutar aplicaciones que son tradicionalmente ejecutados en CPU. La computación de propósito general se conoce como *GPGPU* (*general-purpose computing on Graphics processing unit*) (Guim & Rodero, 2019).

La GPGPU proporciona una mejora en términos de *speedup[[22]](#footnote-22)* de la ejecución de aplicaciones asociadas, estas aplicaciones pueden ser adaptadas a la GPU, se pueden destacar algunas como: Algebra lineal, procesamiento de imágenes, procesamiento de consulta de bases de datos, etc. El principal argumento de GPGPU es utilizar el procesador de fragmentos como una unidad de cómputo, también hay que tener en cuenta que la entrada y salida es limitada, es decir, pueden hacer lecturas arbitrariamente, pero hay restricciones para las escrituras (Guim & Rodero, 2019).

En GPU la computación se realiza normalmente por un flujo de procesador de fragmentos, tendrán que ejecutarse en las unidades funcionales de la GPU correspondientes. Se establece un símil entre el programa en CPU y la rasterización en GPU. La rasterización determina los pixeles de flujo de datos, estos se ven afectados por primitivas generadas a partir de los vértices y generan un fragmento para cada uno (Guim & Rodero, 2019).

#### **Arquitectura Nvidia**

Nvidia permite analizar enormes cantidades de datos y realizar predicciones comerciales precisas a una velocidad sin precedentes.

Nvidia a finales del año 2006 presento un hardware orientado a computación general de prestaciones llamada *Tesla*, esta se empezó a desarrollar a mediados del año 2002, tesla ofrece un hardware de prestaciones muy altas, por ejemplo, bloques de procesadores gráficos. Esta sección se centrará en la arquitectura *G80* (ver figura 15), la arquitectura *G80* representa un importante salto tecnológico por el hecho de implementar una arquitectura unificada (Guim & Rodero, 2019).

**Figura 15**

Esquema de arquitectura G80 de Nvidia



*Nota.* Adaptado de *Esquema de arquitectura G80 de Nvidia* [Gráfico], por Guim & Rodero, 2019.

La figura 15 representa la arquitectura *G80* de Nvidia es una arquitectura unificada en toda su totalidad, se orienta en la ejecución masiva de flujos y se ejecuta en un estándar *IEEE[[23]](#footnote-23)* 754 (Guim & Rodero, 2019).

La arquitectura *G80* de Nvidia presento una mejora en el procesamiento gráfico e incremento de prestaciones, la clave principal es la mejora de la capacidad de cálculo. El funcionamiento básico de la arquitectura G80 para ejecutar un programa es: Los datos de entrada se procesan mediante un hardware especializado, se encarga de distribuir datos que puedan ser utilizados al máximo número de unidades para obtener la capacidad máxima de cálculo durante la ejecución del programa. También un controlador de flujos se encarga de controlar la ejecución de flujos que ejecutan cálculos de manera coordinada (Guim & Rodero, 2019).

### *Funcionamiento Gpu*

Las GPU suelen tener grandes cantidades de núcleos de procesamiento a frecuencias de reloj relativamente bajas. Se basa en las primitivas Nvidia Cuda (ver figura 16) para la optimización de cómputo de bajo nivel, pero expone ese paralelismo de GPU y la velocidad de la memoria de alto ancho de banda a través de interfaces *Python* fáciles de usar.

Nvidia Cuda[[24]](#footnote-24) es como un mini procesador que se encarga de cierto tipo de instrucciones, suelen poder ejecutarse de manera paralela, incluye un compilador, también un conjunto de herramientas de desarrollo por Nvidia. Además, proporciona un entorno de desarrollo para crear aplicaciones aceleradas por GPU de alto rendimento.

#### **Cuda**

Cuda permite explotar todas las ventajas de la GPU utilizando el paralelismo, realiza un sin número de cálculos complejos que se implementan en cantidades enormes de núcleos en las GPUs.

Cuda tiene la posibilidad de paralelizar operaciones de computación mucho más avanzadas, llevan tiempo empleándose en máquinas de tipo *Deep* *Learning[[25]](#footnote-25)* para inteligencia artificial, procesa millones de datos por ciclo de reloj y para la obtención de resultados en tiempo real o el menor tiempo posible dependiendo de la tarea asignada.

Las bibliotecas Cuda aceleradas por GPU permiten la aceleración directa en múltiples dominios, tenemos: Algebra lineal, procesamiento de imágenes y videos, aprendizaje profundo y análisis de gráficos.

Dentro del desarrollo de algoritmos personalizados, puede usar integraciones disponibles con idiomas y paquetes numéricos de uso común, así como API de desarrollo bien publicadas.

Las aplicaciones hechas en Cuda se implementan en todas las familias de GPU Nvidia disponibles en las instalaciones y en instancias de GPU en la nube, utilizan capacidades integradas para distribuir cálculos a través de configuraciones múltiples GPU, los científicos e investigadores pueden desarrollar aplicaciones que escalen desde estaciones de trabajo de GPU como también instalaciones en la nube con miles GPU.

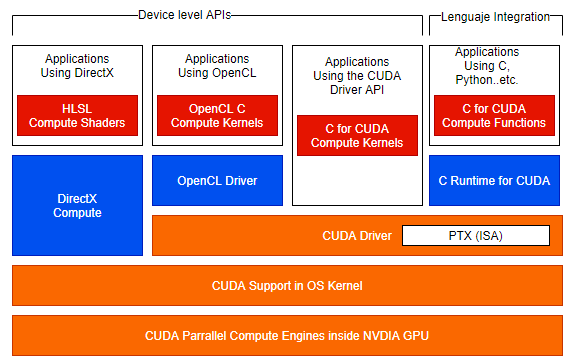
#### **Arquitectura Cuda**

La arquitectura Cuda es de computación paralela y revolucionaria ya que ofrece el rendimiento de la mundialmente conocida como tecnología de procesamiento gráfico de Nvidia para aplicaciones de uso general de cálculo en la GPU.

Las aplicaciones que son ejecutadas en una arquitectura GPU aprovechan una base instalada de cien millones de GPUs habilitadas para Cuda en ordenadores de sobremesa y pórtatiles, estaciones de trabajo profesionales y clusters de superordenadores (NVIDIA, 2009).

**Figura 16**

Arquitectura Cuda



*Nota.* Adaptado de *Arquitectura Cuda* [Gráfico], por NVIDIA, 2009.

En la figura 16 representa la arquitectura Cuda consta de algunos componentes principales, son:

* Motores de cálculo paralelo de las GPUs.
* Soporte a nivel de kernel del Sistema Operativo.
* Controlador en modo Usuario, y esta proporciona una API*[[26]](#footnote-26)* en nivel de dipositivo.
* Arquitectura de conjuntos de instrucciones para núcleos y funcionaes de cálculo paralelo.

Pérez Represa et al (2016) afirma:

En la arquitectura clásica de una tarjeta gráfica podemos encontrar la presencia de dos tipos de procesadores, los procesadores de vértices y los procesadores de fragmentos, dedicados a tareas distintas e independientes dentro del cauce gráfico y con repertorios de instrucciones diferentes.

Esto presenta dos problemas importantes, por un lado, el desequilibrio de carga que aparece entre ambos procesadores y por otro la diferencia entre sus respectivos repertorios de instrucciones. De este modo, la evolución natural en la arquitectura de una GPU ha sido la búsqueda de una arquitectura unificada donde no se distinguiera entre ambos tipos de procesadores.

A continuación, en la tabla 1 se refleja los respectivos núcleos de procesamiento de un dispostivo cuda en función de su capacidad de cómputo.

**Tabla 1**

*Núcleos de procesamiento de un dispositivo Cuda*

|  |  |  |
| --- | --- | --- |
| Capacidad de Cómputo | Nombre de la Arquitectura | Núcleos por multiprocesadór |
| 1.x | *Tesla* | 8 |
| 2.0 | *Fermi* | 32 |
| 2.1 | *Fermi* | 48 |
| 3.x | *Kepler* | 192 |
| 5.x | *Maxwell* | 128 |

Nota: Se indica el nombre de las diferentes arquitecturas así como también el número de núcleos de cómputo o Streaming Processors características de cada una de ellas en función de la capacidad de cómputo (Pérez Represa et al., 2016).

Dentro de la terminología de Cuda recibe el nombre de capacidad de cómputo, se indica mediante dos números de la forma M.m, estos representan la revisión mayor y la revisión menor de la arquitectura del dispositivo (Pérez Represa et al., 2016).

#### **Ventajas y Desventajas Cuda**

Cuda presenta algunas ventajas importantes frente a otros tipos de computación en GPU utilizando APIs gráficas, las principales son:

* Realizar lecturas disperas: consultar cualquier posición de memoria.
* Memoria compartida: disposición del programador en un área de memoria que sea compartida entre hilos. Dado su tamaño y rapidez pueden ser utilizados como cache (El et al., 2015).
* Permite lecturas más rápidas hacia la GPU.
* Ofrece soporte para enteros y operadores a nivel de bit.

Cuda sin embargo presenta algunas desventajas, son:

* No se puede utilizar recursividad, punteros o funciones, variables estáticas dentro de funciones o funciones con numero de parámetro de variable (El et al., 2015).
* No soporta el renderizado de texturas.
* No soporta números desnormalizados.
* Puede existir un cuello de botella entre la CPU y la GPU por los anchos de banda de los buses y sus latencias. (El et al., 2015).
* Los hilos deben lanzarse en grupos de al menos 32.

## **Librería Open Source Rapids**

La biblioteca de código abierto Rapids brinda la capacidad de ejecutar canalizaciones de ciencia de datos y análisis de extremo a extremo en GPU.

Rapidsse enfoca en tareas comunes de preparación de datos para análisis y ciencia de datos. Incluye una API de trama de datos familiar que se integra con una variedad de algoritmos de aprendizaje automático para aceleraciones de canalización de extremo a extremo sin pagar los costos de serialización típicos.

Rapids permite un procesamiento y capacitación enormemente acelerados en tamaños de conjuntos de datos mucho más grandes, brinda a los científicos de datos un enorme salto de rendimiento para resolver los desafíos comerciales más complejos como predecir fraudes con tarjetas de créditos, pronosticar inventarios de venta minorista y comprender el comportamiento de los clientes, etc.

Rapids comenzó a partir de los proyectos *Apache Arrow****,* una plataforma de desarrollo en varios idiomas para datos en memoria**y *GoAi[[27]](#footnote-27)* basados ​​en una estructura de datos columnar en memoria, ofrece un intercambio de datos eficiente y rápido con flexibilidad para soportar modelos de datos complejos (RAPIDS Development Team, 2018).

El principal objetivo de la librería Rapids no es solo acelerar las partes individuales del flujo de trabajo típico de la ciencia de datos, sino acelerar el flujo de trabajo completo de extremo a extremo, se mencionará algunas características muy importantes del porque acelerar el flujo de trabajo:

* Integración sin complicaciones:Acelera su cadena de herramientas de ciencia de datos de Python con cambios mínimos de código y sin nuevas herramientas para aprender.
* Precisión del modelo:Aumenta la precisión del modelo de aprendizaje automático iterando en los modelos más rápido y desplegándolos con mayor frecuencia.
* Código abierto:Personalizable, extensible, interoperable; el software de código abierto es compatible con Nvidia y se basa en Apache Arrow.
* Tiempo de entrenamiento reducido:Mejora drásticamente su productividad con ciencia de datos casi interactiva.

### *Ambiente de Trabajo Rapids*

El ambiente de trabajo de la librería Rapids expone de manera determinada como es el proceso de trabajo aplicando los diferentes puntos como *“Data Preparation”, “Model Training”* y *“Visualization”* y el proceso interno que hay detrás.

**Figura 17**

Ambiente de trabajo de Rapids



*Nota.* Adaptado de Ambiente de trabajo de Rapids [Gráfico], por Srinath & Kraus, 2019.

La figura 17 representa el trabajo de Rapids, cabe destacar que se menciona pricipales bibliotecas que nos permitira realizar la preparación, entrenamiento y visualización, estas bibliotecas serán explicadas de manera más detalladas en las secciones 2.3.3, 2.3.4, 2.3.5 y 2.3.6, además, hablar de Apache Arrow que sera explicado en la sección 2.3.2.

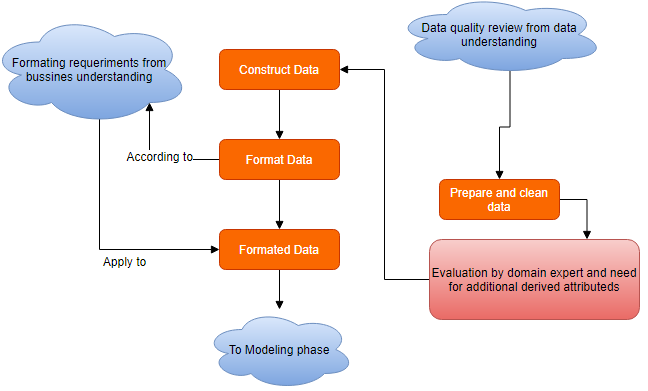
#### **Preparación de la Data**

La preparación de datos es la etapa fundamental del análisis de datos, existen demasiada información de baja calidad en varias fuentes de datos, web y diferentes organizaciones, están sumamente enfocadas en cómo transformar los datos en formularios limpios que puedan usarse con fines de alto beneficio.

El enfoque de la fase de preparación de datos es identificar datos de calidad y formatearlos adecuadamente, lo que puede conducir a la generación de patrones de calidad por los algoritmos de minería de datos elegidos. La preparación de datos genera un conjunto de datos más pequeño que el conjunto de datos originales, pero con mejor calidad y datos relevantes que pueden mejorar significativamente la eficiencia de la fase de modelado (Mansingh et al., 2017).

**Figura 18**

Fases de Preparación de Datos



*Nota.* Adaptado de *Fases de Preparación de Datos* [Gráfico], por Mansingh et al., 2017.

La figura 18 representa la principal interacción de preparación de datos y la fase de modelado, por lo tanto, mientras se preparan los datos, se debe realizar un modelado preliminar, es decir, realizar un modelo que permita determinar que la preparación de los datos queda de manera definitiva para después proceder a utilizar algoritmos Machine Learning.

Rapids se enfoca en la preparación de la data utilizando la biblioteca específica *cudf*, esta permitirá realizar un proceso de configuración a los datos que van a hacer analizados, por ende, realizará una integración, transformación y reducción, más adelante se detallará el funcionamiento de la biblioteca cudf.

#### **Machine Learning**

El entrenamiento de Machine Learning debe obtener una precisión predictiva con los distintos datos, debe ser esencial para los científicos de datos, estos se describen como técnicas básicas y avanzadas para entrenar un modelo, algunos puntos deben ser nombrados en este apartado:

* Entrenar
* Evaluar
* Usar la ingeniería de características para crear características que se utilicen para el entrenamiento del modelo.

Además, es la ciencia que nos permite que los computadores aprendan y actúen como humanos, de una forma autónoma, alimentándolas con información, datos en forma de observaciones e interacciones con el mundo real. Machine Learning utiliza algoritmos programados que reciben y analizan millones de datos para predecir valores de salida.

La aplicación de métodos de aprendizaje automático a grandes bases de datos se denomina minería de datos. La analogía es que un gran volumen de tierra y materia prima se extrae de una mina, que cuando se procesa conduce a una pequeña cantidad de material muy valioso; de manera similar, se procesa un gran volumen de datos para construir un modelo simple con uso valioso, por ejemplo, que tenga una alta precisión predictiva (Alpaydin, 2009).

Rapids entrena el modelo utilizando la biblioteca *cuml*, para proporcionar datos de entrenamiento para aprender de algoritmos de aprendizaje, estos datos previamente deben estar preparados y deben contener los datos ya clasificados y listos para aplicar el conjunto de los algoritmos de Machine Learning que tiene la biblioteca cuml*.*

##### **Tipos de Algoritmos Machine Learning**

Un algoritmo es una secuencia de instrucciones que deben llevarse a cabo para transformar la entrada en salida, los algoritmos aprenden y optimizan sus operaciones para mejorar el rendimiento, desarrollando “inteligencia” con el tiempo. Los algoritmos encuentran patrones en los datos de entrenamiento que asignan los atributos de los datos de entrada el destino y generan un modelo que captura dichos patrones.

Se necesita algoritmos eficientes para resolver el problema de optimización, así como para almacenar y procesar la gran cantidad de datos que generalmente se tiene. Una vez que se aprende un modelo, su representación y solución algorítmica (Alpaydin, 2009).

Se presenta un listado de los principales algoritmos que son utilizados en la actualidad y una breve descripción de cada uno de ellos.

* **Clasificación y Regresión:** Clasificación se enfoca en los procesos de identificar una nueva categoría de observación sobre la base de formación del conjunto de datos que contiene observaciones de las cuales las categorías son desconocidas. El resultado es una clase, entre un número limitado de clases, con clases nos referimos a categorías arbitrarias según el tipo de problema. Regresión consiste que, el resultado es un número, el resultado será un valor numérico dentro de un conjunto infinito de posibles resultados.
* **Inferencias Estadísticas:** Cada sistema a su vez necesita o requieren de un análisis estadístico simple o complejo, gran parte de los datos que son recopilados por organizaciones exhiben valores perdidos múltiples, interacciones complejas de datos.
* ***Clustering*:** El proceso clustering permite medir la similitud entre objetos, se suelen utilizar diferentes formas de distancia. Clustering es una técnica de aprendizaje automático, el aprendizaje realizado es no supervisado, además, juega un papel muy importante en aplicaciones de minería de datos, aplicaciones sobre bases de datos espaciales, aplicaciones web, marketing, diagnóstico médico, análisis de *ADN* en biología computacional, entre otras más (Garre et al., 2007).
* **Descomposición y Dimensión de reducción:** La reducción de dimensión es una metodología principalmente utilizada en distintos campos ligados al procesamiento de datos, representa una etapa de preproceso o ser un elemento esencial para la representación y clasificación de datos, obtener una nueva representación de los datos originales en un espacio de menor dimensión, de forma que se produzca información más depurada, reduzca el tiempo del procesado y genere representaciones visuales entendibles para el ser humano (Castro, 2018).

En la sección 2.3.4 abarcaremos más información de cuml*,* la biblioteca está conformada por un conjunto de algoritmos de Machine Learning, además, indicar el funcionamiento, conceptos y uso de esta biblioteca que pertenece a Rapids.

#### **Visualisación**

La visualización es una representación gráfica de datos, presenta los datos como una imagen o gráfico para facilitar la identificación de patrones y comprender conceptos difíciles, la tecnología permite que los usuarios interactúen con los datos cambiando los parámetros para ver más detalles y crear nuevas ideas (RAPIDS Development Team, 2018).

Rapids permite que se juegue con un enfoque muy dinámico con las bibliotecas de visualización de ciencia de datos. Para un excelente rendimiento aún mayor, para proporcionar capacidades de visualización de datos de alto rendimiento y alto *FPS[[28]](#footnote-28)*, incluso con conjuntos de datos muy grandes.

Los datos que son visualizados tienden a tener más valor cuando se procesan, analizan, este proceso lo realiza la biblioteca *cuxfilter*, se hablara de manera más específica en la sección 2.3.6; esto lo hace aún más valioso para qué es más fácil consumir información, que a su vez se convierte en conocimiento.

### *Gpu Memory Apache Arrow*

Apache Arrow es una plataforma en varios idiomas para datos en memoria; especifica un formato de memoria columnar estandarizado independiente del lenguaje para datos planos y jerárquicos, organizado para operaciones analíticas eficientes en hardware moderno. Proporciona bibliotecas computacionales y mensajes de transmisión de copia cero y comunicación entre procesos (Arrow, 2016).

Apache Arrow especifica un formato en memoria dirigido a grandes conjuntos de datos y bibliotecas para varios idiomas para interactuar con los datos.

Arrow evita la necesidad de serialización entre diferentes tiempos de ejecución del lenguaje y comunicación entre conjuntos de datos entre procesos sin copia (Peltenburg et al., 2019).

#### **Funcionamiento de Apache Arrow**

Los conjuntos de datos de Apache Arrow suelen ser tabulares y almacenados en una abstracción llamada *RecordBatch*. Un RecordBatch contiene varias columnas para cada campo de un registro, que están en Arrow llamadas matrices. Estas respectivas matrices contienen todo tipo de datos, desde cadenas hasta listas de enteros, listas de marcas de tiempos y otros. Las matrices consisten en varios buffers contiguos de Arrow, están relacionados para almacenar los datos de un tipo específico (B et al., 2019).

El funcionamiento de Apache Arrow, lo detallamos por las principales características, las siguientes son:

Rápido:Apache Arrow permite a los principales motores de ejecución aprovechar las últimas operaciones de *SIMD[[29]](#footnote-29)* incluidas en los procesadores modernos, para la optimización vectorizadas nativa al procesamiento de datos analíticos. El diseño en columnas está optimizado por la localidad de los distintos datos para un respectivo rendimiento en hardware moderno tanto para CPU y GPU (Arrow, 2016).

Flexible:Apache Arrow actúa como una interfaz de alto rendimiento entre varios sistemas, es decir, dentro de cada sistema Apache Arrow se visualiza como una interfaz nueva para cada sistema. También se centra en adquirir una amplia variedad de lenguajes de programación estándar. Las implementaciones de *C, C ++, C#, Go, Java, JavaScript, MATLAB, Python, R, Ruby y Rust* están en progreso y se aceptan más idiomas (Arrow, 2016).

**Figura 19**

Sin Apache Arrow



*Nota.* Adaptado de *Sin Apache Arrow* [Gráfico], por Arrow, 2016.

La figura 19 representa la no utilización de Apache Arrow, por ello existen un flujo muy extenso de copiar y convertir datos en diferentes sistemas, cada sistema tiene su propio formato de memoria interna, un 70 a 80 % de cálculo desperdiciado en serialización y deserialización, hay un cargo excesivo de flujo de los datos, donde se pierde tiempo.

**Figura 20**

Con Apache Arrow



*Nota.* Adaptado de *Con Apache Arrow* [Gráfico], por Srinath & Kraus, 2019.

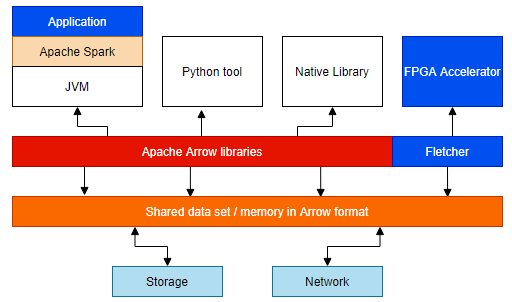
La figura 20 representa la utilización de Apache Arrow*,* se demuestra de manera específica y eficiente que no hay un gasto excesivo de comunicación entre los sistemas, es decir, Apache Arrow se encarga de realizar un acceso común para los mismos, utilizan el mismo formato de memoria, además, se destaca el procesamiento paralelo y pueden compartir funcionalidad.

#### **Arquitectura de Apache Arrow**

La arquitectura de Apache Arrow*,* indica una representación estandarizada como capa de datos común, formato de columnas, estas son compatibles con el *hardware* al iterar entradas en una sola columna (*SIMD*, *caches*, etc.), mejor para muchos algoritmos, bibliotecas y API para más de 10 idiomas para construir y acceder a conjuntos de datos.

**Figura 21**

Arquitectura Apache Arrow



*Nota.* Adaptado de *Arquitectura Apache Arrow* [Gráfico], por Peltenburg et al., 2019.

La figura 21 representa la Arquitectura de Apache Arrow, proporciona una capa de acceso a datos común a todas las aplicaciones, optimiza la localidad de datos y procesamiento paralelo. Es adecuado para los tipos de procesamiento vectorizado de SIMD, admite GPU. Todos los sistemas utilizan el mismo formato de memoria, sin gastos generales para la comunicación entre sistemas y los proyectos pueden compartir funcionalidades.

### *Biblioteca Cudf*

Cudf es la biblioteca de Rapids, de manipulación de DataFrames basada en ApacheArrow, está enfocada en la preparación de datos, acelera la carga, el filtrado y la manipulación de datos, contiene la definición del modelo de memoria gdf, específicamente la columna *gdf\_*, y las funciones de procesamiento de datos pueden operar en esas respectivas columnas (Aramburo, 2019). Cudf actualmente tiene dos *API*: *API* de *C ++,* permite el acceso a todas las funciones de alto nivel y una API de Python, imita frecuentemente a Pandas.

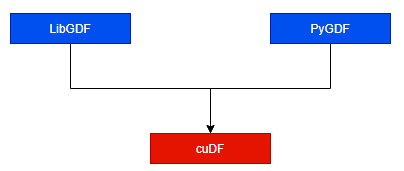
Hablando un poco más acerca de la API de Python, esta es la más fácil de usar, los científicos de datos pueden interactuar con ella casi exactamente como lo harían con Pandas para las respectivas transformaciones de datos, la limpieza y el análisis respectivo que se requiere a los datos a evaluar (Aramburo, 2019).

#### **Características de Cudf**

Conformado por dos aspectos principales *Libgdf, Pygdf,* hace que la biblioteca trabaje de manera eficiente para el procesamiento y análisis de los datos.

**Figura 22**

Estructura de biblioteca Cudf



*Nota.* Adaptado de *Estructura de biblioteca Cudf* [Gráfico], por Patterson, 2019.

La figura 22 representa como está estructurado la biblioteca cudf, se enfoca en puntos primordiales que son Libgdf y Pygdf, esto hace que la biblioteca se enfoque en realizar un trabajo de manera satisfactoria, además, es un repositorio único que contiene tanto la implementación de bajo nivel como envolturas y API de alto nivel, futuros enlaces de idiomas de alto nivel basados en la demanda, comentarios y contribuciones de la comunidad.

Libgdf es una biblioteca de bajo nivel que contiene implementaciones de funciones y *API C, C++.* Importar y exportar un *gdf* utilizando el mecanismo de *cuda IPC,* sus núcleos cuda para realizar operaciones matemáticas basadas en elementos en columnas DataFrameGPU y operaciones de clasificación, unión, agrupación y reducción cuda en marcos de datos GPU.

Pygdf es una biblioteca de Python para manipular marcos de datos de GPU, su interfaz de Python para la biblioteca Libgdf con funcionalidad adicional permite crear gdf a partir de matrices *Numpy*, Pandas DataFrame y *PyArrow* *Tables* y su compilación *JIT* de funciones definidas por el usuario usando *Numba*.

#### **Funcionamiento de Cudf**

En la sección anterior se mencionó acerca de funciones de procesamiento de datos, estas pueden operar en respectivas columnas, dentro de esta sección se hablará de las funciones que tiene cudf como también el funcionamiento de esta biblioteca.

Las funciones de procesamiento de datos cubren un rango, desde simples primitivas de manipulación de transformación de datos tales como concatenación, funciones aritméticas y clasificación, hasta respectivas funciones de análisis de datos que son de nivel superior como unir, agrupar y agregar.

Cudf contiene otras funciones que permiten muchas operaciones adicionales con los usuarios que están familiarizados con Pandas, Numpy o SQL*.* La biblioteca cudf contiene funciones de ingreso de datos, estos permiten a los usuarios leer archivos *csv, json, txt,* entre otros o *Apache Parquet* (*Apache ORC* próximamente) directamente en un gdf(Aramburo, 2019).

### *Biblioteca Cuml*

Cuml es un conjunto de bibliotecas que implementan algoritmos de aprendizaje automático aceleradas por GPU y funciones primitivas matemáticas que comparten API compatibles con otros proyectos de Rapids. Cuml permite a los científicos de datos, investigadores e ingenieros de software ejecutar tareas tradicionales de MachineLearning en GPU, cumpliendo un processo de división de datos en entreamiento y prueba, preprocesamiento de los datos, es decir, cambiar de valores normales a valores categóricos y por último utilizar el conjunto de algoritmos que tiene cuml. Cuml presenta operaciones con múltiples GPU, múltiples nodos y también una lista creciente de algoritmos.

#### **Algoritmos soportados por Cuml**

**Tabla 2**

*Algoritmos soportados por Cuml*

|  |  |  |
| --- | --- | --- |
| Categoría | Algoritmo | Notas |
| *Clustering* | *Density-Based Spatial Clusterind of Applications with Noise (DBSCAN).* |  |
| *K-means.* | Multi-nodo multi-GPU a través de Dask |
| *Dimensionality Reduction* | *Principal Componentes Analysis (PCA).* | Multi-nodo multi-GPU a través de Dask |
| *Truncated Singular Value Decomposition (tSVD).* | Multi-nodo multi-GPU a través de Dask |
| *Uniform Manifold Approximation and Projection (UMAP).* |  |
| *Random Projection.* |  |
| *Linear Models for Regression or Classification* | *Linear Regression (OLS).* | Multi-GPU disponible en el paquete conda CUDA 10. |
| *Linear Regression with Lasso or Ridge Regularization.* |  |
| *ElasticNet Regression* |  |
| *Logistic Regression.* |  |
| *Stochastic Gradient Descent (SGD), Coordinate Descent (CD), and Quasi-Newton (QN) (including L-BFGS and OWL-QN) solvers for linear models.* |  |
| *Nonlinear Models for Regresion or Classification* | *Random Forest (RF) Classification.* | GPU multi-nodo-experimental a través de Dask. |
| *Random Forest (RF) Regression.* | GPU multi-nodo-experimental a través de Dask. |
| *K-Nearest Neighbors (KNN) Classification.* | Multi-nodo multi-GPU a través de Dask, disponible en la versión 0.12 y paquetes conda nocturnos. Utiliza Faiss para la consulta de vecinos más cercanos. |
| *K-Nearest Neighbors (KNN) Regression.* | Multi-nodo multi-GPU a través de Dask, disponible en la versión 0.12 y paquetes conda nocturnos. Utiliza Faiss para la consulta de vecinos más cercanos. |
| *Support Vector Machine Classifier (SVC).* |  |
| *Time Series* | *Support Vector Machine Classifier (SVC).* |  |
| *Linear Kalman Filter.* |  |
| *Holt-Winters Exponential Smoothing.* |  |
| *Auto-regressive Integrated Moving Average (ARIMA).* |  |

Nota: Representa los principales conjuntos de algoritmos que tiene Cuml, están agrupados por categoria y dentro de cada categoría los algoritmos qe corresponden a cada uno de ellos. Todos estos algoritmos son soportados por la librería Cuml, facilita el processo de aprendizaje automático com rapidez y forma parte de la librería Rapids (RAPIDS Development Team, 2018).

A continuación, se explica de manera concisa y muy explícita de que trata algunos algoritmos de cuml*,* características y funcionalidades:

* *Linear Regression:* Se usa en tareas de regresión donde uno quiere predecir, por ejemplo, ventas o precios de la vivienda, además, se usa en tareas de extrapolación o series de tiempo, modelado de sistemas dinámicos y muchas otras tareas de aprendizaje automático.
* *Ridge Regression:* Se usa igualmente como *Linear Regression*, se usa con más frecuencia ya que no sufre problemas de multicolinealidad, se utiliza en la predicción de primas de seguros, análisis de bolsa y mucho más.
* *Stochastic Gradient Descent:* Se enfoca en optimizar alguna función de costo mediante pasos de gradiente, esto hace que sea muy atractivo para grandes problemas difíciles o incluso imposible de encontrar.
* *Nearest Neighbors:* Incluyen sistemas de recomendación donde se utiliza el contenido o el filtrado colaborativo, dado que es un modelo generativo relativamente simple, se usa en tareas de visualización de datos y regresión o clasificación.
* *K-Means Clustering:* La ventaja de K-Means es su velocidad y simplicidad esto hace que, sea la primera opción de muchos profesionales de un algoritmo de agrupamiento, cuando se conoce aproximadamente la cantidad de clústeres, como las tareas de agrupación de datos grandes, la segmentación de imágenes y la agrupación médica.
* *Dbscan:* Puede encontrar grupos de forma no lineal, permite que sea resistente al ruido, dbscan se aplicado para el análisis de “*particle collisons in the Large Hadron Collider”*, la segmentación de clientes en análisis de marketing y mucho más.
* *Principal Component Analysis:* Se usa en la práctica para la visualización y compresión de datos, además, visualiza incrustaciones de palabras extremadamente grandes como *Word2Vec[[30]](#footnote-30)* y se ha utilizado para distinguir células cancerosas de células sanas.
* *Truncated SVD:* Se utiliza en tareas de recuperación de información, sistemas de recomendación y comprensión de datos.

Se planean más algoritmos de ML en cuml y más primitivas de ML para futuras versiones que incluyen:

* *Spectral* *embedding* (Inclusión espectral).
* *Sprectral clustering* (Agrupación espectral).
* *Support vector machines* (Máquinas de vectores de soporte).
* *Additional time series methods* (Métodos de series de tiempo adicionales).

### *Biblioteca Cugraph*

Cugraphes una biblioteca de algoritmos de gráficos que se integra a la perfección con el ecosistema de ciencia de datos de rapids, permite al analista de datos llamar fácilmente a algoritmos de gráficos utilizando los datos almacenados en un marco de datos de GPU.

Cugraphhace que el análisis del gráfico sea omnipresente hasta el punto de que los usuarios solo piensen en términos de análisis y no en tecnologías o marcos. La velocidad de memoria interna de una GPU permite que cugraph cambie rápidamente la estructura de datos para adaptarse mejor a las necesidades de la analítica en lugar de limitarse a una sola estructura de datos.

También es una colección de datos para analizar gráficos que procesa los datos que se encuentran en los marcos de datos de GPU. El principal objetivo que tiene cugraph es proporcionar una API similar a *NetworkX[[31]](#footnote-31):* es una biblioteca que se enfoca en estudiar gráficos y redes, para construir flujos de trabajo acelerados por GPU más fácilmente.

Cugraphutiliza otros proyectos basados en dask de rapids como *dask-cuda*: ayuda al despliegue y la gestión de los trabajadores de dask en sistemas con múltiples GPU en el apartado 2.3.7 se explica dask*-* cugraph; por medio de cugraph distribuirá de manera inteligente la carga de trabajo entre los recursos disponibles (RAPIDS Development Team, 2018).

#### **Algoritmos soportados por Cugraph**

**Tabla 3**

*Algoritmos soportados por Cugraph*

|  |  |  |
| --- | --- | --- |
| Algoritmo | Escala | Notas |
| *PageRank* | *Multi-GPU* |  |
| *Personal PageRank* | *Single-GPU* |  |
| *Katz Centrality* | *Single-GPU* |  |
| *Jaccard Similarity* | *Single-GPU* |  |
| *Weighted Jaccard* | *Single-GPU* |  |
| *Overlap Similarity* | *Single-GPU* |  |
| *SSSP* | *Single-GPU* | Actualizado para proporcionar información de ruta |
| *BFS* | *Single-GPU* | Tambien versión BSP |
| *Traingle Counting* | *Single-GPU* |  |
| *K-Cores* | *Single-GPU* |  |
| *Core Number* | *Single-GPU* |  |
| *Subgraph Extraction* | *Single-GPU* |  |
| *Spectral Clustering-Balanced-Cut* | *Single-GPU* |  |
| *Spectral Clustering-Modularity Maximization* | *Single-GPU* |  |
| *Louvain* | *Single-GPU* |  |
| *Renumbering* | *Single-GPU* |  |
| *Basic Graph Statistics* | *Single-GPU* |  |
| *Weakly Connected Components* | *Single-GPU* |  |
| *Strongly Connected Components* | *Single-GPU* |  |

Nota: Representa los principales conjuntos de algoritmos que tiene Cugraph, cada algoritmo obtiene una escala de Multi-GPU (puede analizar un número de máquinas que contengan GPU) o Single-GPU (puede analizar solo una máquina que contenga GPU). Todos los algoritmos demostrados en la tabla sirven para poder analizar distintas gráficas, es decir analisar datos de gráficos (RAPIDS Development Team, 2018).

A continuación, se explica de que trata algunos algoritmos de cugraph*.* Estos algoritmos están diseñados para ejecutarse en una sola GPU con conjuntos de datos de alrededor de 500 millones de bordes o menos:

* *Jaccard Similarity:* Mide la similitud de vecindario entre vértices conectados; dentro de los sistemas de recomendaciones, esto es muy útil para encontrar clientes con un comportamiento similar o igual (Rees, 2019).
* *Weighted Jaccard:* Es similar a Jaccard, con la excepción que el algoritmo suma los pesos de los vértices.
* *Page Rank:* Utilizada por motores de búsqueda, sin embargo, tiene aplicaciones en análisis de redes sociales, sistemas de recomendación y para nuevos usos en ciencias naturales al estudiar la relación entre proteínas y en redes ecológicas.
* *Single Source Shortest Path:* Se enfoca en identificar la ruta más corta entre un par de vértices; por ejemplo, dentro de una red de carreteras se puede utilizar para encontrar la ruta más rápida de A a B, además, se puede utilizar para optimizar una amplia gama de problemas logísticos (Rees, 2019).
* *Breadth-First Search:* Es un algoritmo de búsqueda clásico que explora iterativamente el gráfico, comenzado en un punto inicial, el algoritmo da un salto por iteración.
* *Spectral Clustering:* Agrupa vértices en función de sus características, de modo que haya una gran similitud intragrupo y una baja similitud entre grupos. Spectral Clustering construye una matriz, resuelve un problema de valor propio asociado y extrae la información de división de los vectores propios calculados (Rees, 2019).
* *Louvain Clustering:* Usa la modalidad como la métrica para combinar iterativamente vértices en grupos, comienza con cada vértice en su propio grupo y funciona iterativamente grupos basados en el modularidad (Rees, 2019).

### *Biblioteca Cuxfilter*

Como la visualización es un componente clave para un científico de datos, existe cuxfilter, es un framework de Rapids que conecta visualizaciones web a filtros cruzados por GPU, además, realiza un filtrado multidimensional interactivo y súper rápido de más de 100 millones de conjuntos de datos tabulares de filas a través de cudf.

#### **Arquitectura Cuxfilter**

*Cuxfilter* resuelve los problemas al aprovechar el poder de la pila rapids.ai, principalmente cudf, los datos se mantienen en una GPU como un marco de datos de GPU y las operaciones como agregaciones grupales, clasificación y consulta se realizan en la propia GPU, solo devolviendo el resultado como salida a los gráficos (RAPIDS Development Team, 2018).

Cuxfilter actúa como una biblioteca de conectores, proporciona las conexiones entre diferentes bibliotecas de visualización y un marco de datos de GPU, permite al usuario usar gráficos de diferentes bibliotecas en un solo tablero.

Para Enemark (2018) la arquitectura de cuxfilter general, necesita un backend completo y una API del lado del cliente, utiliza un servidor *Sanic* para acceder a las funciones cudf de python, un servidor *express* con *node.js* para manejar las llamadas de encadenamiento y una conexión *socket.io* a la API cuxfilter para conectarse a cualquier biblioteca de visualización basada en *JavaScript.*

**Figura 23**

Arquitectura Cuxfilter



*Nota.* Adaptado de *Arquitectura Cuxfilter* [Gráfico], por RAPIDS Development Team, 2018, Flickr (https://rapids.ai).

La figura 23 representa la arquitectura de cuxfilter, aprovecha el portátil de Jupyter *(*es una app cliente servidor, proporciona un entorno pensado para el flujo de trabajo de ciencia de datos y la simulación numérica), el servidor *bokeh* (facilita a los usuarios de python crear app web interactivas, cuando se manipula el gráfico, se sincroniza com el servidor de bokeh y se actiban devoluciones automaticamente con el navegador y este se utiliza durante el análisis exploratorio, en un cuaderno jupyter) para reducir en gran medida la complejidad del *backend*.

Además, cuxfilter está inspirado o basado en la biblioteca Crossfilter, es un mecaniso de filtrado rápido en el navegador a través de múltiples dimensiones y ofrece características que hace opraciones grupales, una de las limitaciones de Crossfilter es que mantiene los datos en memoria de un navegador del lado del cliente, que lo hace ineficiente para procesar grandes conjunto de datos, cuxfilter trabaja con los siguientes módulos que seran representados en la siguiente tabla.

**Tabla 4**

*Modulos Cuxfilter*

|  |  |
| --- | --- |
| Modulo | Descripción |
| *cuxfilter.DataFrame* | Permite crear un DataFrame (Marco de Datos) tipo cuxfilter. |
| *cuxfilter.DashBoard* | Permite crear el gráfico. |
| *cuxfilter.charts* | El tipo de gráfico que se va a graficar. |
| *cuxfilter.layouts* | El diseño del gráfico. |
| *cuxfilter.themes* | El tema que va a tener el gráfico, es decir, el estilo que va a tener. |

Nota: Los distintos módulos que tiene cuxfilter para crear los gráficos estadísticos(PyTorch, 2019)(PyTorch, 2019)(PyTorch, 2019)(PyTorch, 2019).

### *Dask*

Las computadoras portátiles y las estaciones de trabajo de hoy en día son sorprendentemente útiles y se usan correctamente, además, pueden manejar conjuntos de datos y cálculos para los que anteriormente se dependía de los clústeres.

Los científicos de datos utilizan a menudo herramientas como Pandas*, Scikit-Learn,* numpy y el resto del ecosistema python para analizar datos en su computadora personal, sin embargo, cuando elijen analizar un conjunto de datos más grandes, descubren que esas herramientas no fueron diseñadas para escalar más allá de una sola máquina.

Dask proporciona formas de escalar los flujos de trabajo de Pandas, Scikit-Learn, numpy de forma más nativa, con una reescritura mínima, descubre como dividir grandes cálculos y enrrutar partes de ellos de manera más eficiente en hardware distribuido. Dask se ejecuta rutinariamente en clústeres de miles de máquinas para procesar cientos de terabytes de datos de manera eficiente dentro de entornos seguros (Team, 2016).

Dask es un programador de cómputo distribuido, es extremadamente modular con la programación, transferencia de datos y manejo fuera del núcleo, todo desunido, lo que permite conectar nuestras propias implementaciones (Patterson, 2019).

Dask puede capacitar a los científicos de datos y analistas para manipular conjuntos de datos de 100GB en la computadora portátil o conjuntos de datos de 1TB sin molestarse en absoluto con un clúster.sz, cálculos paralelos eficientes en máquinas individuales al aprovechar sus CPU de múltiples núcleos y transmitir datos de manera eficiente desde el disco, dask cambia el clúster por planificadores de una sola máquina que son livianos, no requieren configuración y pueden ejecutarse completamente dentro del mismo proceso de sesión del usuario (Team, 2016).

#### **Dask y Rapids**

Rapidsutilizadask para escalar cargas de trabajo de python desde computadores portátiles a clústeres de supercomputadoras.

Puede ejecutar fácilmente varios trabajadores dask por nodo para permitir un modelo de desarrollo más sencillo de un trabajador por GPU, independientemente del entorno único o de múltiples nodos.

***Dask-cudf:***Dask puede escalar hacia arriba y hacia afuera con cudf, utiliza primitivas cudf debajo en operaciones de estilo de reducción de mapas con la misma API de alto nivel, aprovecha los de hardware utilizando un marco de comunicaciones llamada *OpenUCX[[32]](#footnote-32)* (Patterson, 2019)*.*

***Dask-cuml:*** Una integración nativa con Dask-cudf, puede usar fácilmente a los trabajadores de Dask para inicializar avances *NCCL* para operaciones de recolección o dispersión optimizadas, además, proporciona primitivas fáciles de usar y de alto nivel para la sincronización de trabajadores, es necesario para muchos algoritmos de ML(Patterson, 2019)*.*

***Dask-cugraph:*** Contiene algoritmos de análisis de gráficos paralelos que pueden hacer uso de múltiples GPU en un solo host, es capaz de jugar muy bien con otros proyectos en el ecosistema dask, así como otros proyectos de rapids, como dask-cudf y dask-cuml.

### *Integración con Bibliotecas de Deep Learning*

Rapids se integra a la perfección con bibliotecas de aprendizaje profundo, para la computación acelerada, este aparatado se enfoca en algunas librerías que trabajando con aprendizaje profundo y tienen integración con rapids.

#### **Chainer**

*Chainer* es un framework de python, permite principalmente a los investigadores, entrenar y evaluar rápidamente modelos de aprendizaje profundo.

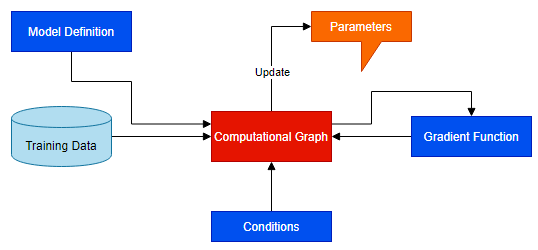
Chaineres un marco de código abierto diseñado principalmente para la investigación eficiente y el desarrollo deDeepLearning. El enfoque dechaineres único: construir el grafico computacional “sobre la marcha” durante el entrenamiento. Permite a los usuarios cambiar el gráfico en cada iteración o por cada muestra, según las condiciones.

Chainer proporciona formas imperativas de declarar redes neuronales para admitir operaciones compatibles con Numpy entre matrices y también incluye computación numérica basada en GPU (Shohei, 2016)*.*

Para entrenar una red neuronal en Chainer se necesitan tres pasos muy importantes: construir un gráfico computacional a partir de la definición de red, ingresar datos de entrenamiento, calcular la función perdida y por último actualizar los parámetros usando optimizador y repetir hasta la convergencia (Shohei, 2016).

**Figura 24**

Proceso de Chainer Define-by-Run



*Nota.* Adaptado de *Chainer Define-by-Run* [Gráfico], por Shohei, 2016.

La figura 24 representa cómo funciona la biblioteca Chainer para Deep Learning, el gráfico computacional no se proporciona antes del entrenamiento, sino que se obtiene en el curso del entrenamiento. El cálculo directo corresponde al gráfico computacional y la propagación hacia atrás de él, cualquier modificación en el respectivo gráfico computacional se hará el cálculo directo en cada interacción e incluso para cada muestra (Shohei, 2016).

##### **Características de Chainer**

Está conformado por tres puntos principales, que permite que este framework sea utilizado por bastantes científicos e investigadores, los siguientes son:

**Tabla 5**

*Fast Chainer*

|  |  |  |
| --- | --- | --- |
| **Fast** | *CUDA* | *Supports GPU acceleration using CUDA wint CuPy* |
| *CuDNN* | *High-speed traininginference with cuDNN optimixzed Deep Learning functions with CuPy.* |
| *NCCL* | *Supports a fast, multi-GPU Learning using NCCL with CuPy.* |

Nota: Admite computo Cuda; requiere de unas pocas líneas de código para aprovechar una GPU, también se ejecuta em múltiples GPU con poco esfuerzo (Chainer, 2019).

**Tabla 6**

*Intuitive Chainer*

|  |  |  |
| --- | --- | --- |
| ***Intuitive*** | *Define-by-Run* | *Easy and intuitive to Write a network. Supports Dynamic graphs.* |
| *High debuggability* | *User-friendly error messages. Easy to debug using pure Python debuggers.* |
| *Easy to use APIs* | *Well-abstracted common tools for varios NN Learning, easy to write set of Learning flows.* |
| *Low Learning curve* | *No needs to learn a new tensor API since Chainer uses Numpy and CuPy.* |
| *Maintainable codebase* | *Written in pure and well documented.* |

Nota: Incluye cualquier declaración de flujo de control de python sin carecer de la capacidades de propagación hacia atrás, además, el código es intuitivo y fácil de depurar (Chainer, 2019).

#### **Mxnet**

Mxnetes una librería de aprendizaje profundo, verdaderamente de código abierto adecuado para la producción y creación de prototipos de investigación flexible.

Mxnetes un marco diseñado para la eficiencia y la flexibilidad, permite mezclar programación simbólica e imperativa para poder maximizar la eficiencia y la productividad, contiene un programador de dependencia dinámico que paraleliza automática las operaciones simbólicas e imperativas sobre la marcha; *mxnet* hace que la ejecución sea simbólica, rápida y eficiente en la memoria. Mxnetes portátil y liviano, escalable de manera efectiva a múltiples GPU y múltiples máquinas (MXNET, 2019).

##### **Características de Mxnet**

Frontal Híbrido:Un front-end hibrido pasa a la perfección entre el modo imperativo ansioso de *Gluon[[33]](#footnote-33)* y el modo simbólico para proporcionar flexibilidad y velocidad.

Entrenamiento distribuido:La capacitación es distribuida escalable y la optimización del rendimiento en investigación y producción están habilitadas por el servidor dual de parámetros y el soporte de *Horovod* es un entorno de entrenamiento distribuido popular y de código abierto para escalar entrenamiento con *TesorFlow* en múltiples GPU (MXNET, 2019).

Lenguage Bindings:Integraciónprofunda con el lenguaje Python, tiene soporte para otros lenguajes como *Scala, Julia, Clojure, Java, C++, R y Perl.*

Herramientas y Librerías:Consta de un próspero ecosistema de herramientas y librerías, permite casos de usos de visión por computadora, *PNL[[34]](#footnote-34)*, series de tiempo y más.

#### **Pytorch**

*Pytorch* está diseñado para ser intuitivo, lineal en pensamiento y fácil de usar por los programadores, cuando se ejecuta una línea de código, está se ejecuta, no existe visión asincrónica del mundo. Cuando se ingresa un depurador o recibe mensaje de error y rastrea fallos, estos son fáciles de comprenderlos y son sencillos (PyTorch, 2019).

**Tabla 7**

*Componentes Pytorch*

|  |  |
| --- | --- |
| Componente | Descripción |
| *Torch* | Una biblioteca Tensor con *Numpy*, con un fuerte soporte de GPU. |
| *Torch.autograd* | Una librería de diferenciación automática basada en cinta que admite todas las operaciones de Tensor diferenciables en *Torch*. |
| *Torch.jit* | Una pila de compilación (*TorchScript*) para crear modelos con *Autogrand* diseñada para la máxima flexibilidad. |
| *Torch.nn* | Una librería de redes neuronales profundamente integrada con Autograd diseñada para la máxima flexibilidad. |
| *Torch.multiprocessing* | Multiprocesamiento de Python, pero con un intercambio mágico de memoria de los tensores de Torch en los procesos. |
| *Torch.utils* | *DataLoader* y otras funciones de utilidad para mayor comodidad. |

Nota: Pytorch está conformado por algunos componentes principales y su descripción de cada componente, ayudan que el trabajo sea más eficiente para los processos de Deep Learning (PyTorch, 2019).

Pytorch netamente lo usan porque es un remplazo de numpy para poder usar el poder de las GPU y también como una plataforma de investigación de aprendizaje profundo que proporciona la máxima flexibilidad y velocidad.

Pytorch proporciona una amplia variedad de rutinas tensoras para acelerar y ajustar sus necesidades de computación científica, como segmentación, indexación, operaciones matemáticas, algebra lineal y reducciones (PyTorch, 2019).

##### **Características de Pytorch**

TorchScript***:*** Proporciona una transición perfecta entre el modo ansioso y el modo gráfico para acelerar el camino hacia la respectiva producción (PyTorch, 2019).

Entrenamiento distribuido:Capacitación distribuida escalable y la optimización del rendimiento en investigación y producción habilitadas por el backend distribuído.

Herramientas y librerías:Ecosistema de herramientas y librerías Pytorch, admite el desarrollo en visión por computadora.

Socios en la nube:Soportado en las principales plataformas en la nube, proporciona un desarrollo sin fricción y escalado fácil.

#### **Numba**

El compilador numba permite a los respectivos usuarios anotar funciones costosas, luego son compiladas por *LLVM[[35]](#footnote-35)* al momento de ser llamadas. Numba hace uso de la sintaxis del decorador de python, permite que su uso se aplique de forma incremental sin requerir una reescritura importante (Crist, 2016). La forma más común de utilizar numba es por medio de una colección de decoradores que se pueden aplicar a sus respectivas funciones para indicarle a Numba que las compile. Cuando se realiza una respectiva llamada a una función propia de numba, esta compila el código de máquina *“just-in-time”* para su ejecución y todo o parte de su código puede ejecutarse posteriormente a la velocidad del código de máquina nativo (Numba, 2019).

##### **Python y Numba**

Numba hace que el código de pythonsea rápido (ver figura 26), realiza una llamada a un compilador de código abierto, se traduce un subconjunto de códigopythonynumpyen código rápido de máquina.

Numba puede acelerar todas las funciones escritas en python centradas principalmente en cálculo y pesadas computacionales, tiene soporte para biblioteca numpy, por lo tanto, realiza cálculos y acelera el cálculo general ya que los bucles en python son muy lentos.

**Figura 25**

Proceso de Trabajo Python-Numpy



*Nota.* Adaptado de *Proceso de Trabajo Python-Numpy* [Gráfico], por Puneet, 2018.

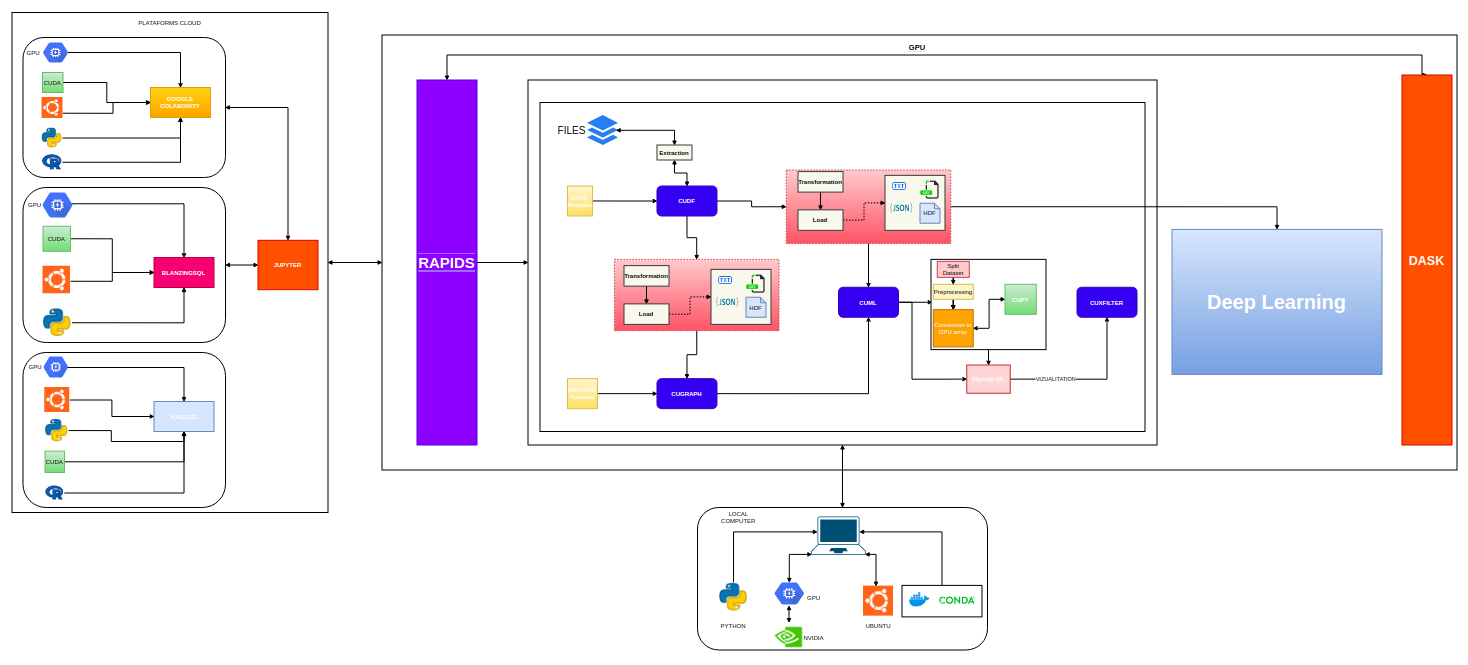
En la figura 25 representa el proceso de trabajo de python-numpy, la función python se toma, optimiza y se convierte en la representación intermedia de numba, después de la inferencia de tipo, que es como la inferencia de tipo numpy, se convierte en código interpretable LLVM, este código se envía al compilador justo a tiempo de LLVM para entregar el código de la máquina (Puneet, 2018).

Generación de código sobre la marcha (tiempo de importación o tiempo de ejecución, según las preferencias del usuario), tambien permite generación de código nativo para el hardware de la CPU (predeterminado) y GPU e integración con la pila de software científico de Python.

### *Análisis e implementación de la Arquitectura y características de Rapids*

**Figura 26**

Arquitectura Rapids



*Nota.* Arquitectura Rapids [Gráfico], por Autor, 2020.

La figura 26 representa el análisis e implementación de la arquitectura propuesta para Rapids, se explica de manera dinámica, concisa y sólida, a través de los siguientes puntos:

Rapids trabaja de forma rápida, eficiente y eficaz, necesita estar en un ambiente dedicado al entorno de ciencia de datos, para construir un ambiente de tal magnitud se debe realizar un proceso respectivo:

* Rapids permite extraer, leer y analizar archivos tipo *csv, json, txt, hdf,* entre otros.
* El conjunto de bibliotecas principales de rapids permitirá llevar a cabo un flujo de trabajo, ETL (Extracción, transformación y Carga), este proceso lo realiza la biblioteca cudf para el análisis exploratório y procesamiento de los datos, además, rapids permite realizar el análisis de gráficos, con la biblioteca cugraph, está biblioteca trabaja a la par con cudf y cuml.
* Una vez realizado el proceso de ETL los datos ya se encuentran totalmente limpios y la información es entendible, se procede a dividir todo el dataset en data de entrenamiento y data de prueba, para esto se utiliza la biblioteca cuml, despues, se realiza un proceso de preprocesamiento de los datos, es decir, cambiar los datos normales a datos categóricos, con el fin de poder utilizar algoritmos Machine Learning.
* Para los algoritmos de Machine Learning lo realiza la biblioteca cuml, la misma ya contiene un conjunto de algoritmos para poder utilizar (ver tabla 2), para que el proceso de entrenamiento sea mucho más rápido, se convierte los datos a GPU array utilizando la librería cupy (es una biblioteca de código abierto acelerada, proporciona computación acelerada por GPU con python), esto permite que el entrenamiento del modelo del algoritmo selecionado sea más rápido trabajando de manera paralela.
* Visualización, cuxfilter para la respectiva representación gráfica en forma de gráficos estadísticos.

Rapids se puede integrar con bibliotecas de Deep Learning, para mejorar la aceleración de extremo a extremo, cabe recalcar que los científicos de datos y analistas se enfocan de manera profunda en este tipo de algoritmos, para trabajar con este tipo de bibliotecas debe pasar por el proceso de ETL que aplica la biblioteca cudf.

Rapids para mejorar su rendimiento y tiempos de esfuerzo, tiene una fusión con dask, particularmente si estas analizado y procesando bastante información, dask con rapids permiten crear clusteres en la propia máquina personal, com el fin de acelerar el flujo de trabajo de python desde tu computadora portátil y ejecutar fácilmente varios trabajadores.

Rapids debe trabajar en el lenguaje de programación de python, algunos características importantes del porque trabajar con este lenguaje de programación:

* En el 2018, python fue el lenguaje de programación más popular para la ciencia de datos, año tras año es cada vez más atraído por los científicos de datos (Aguerzame et al., 2019).
* Python reúne características necesarias para la ciencia de datos, permite simplificar muchas cosas como también mejorar el código sea legible a través de la sintaxis.
* Los científicos de datos necesitan lidiar, analizar, procesar problemas complejos, Python proporciona todas las herramientas necesarias para llevar el respectivo caso de ETL, ML y visualización trabajando con la librería dedica rapids.
* Python permite equipar a los científicos de datos para implementar soluciones factibles, al mismo tiempo sigue los estándares de los algoritmos requeridos del proceso de análisis, desarrollo y solución.

Rapids trabaja en dos ambientes diferentes, con plataformas en la nube (cloud) y en un computador personal (localmente); como plataformas en la nube tenemos las siguientes:

* + Rapids con *Google* *Colabority.*
  + Rapids con *BlazingSQL.*
  + Rapids con *Kaggle*.

En el capítulo 4 abarcaremos la instalación de rapids en las plataformas en la nube, se detallará el proceso de funcionamiento y entornos de configuración en Google Colabority y BlazingSQL*.* Además, el proceso de instalación en computador personal (localmente), también se detallará en el capítulo 4.

# Capítulo tres

# Trabajos Relacionados



## **Fuentes de Información**

*Google Scholar* es un *b*uscador que permite a los investigadores, científicos y estudiantes localizar varios documentos académicos como artículos, trabajos de titulación, libros, etc.

Rapidsfuente de información que permite a los científicos de datos, estudiantes y desarrolladores localizar varios documentos y blogs con respecto a la Librería Open Source Rapids.

## **Cadenas de Busqueda**

### Tabla 8

### *Cadenas de Busqueda*

|  |  |  |
| --- | --- | --- |
| **Bases de Datos** | **Cadena de Búsqueda** | **Trabajo** |
| Scholar | “GPU” +” using NVDIA RAPIDS” +” Computing Acelerated” | Accelerated Simulation of Air Pollution Using NVIDIA RAPIDS. |
| Scholar | “GPU” +” using NVDIA RAPIDS” +” Computing Acelerated” | GPU Acceleration of PySpark using RAPIDS AI. |
| Scholar | “GPU” +” using NVDIA RAPIDS” +” Computing Acelerated” | Accelerating recommender system training 15x with RAPIDS. |
| Rapids | Ninguna | Security Alert Analysis Using GPUs. |
| Rapids | Ninguna | Accelerating Random Forest up to 45x using cuML. |
| Rapids | Ninguna | Combining Speed & Scale to Accelerate K-Means in RAPIDS cuML |

Nota: Todas las cadenas de busqueda que se realizó para encontrar los trabajos relacionados para el presente trabajo de titulación y hacer un análisis completo de cada trabajo.

## **Criterios de Inclusión y Exclusión**

### Tabla 9

### *Criterios de Inclusión y Exclusión*

|  |  |
| --- | --- |
| **Criterio de Búsqueda** | **Parámetro** |
| Rango de Fecha | 2020 al presente |
| Tipo de Documento | Full papers, books, article |
| Tipo de Acceso | Todos |
| Idioma | Inglés |

Nota: Criterios de inclusión y exclusión para el presente trabajo de titulación.

## **Análisis de Trabajos Relacionados**

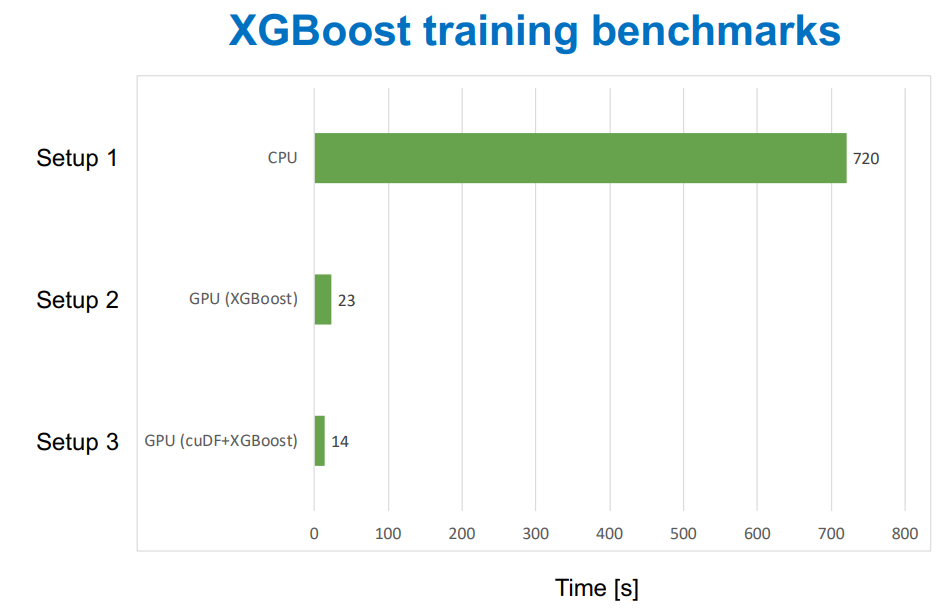
### *Accelerated Simulation of Air Pollution Using Nvidia Rapids*

La simulación presenta un enfoque alternativo para el cálculo de la química atmosférica basado en el aprendizaje automático. Utilizan un conjunto de datos de entrenamiento, se produce utilizando el modelo del Sistema de Observación de la Tierra *Goddard* de la *NASA* (*GEOS*) con química *GEOS-Chem,* se ejecuta en el centro de simulación climática de la NASA para descubrir el clúster de supercomputación en 384 núcleos *Intel Xeon Haswell[[36]](#footnote-36).*

El conjunto de datos contiene las concentraciones de contaminación del aire, parámetros físicos como la temperatura y la intensidad del sol. La aplicación usara Dask*-*Cudf y *Dask-XGBoost* conRapids utilizando 8 GPU Tesla *V100* para el entrenamiento de los datos utilizan modelos de árboles de decisión impulsados por gradiente que pueden reproducir la simulación de la cinética química, además la aplicación aprovecha al máximo los recientes avances en Dask-XGBoost[[37]](#footnote-37), como el escalado de múltiples nodos y múltiples GPU para la distribución de grandes datos. Muestran los beneficios de este enfoque y se discute la posible aceleración de este modelo de química atmosférica acelerada de aprendizaje automático. Incorporan modelos de árboles potenciados en el modelo de referencia GEOS utilizando la APIC de XGBoost, lo que permite una integración perfecta para los modelos entrenados por GPU en GEOS-Chem (Keller et al., 2019).

**Figura 27**

Evaluación comparativa de XGBoost con CPU, GPU y XGBoost cudf + GPU



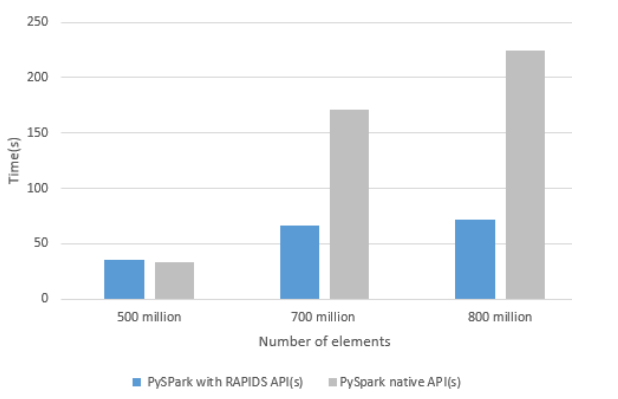
*Nota.* Adaptado de *Evualación comparativa de XGBoost con CPU, GPU y XGBoost cudf + GPU* [Gráfico], por Keller et al., 2019.

### *Gpu Acceleration of PySpark using Rapids AI*

La investigación e implementación se centra en aprovechar muchos núcleos y una gran plataforma de memoria, con un enfoque de escalamiento en mente, el enfoque es implementar en algunos servidores grandes *BullSequana[[38]](#footnote-38)* con muchos núcleos y memoria grande, además, está dedicado a exponer la solución que aprovecha las GPU en el flujo de trabajo de PySpark utilizando la biblioteca Rapids*.* Utilizan Docker acoplable que va a contener todas las bibliotecas necesarias, también la etiqueta de imagen “*cuda9.2-runtime-ubuntu16.04*” y un portátil Jupyter y le agregan Spark, el servidor consta con 8x12 núcleos y memoria RAM de 4 terabytes. El proceso de verificación para asegurar la conexión entre Rapidsy Spark sea posible y delegar tareas de Spark en la GPU, abre nuevas posibilidades para que Apache Spark sea capaz de aprovechar servidores de gama alta como BullSequana acoplados a GPU(Aguerzame et al., 2019).

**Figura 28**

PySpark + Rapids y PySpark nativo



*Nota.* Adaptado de *PySpark + Rapids y PySpark nativo* [Gráfico], por Aguerzame et al., 2019.

### *Accelerating recommender system training 15x with Rapids*

El documento proporciona una aceleración de la generación de funciones y el tiempo de entrenamiento del modelo, aceleran la capacitación del modelo en un factor de 15.6x, desde un flujo de trabajo de 891.8s a 57.2s, a través de una combinación de la biblioteca Rapids, cudf para preprocesamiento, un cargador de datos por lotes, *LAMB* y tamaños de lotes extremos y una respectiva actualización del kernel responsable de calcular el gradiente de incrustación en Pytorch. Usando cudf aceleran a un factor de 9.7x al realizar cálculos en la GPU, reduciendo el tiempo necesario para generar características sumamente primordiales de 51 min a 5 min (Rabhi et al., 2019).

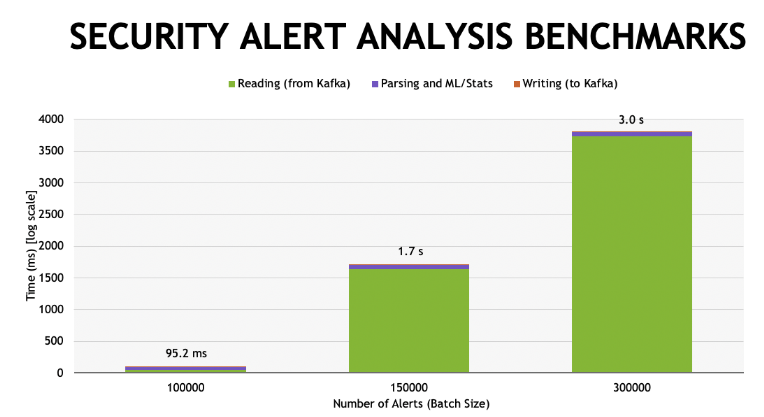
### *Security Alert Analysis Using Gpus*

El documento muestra cómo se puede procesar más de 300.000 alertas sin procesar en menos de 3s con un solo Tesla *V100*. Además, permitir que los equipos de seguridad ingieran y analicen más tipos de registros y las cantidades de registros aumenta la visibilidad en toda la red y proporciona meta-alertas complejas en tiempo real. Al utilizar la aceleración de GPU, se puede proporcionar simultáneamente análisis en tiempo real de estas alertas cada vez mayores. Las alertas se agregan por tipo y por día, las normalizan por el número total de alertas por tipo y la traza como un mapa de calor para que los analistas puedan ver rápidamente las tendencias en sus alertas y compararlas entre sí.

Para Allen & US (2020) los componentes del flujo de trabajo del análisis de alertas son tanto el análisis del registro de alertas como el flujo de trabajo analítico. *CLX[[39]](#footnote-39)* tiene un módulo útil que analiza formatos de registros comunes usando Rapids. *Regex[[40]](#footnote-40)* para el análisis de alertas se almacena en un archivo *yaml* y el analizador notable *Splunk[[41]](#footnote-41)* lee ese archivo y ejecuta la funcionalidad de análisis. Debido a que CLX se construye utilizando Rapids, se puede analizar una gran cantidad de alertas muy rapidamente.

**Figura 29**

Análisis comparativo de alertas con una sola GPU



*Nota.* Adaptado de *Análisis comparativo de alertas con una sola GPU* [Gráfico], por Allen & US, 2020, Flickr (<https://bit.ly/2ZJEeqg>)

### *Accelerating Random Forest up to 45x using cuml*

El documento proporciona una revisión de algoritmos básicos de Random Forest, como su entrenamiento puede ser paralelo a las GPU Nvidia y números de referencia que demuestran el rendimiento. Cada árbol de decisión se entrena sobre una muestra diferente con reemplazo del conjunto de datos original.

La biblioteca Random Forest de cuml contiene dos algoritmos divididos de alto rendimiento para seleccionar que valores exploran para cada combinación de característica más nodo: El algoritmo cuml puede construir varios árboles en paralelo en una sola GPU, cada árbol está construido en su propio flujo cuda controlado.

Para Vishal (2019) garantizar el mejor rendimiento, utiliza un marco de datos de GPU como entrada para cuml y una matriz numpy como entrada para sklearn. Además, para comprender completamente el mejor rendimiento que se puede obtener del entrenamiento forestal aleatorio utilizando GPU, se amplía la prueba a las ejecuciones de múltiples GPUs utilizando el enfoque distribuido basado en dask usando un conjunto de datos de 8.8M filas para entrenar y 1000 filas para probar, hardware de servidor *DGX-1* con ocho GPU V100-16GB, para las pruebas múltiples GPU se utiliza 1000 árboles por modelo y una profundidad máxima a 8, 12 o 16 (Vishal, 2019).

### *Combining Speed & Scale to Accelerate K-Means in Rapids cuml*

K-Meansde cuml proporciona el método inherentemente secuencial, combina muestreo aleatorio junto con distribuciones de distancias de puntos a cada muestra para dispersar los centroides iniciales a través del espacio donde existen puntos reales.

Muchos algoritmos de cuml se construyen a través de las primitivas de cuda; los algoritmos de multi-nodo, multi- GPU se ejecutan dentro del entorno de Dask, lo que facilita la carga de conjuntos de datos de gran tamaño en un marco de datos distribuido por cudf y utiliza modelos de aprendizaje automático en GPU.

Para Nolet (2019) que realizo el trabajo identifica que, una vez entrenados los centroides, se mantienen en un solo trabajador Dask, la predicción se realiza de manera paralela trasmitiendo los centroides a los trabajadores, cuml puede escalar a un gran número de GPU y nodos. Se comparó dos algoritmos de *K-Means-cuML* con K-Means de Dask-cuML, ambos algoritmos fueron comparados en dos máquinas, utilizando 8 trabajadores por nodo(Nolet, 2019).

**Figura 30**

Evaluación comparativa de múltiples nodos, múltiples GPUs cuml y Dask-Cuml



*Nota.* Adaptado de *Evaluación comparativa de mútiples nodos, múltiples GPUs cuml y Dask-Cuml* [Gráfico], por Nolet, 2019, Flickr (https://bit.ly/2WF3Keo).

### *Gpu Accelerated Cyber Log Parsing with Rapids*

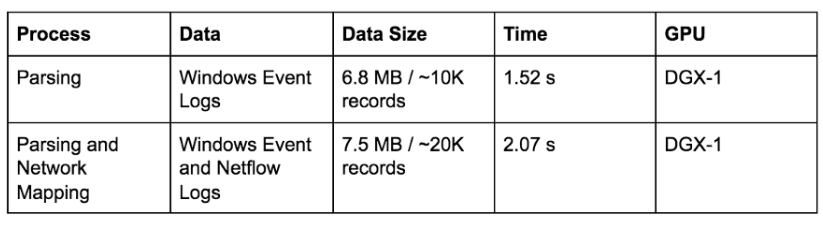
El documento presenta un caso de uso para abordar el problema de operaciones de registros de seguridad cibernética, el flujo de trabajo se centra en dos bibliotecas Rapids; cudf y dask-cudf. Se demuestra el uso de Rapids para analizar los registros de eventos de Windows (registros WinEVT) proporciona un aumento de velocidad y ofrece benefícios inmediatos de integración con técnicas de aprendizaje automático.

Rhodes (2019) pronuncia que, los datos que se utilizaron son registros de eventos de Windows de tipo 4624 y 4625, los datos son proporcionados por el Laboratorio Nacional de los Alamos, el código 4624 representa un evento de inicio de sesión exitoso y el código de evento 4625 representa un evento de inicio de sesión fallido.

A los respectivos datos se realizo un proceso de ETL, eliminan caracteres no imprimibles, se aplica expresiones regulares diferentes a cada tipo de registro, en fin, utilizan Rapids para analizar los datos del registro de eventos utilizando la GPU, el tiempo de procesamiento de todo el análisis de los datos dentro del cuaderno Jupyter se demuestra en la siguiente imagen.

**Figura 31**

Tiempo de procesamiento a los datos de registro de eventos Windows



*Nota.* Adaptado de *Tiempo de procesamiento a los datos de registros de eventos de Windows* [Gráfico], por Rhodes, 2019 Flickr (https://bit.ly/30uMjQ7).

# Capítulo cuatro

# Implementación con Rapids

## **Instalación de la Librería Rapids**

Para todo el proceso de implementación de la librería Rapids se ira detallado paso por paso como es el proceso de instalación en las plataformas cloud como también en la máquina personal localmente, además se ira cumpliendo el flujo de trabajo de la arquitectura analizada e implementada que se explico en la sección 2.3.9. La instalación de Rapids se puede realizar de dos maneras, es necesario explicarlas y dar a conocer a los desarrolladores, estudiantes y científicos de datos que Rapids obtiene la misma funcionalidad.

### *Instalación en Máquina Personal*

La instalación de Rapids en un computador portátil localmente necesita tener ciertos prerrequisitos para ser uso de la librería, se incorpora los principales puntos para la respectiva instalación.

**Tabla 10**

*Prerrequisitos de Instalación Rapids*

|  |  |  |  |
| --- | --- | --- | --- |
| **GPU Nvidia** | **Ubuntu** | **Docker** | **Cuda y Nvidia Drivers** |
| Titan RTX  Tesla  GeForce | Versión 16.04  Versión 18.04 | Versión 19.03 | Versión 10.0  Versión 10.1  Versión 10.2 |

Nota: Los prerrequisitos que debe tener un computador personal para instalar la librería Open Source Rapids.

La máquina personal donde se realizó la instalación de Rapids cumple con los prerrequisitos mencionados, se detalla las especificaciones y características que tiene el computador.

**Tabla 11**

*Especificaciones computador portátil*

|  |  |  |  |
| --- | --- | --- | --- |
| **Marca Portátil** | **Intel** | **Sistema Operativo** | **GPU** |
| Dell | Core I7 8th Gen | Ubuntu 18.04 | Nvidia GeForce MX150 |

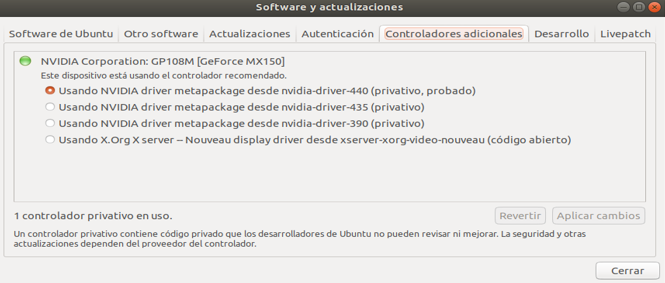
Nota: Características que tiene la máquina personal, se detalla los requerimentos que tiene el mismo y donde se procedió a instalar la librería Rapids.

Cabe destacar que, para instalar la librería se debe tener como base el sistema operativo *Ubuntu* y GPU deNvidia, si no contiene estos dos aspectos la librería no se podrá instalar.

Primeramente verificar si el sistema operativo tiene instalado y actualizado los controladores o drivers de Nvidia para determinar que el computador cuenta con GPU y los drivers se encuentren activos.

**Figura 32**

Nvidia activa

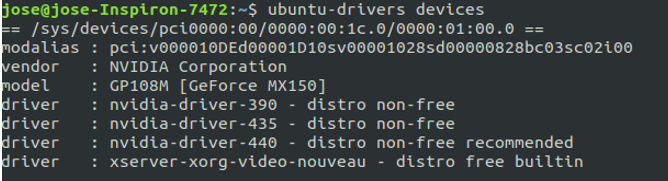


*Nota.* Nvidia activa [Gráfico], por Autor, 2020.

La figura 32 representa que, dentro del sistema operativo se encuentran instalados los controladores de Nvidia y funcionando en su totalidad.

**Figura 33**

Drivers disponibles





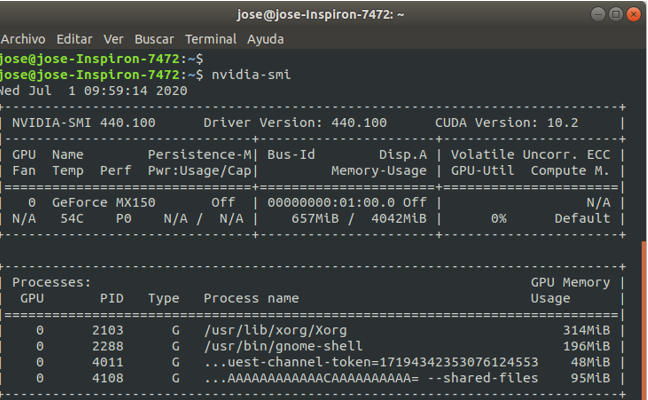
*Nota.* Drivers disponibles y controlador [Gráfico], por Autor, 2020.

La figura 33 representa el comando de instalación del controladorGPU Nvidiapara el Sistema Operativo Ubuntu.

Una vez instalado el controlador abrir nuevamente otro terminal para verificar si ya se encuentra activa la GPU.

**Figura 34**

Gpu Nvidia Activa



*Nota.* Gpu Nvidia Activa [Gráfico], por Autor, 2020.

La figura 34 representa la GPU Nvidia activa y funcionando con todas las características de la GPU, es decir, que la máquina portatil ya tiene activa la GPU.

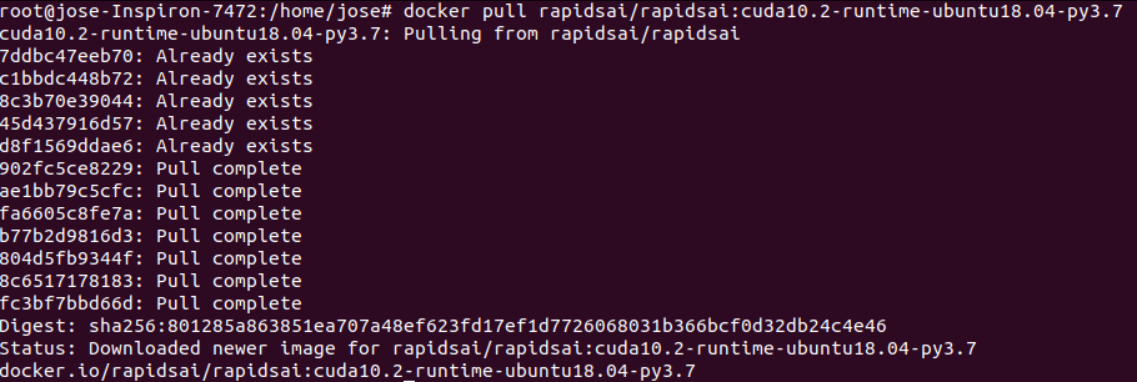
#### **Instalación con Docker**

Docker crea contenedores ligeros y portables, para distintas aplicaciones de software, dentro de cada contendor se puede instalar librerías, herramientas, bases de datos, entre otras, independientemente del sistema operativo que el computador tenga por debajo, con el fin de facilitar los despliegues.

El proceso de instalación de la librería Rapids con docker se detalla de la siguiente manera, a continuación el proceso siguiente.

**Figura 35**

Instalación de la librería Rapids al contenedor creado en la máquina personal



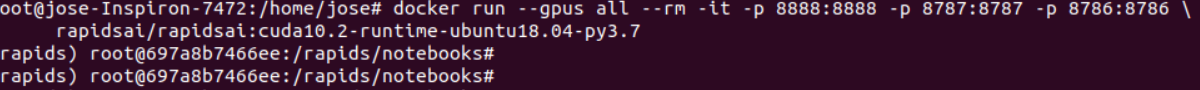
*Nota.* Instalación de la librería Rapids al contenedor creado en la máquina

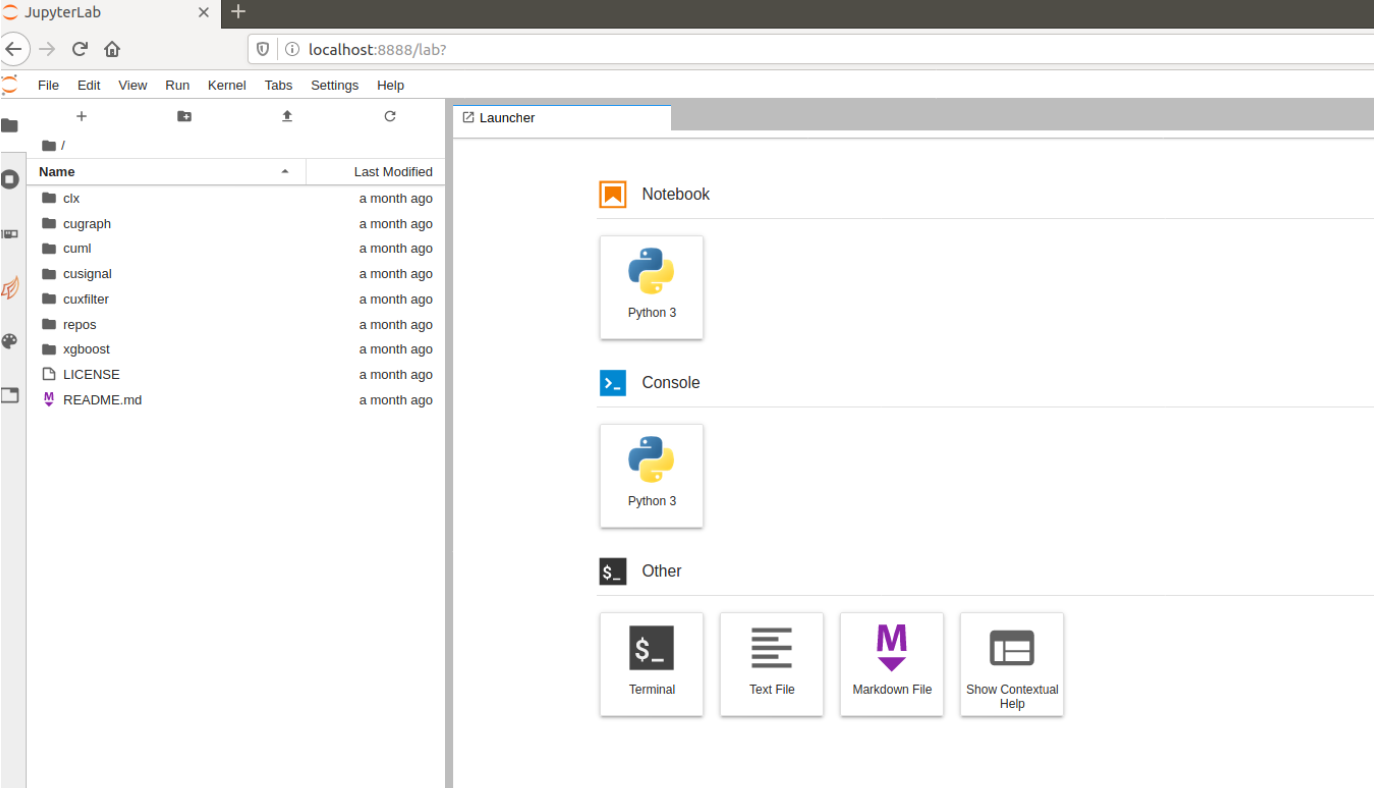
personal [Gráfico], por Autor, 2020.

La figura 35 representa la instalación de Rapids con todo su conjunto de bibliotecas, características y su funcionamiento, además, se instala el lenguaje de programación Python a su última versión 3.7.

**Figura 36**

Iniciar los cuadernos de JupyterLab con Docker + Rapids





*Nota.* Iniciar los cuadernos de JupyterLab con Docker +

Rapids [Gráfico], por Autor, 2020.

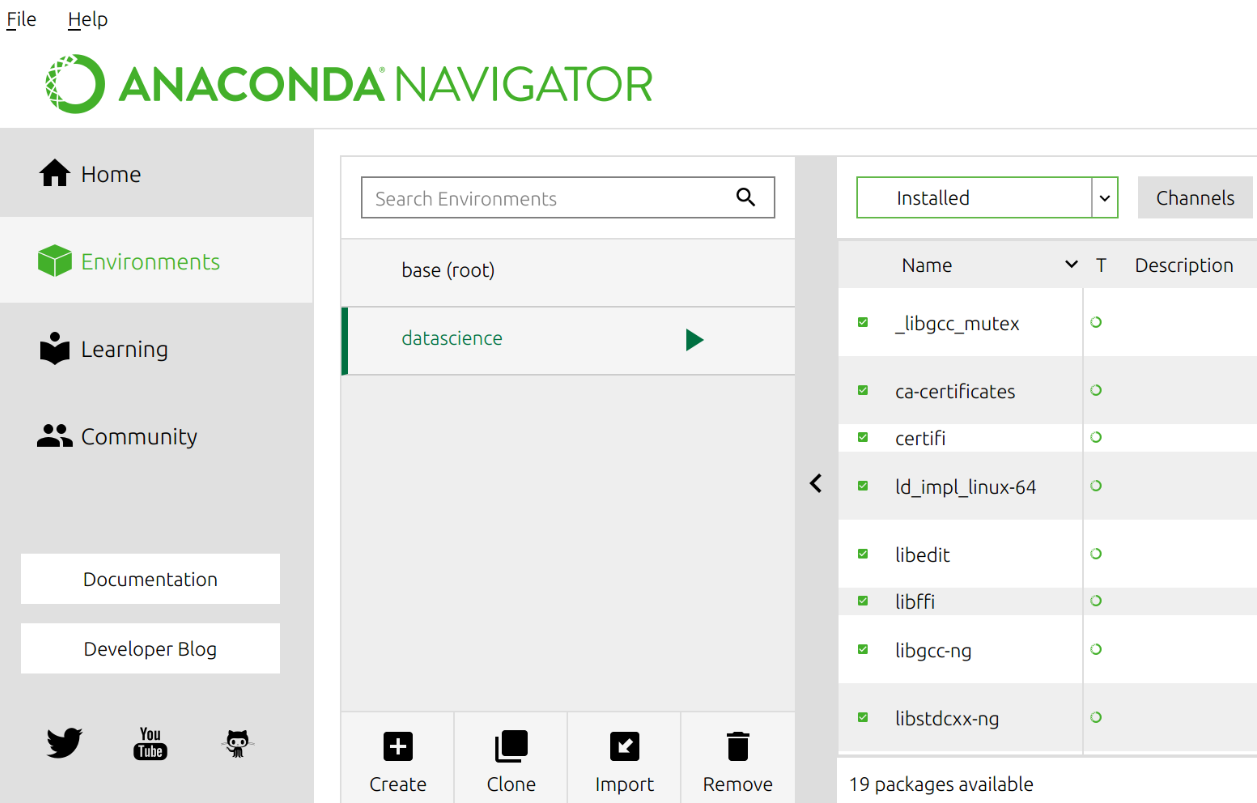
La figura 36 permite correr el contenedor donde se encuentra instalada la librería Rapids, también inicia el entorno de trabajo de los cuadernos de JupyterLab (se encuentra en ejecución a través del puerto 8888) donde se va a codificar y hacer uso de la librería, el entorno de trabajo contiene cuadernos JupyterLab (como ejemplos donde se utiliza la librería Rapids), además, se obtiene el funcionamiento de toda la librería dask.

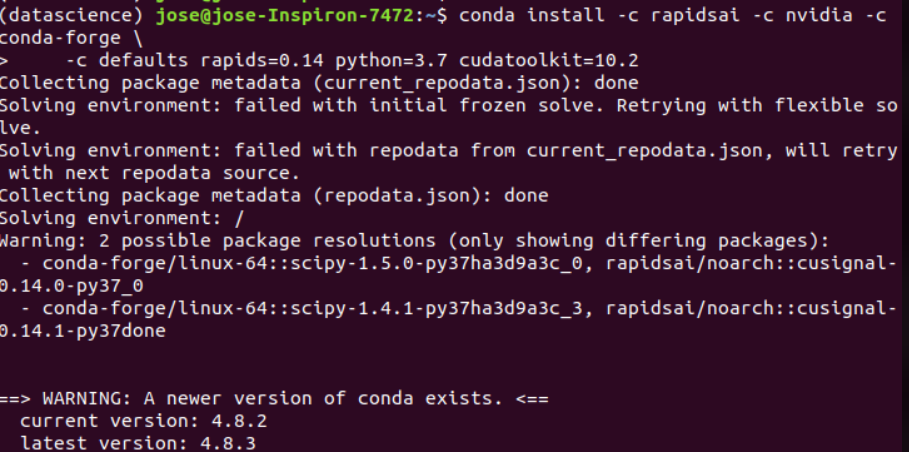
#### **Instalación con Conda**

*Conda* es un sistema de gestión de paquetes y un sistema de gestión de entornos pueden ser ejecutados en sistemas operativos como *Linux*, *Windows*, *macOS*; Conda es utilizada para la Ciendia de Datos y aprendizaje automático, incluye un procesamiento de grandes volúmenes de información, análisis predictivo y cómputos científicos impulsada por python, además, es gratis porque es un proyecto de código abierto, orientado para poder simplificar el despliegue, administración de los paquetes, desarrollar aplicaciones de manera eficiente, rápida y sencilla. El proceso de instalación de la librería rapids con Conda se detalla de la siguiente manera.

**Figura 37**

Crear entorno en la plataforma de Conda e Instalación de Rapids





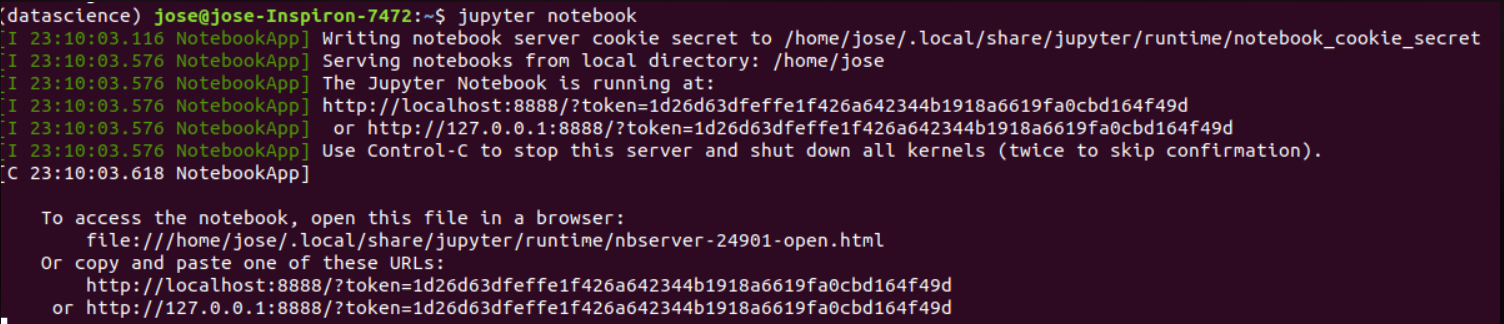
*Nota.* Crear entorno en la plataforma de Conda e Instalación de Rapids

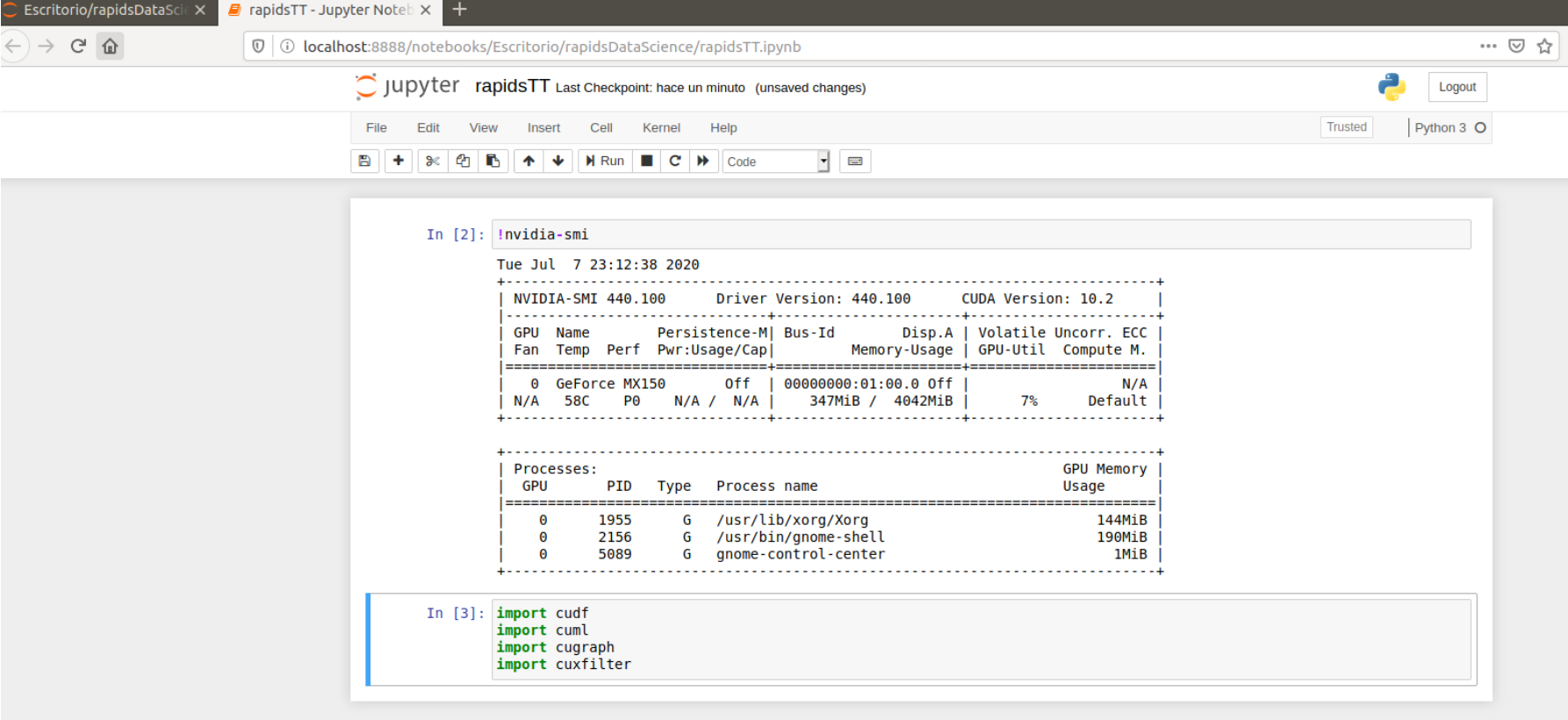
[Gráfico], por Autor, 2020.

La figura 37 representa el panel de administración de Conda, proporciona una interfaz gráfica amigable donde se crea el entorno de trabajo llamado *datascience* y se instala la librería Rapids a través del comando especificado en la figura.

**Figura 38**

Plataforma de desarrollo Jupyter Notebook con Conda + Rapids





*Nota.* Plataforma de desarrollo Jupyter Notebook con Conda + Rapids [Gráfico], por Autor, 2020.

La figura 38 representa el entorno de trabajo a través de Jupyter Notebook se ingresa al servidor local por el puerto 8888 (pertenece el puerto de ingresso a los de Jupyter-Notebook), en la figura se puede visualizar que la librería Rapids ya está funcionando con todo su conjunto de bibliotecas.

### Plataforma Google Colabority

Google Colabority es una plataforma de entorno gratuito de Google basado en Jupyter Notebook que permite escribir, ejecutar código en python en un navegador como Google Chrome, no necesita de configuración requerida, tiene acceso gratuito a GPU y facilidad de compartir código y un Sistema Operativo Linux preinstalado.

Google Colabority permite crear modelos de Ciencia de Datos, Machine Learning y Deep Learning, libera a la máquina personal de tener un trabajo demasiado extenso, costoso en tiempo y potencia, incluso permite a la máquina trabajar de manera eficiente en la nube.

#### ***Entorno de Ejecución***

Se debe habilitar el entorno de ejecución a GPU, se realiza este cambio para poder utilizar la librería Rapids, ya que la misma trabaja en ambiente GPU.

**Figura 39**

Entorno de ejecución GPU Google Colabority





*Nota.* Entorno de ejecución GPU Google Colabority [Gráfico], por Autor, 2020.

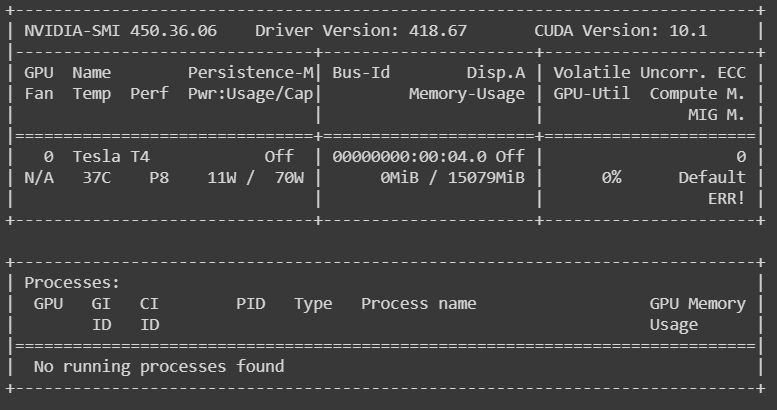
La figura 39 representa parte de la plataforma de desarrollo de Colabority como también la configuración al entorno de ejecución que debe tener la plataforma, el cual debe estar en GPU para que funcione la librería Rapids.

#### ***Instalación Rapids***

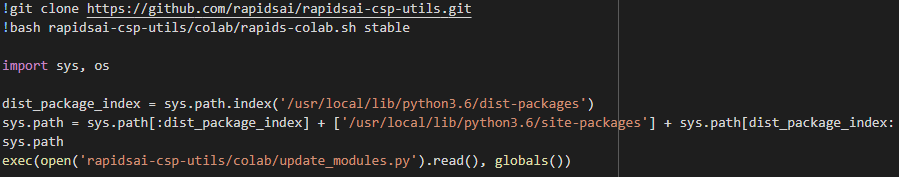
Google Colabority permite verificar el tipo de GPU con el que se va a trabajar, además, esta plataforma ya tiene preinstaladas distintos tipos de GPUs

**Figura 40**

Tipos GPU Tesla en Google Colabority e Instalación de Rapids







*Nota.* Tipos de GPU Tesla en Google Colabority e Instalación de Rapids [Gráfico], por Autor, 2020.

La figura 40 representa los tipos de GPUs Tesla y la instalación de Rapids en Google Colabority, la librería Rapids para que funcione en la plataforma se debe trabajar con la arquitectura de GPU Nvidia Tesla T4.

### Plataforma BlazingSQL

BlazingSQL ejecuta demostraciones libres, se encuentra en un entorno de trabajo JupyterLab y es de uso gratuito, permite ejecutar rapidamente BlazinSQL + Rapids, permite acelerar el procesamiento de datos, además, es un motor SQL acelerado por GPU construído en el ecosistema de Rapids.

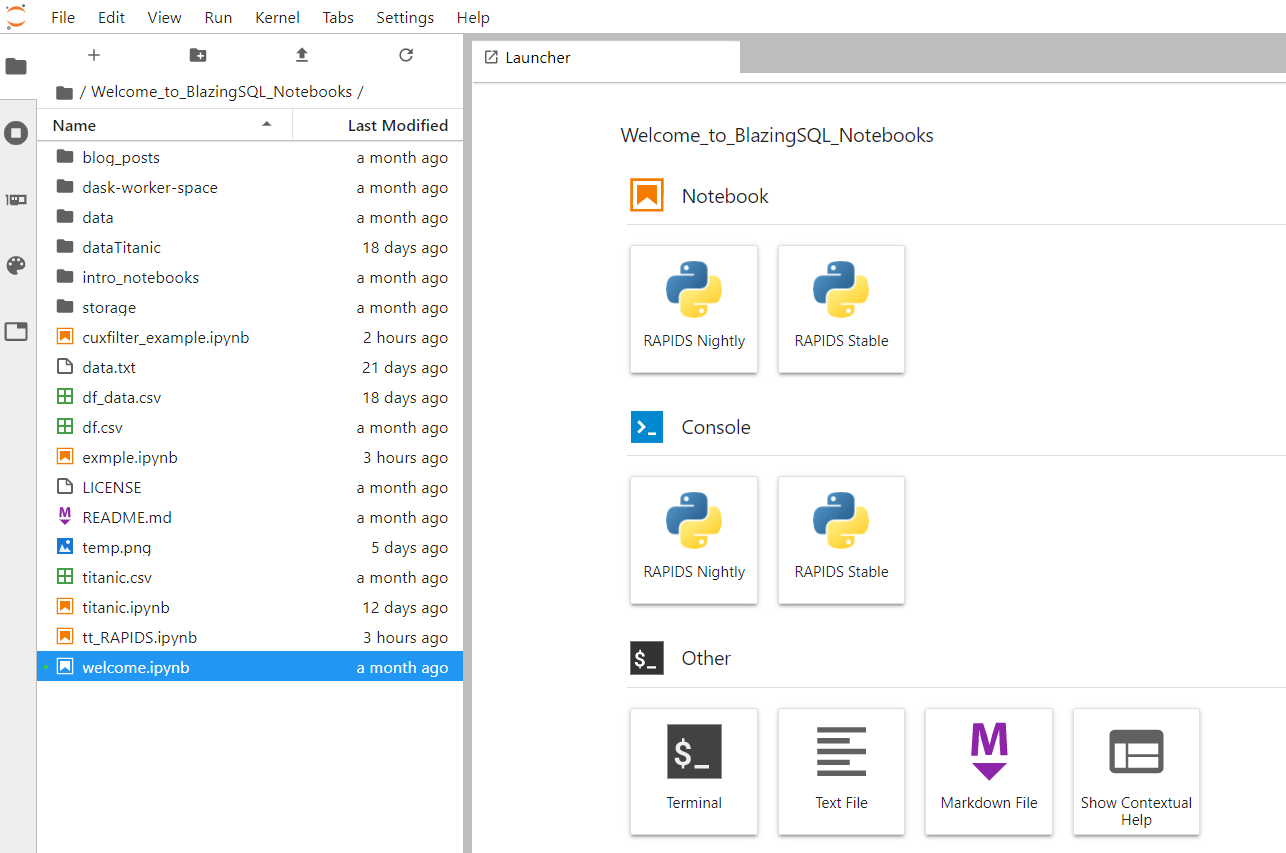
Se creó para abordar los gastos, la complejidad, el grande consumo de memoria en las CPUs y el ritmo lento con el que se enfrentan los usuarios cuando trabajan con un gran número de información.

BlazingSQL se puede utilizar de manera fácil, el procedimento para utilizar esta plataforma es el siguiente:

* Ingresar a su página principal: <https://app.blazingsql.com/>
* Seguidamente logearse con una cuenta de Gmail.
* Por ultimo, solo se procede a selecionar el cuaderno de trabajo de Rapids, es importante utilizar los cuadernos estables de Rapids que actualmente se encuentra en la versión 0.14.

**Figura 41**

Plataforma BlazingSQL + Rapids



*Nota.* Plataforma BlazingSQL + Rapids [Gráfico], por Autor, 2020.

La figura 41 representa el entorno de trabajo de desarrollo en la plataforma BlanzingSQL + Rapids, se puede evidenciar en la imagen que BlazingSQL ya cuenta con los cuadernos de Rapids Stable.

## **Proceso de Extracción, Transformación y Carga**

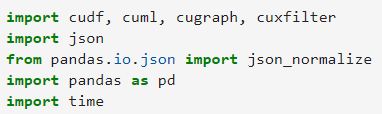
Todo el proceso de ETL utilizando la librería Rapids se desarrollo en la plataforma cloud BlazingSQL, cade destacar que esta plataforma ya tiene preinstalada una GPU que es primordial para acelerar el flujo de trabajo y el proceso de ETL utilizando Rapids que trabaja en ambientes GPU.

### *Extracción con la Biblioteca Cudf*

La fase de extracción se trabajo con los datos de OpenCampus, la data contiene 200.000 filas de información, los respectivos datos no contenían un esquema fijo, era información con datos semiestructurados, no tenían ningun tratamiento, no generaban valor alguno y no era entendible. A continuación se detalla el proceso de extracción y lectura de los datos.

**Figura 42**

Importar bibliotecas de Rapids conjuntamente con otras librerías a utilizar

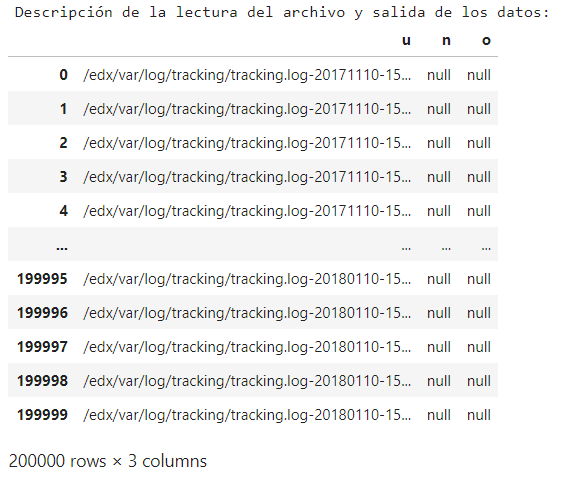


*Nota.* Importar bibliotecas de Rapids conjuntamente con otras librerías a utillizar [Gráfico], por Autor, 2020.

**Figura 43**

Extracción y Lectura de la Información con Rapids





*Nota.* Extracción y Lectura de la Información con Rapids [Gráfico], por Autor, 2020.

La figura 42 y 43 representa la importación del conjunto de bibliotecas de Rapids con otras librerías que se utilizaran en el transcurso del desarrollo de todo el proceso de ETL, además, se incorpora la respectiva característica de Rapids que permite leer archivos csv, como es cudf.read\_csv y por ultimo la salida de los datos.

A continuación una tabla que demuestra los diferentes tipos de extracción y lectura que se puede hacer con la biblioteca cudf.

**Tabla 12**

*Tipos de extracción y lectura con cudf*

|  |  |  |
| --- | --- | --- |
| **Tipo** | **Cudf** | **Rapids** |
| json | cudf.read\_json | Stable 0.14 |
| csv | cudf.read\_csv | Stable 0.14 |
| parquet | cudf.read\_parquet | Stable 0.14 |
| orc | cudf.read\_orc | Stable 0.14 |
| avro | cudf.read\_avro | Stable 0.14 |
| hdf | cudf.read\_hdf | Stable 0.14 |

Nota: Diferentes tipos para extraer y leer archivos utilizando la biblioteca cudf de Rapids.

### *Trasnformación con la Biblioteca Cudf*

La fase de transformación de los datos se realiza un análisis exploratorio de toda la información, en donde se procedera a eliminar valores nulos, duplicados, vacíos, en fin, que la data sea entendible y lista para proceder a cargarla en un nuevo dataset y posteriormente utilizar los algoritmos de ML que se detalla en la sección 4.3.

**Figura 44**

Fase de Transformación de la Data proceso 1

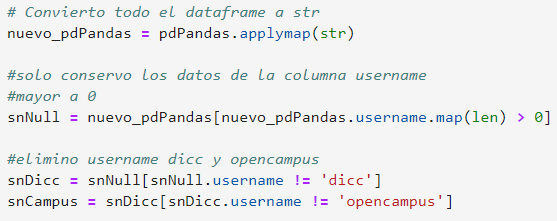


*Nota.* Fase de Transformación de la Data proceso 1 [Gráfico], por Autor, 2020.

La figura 44 representa la descomposición de todo el dataset, se convierte toda la data en una lista pero solo de la columna u, seguidamente se recorre toda la lista y realiza una separación a partir del valor .gz, una vez separado utiliza la funcionalidad de json.loads para cargarlo en formato json y guardarlo en una variable que es de tipo arreglo.

**Figura 45**

Fase de Transformación de la Data proceso 2



*Nota.* Fase de Transformación de la Data proceso 2 [Gráfico], por Autor, 2020.

La figura 45 representa la creación de un nuevo DataFrame tipo Pandas, esta figura representa la interacción de dos librerías Pandas y Rapids; se crea el DataFrame con Pandas y todo el DataFrame se convierte en un str, conservar los datos de la columna username que sean mayores a 0, despues se elimina los valores dicc y opencampus.

**Figura 46**

Fase de Transformación de la Data proceso 3



*Nota.* Fase de Transformación de la Data proceso 3 [Gráfico], por Autor, 2020.

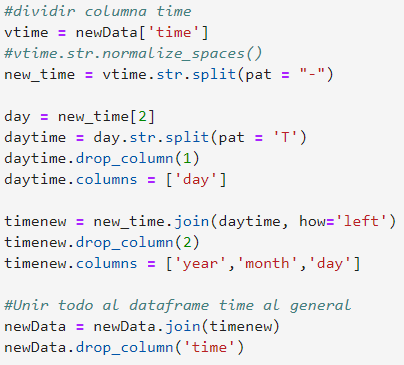
La figura 46 representa la conversión de un DataFrame de Pandas a un nuevo DataFrame de cudf, utilizando la característica cudf.DataFrame.from\_pandas.

**Figura 47**

Fase de Transformación de la Data proceso 4







*Nota.* Fase de Transformación de la Data proceso 4 [Gráfico], por Autor, 2020.

La figura 47 se procede a eliminar datos perdidos utilizando la característica de cudf dropna, se analiza y elimina las columnas que no tienen mucha relevancia en el transcurso de la analítica de los datos con drop\_column, se divide la columna context, dicha columna se encuentra en formato json, a la misma columna normalizamos espacios con str.normalize\_spaces, la separamos con str.split y se elimina columnas que son innecesarias cuando se separa la columna context, la columna dividida (context) se la une nuevamente al dataframe general con join, eliminamos valores duplicados con drop\_duplicates, reseteamos el index con reset\_index y por último se separa toda la columna time y la unimos nuevamente al DataFrame general.

**Figura 48**

Exploración y visualización de la Data Transformada



*Nota.* Exploración y visualización de la Data Transformada [Gráfico], por Autor, 2020.

La figura 48 representa la salida de toda la data transformada, analizada y explorada, verificando las columnas del DataFrame con columns de cudf, la dimensión que ahora tiene la data con shape, los tipos de datos de cada columna con dtypes y visualizar toda la información de manera general utilizando head de cudf.

A continuación una tabla que demuestra algunas de las características de cudf para la exploración, analítica y transformación que se utilizaron para la transformación a los datos de OpenCampus.

**Tabla 13**

*Características de Transformación a los datos con cudf*

|  |  |  |
| --- | --- | --- |
| **cudf** | **descripción** | **Rapids** |
| cudf.DataFrame.from\_pandas | Convertir de un dataframe de pandas a un dataframe cudf | Stable 0.14 |
| to\_pandas | Convertir a un dataframe pandas | Stable 0.14 |
| dropna | Eliminar filas con datos perdidos | Stable 0.14 |
| drop\_column | Eliminar columnas | Stable 0.14 |
| str.ip2int | Convertir direcciones ip a enteros | Stable 0.14 |
| str.normalice\_spaces | Normalizar espacios vacios | Stable 0.14 |
| str.split | Seperar columnas | Stable 0.14 |
| columns | Incorporar nuevos nombres de columnas | Stable 0.14 |
| drop\_duplicates | Eliminar valores duplicados | Stable 0.14 |
| reset\_index | Resetar el index de forma ordenada | Stable 0.14 |
| join | Unir dataframes | Stable 0.14 |
| head | Información general de todo el dataframe | Stable 0.14 |
| shape | Dimesion del dataframe | Stable 0.14 |
| dtypes | Tipos de datos de cada columna del dataframe | Stable 0.14 |
| cudf.DataFrame | Crear un dataframe com cudf | Stable 0.14 |
| tolist | Convertir a una lista | Stable 0.14 |
| sort\_values | Ordenar por los valores en fila | Stable 0.14 |

Nota: Algunas de las características de cudf que se utilizó para el proceso de transformación a los datos de OpenCampus.

### *Carga con la Biblioteca Cudf*

La carga de la información transformada anteriormente correspondía a datos de OpenCampus, como los datos ya se encuentran limpios se procede a realizar la carga en un nuevo archivo csv.

**Figura 49**

Carga de los datos



*Nota.* Carga de los datos [Gráfico], por Autor, 2020.

La figura 49 representa la carga de toda la información que se analizo, exploro y transformo a un nuevo archivo csv, se crea el archivo data\_mejora.csv y utilizando to\_csv de cudf convierte todo el DataFrame a un archivo csv, con esto concluye el proceso de ETL.

A continuación una tabla que demuestra algunas de las características de cudf para la carga de nuevos datos o información.

**Tabla 14**

*Características de carga a los datos con cudf*

|  |  |  |
| --- | --- | --- |
| **cudf** | **descripción** | **Rapids** |
| to\_csv | Convertir y escribir a un archivo csv | Stable 0.14 |
| to\_hdf | Convertir y escribir a un archivo hdf | Stable 0.14 |
| to\_json | Convertir y escribir a un archivo json | Stable 0.14 |
| to\_orc | Convertir y escribir a un archivo ORC | Stable 0.14 |
| to\_parquet | Convertir y escribir a un archivo parquet | Stable 0.14 |
| to\_feather | Convertir y escribir a un archivo feather | Stable 0.14 |

Nota: Características que permite convertir y escribir a distintos tipos de archivos utilizando la biblioteca cudf de Rapids.

## **Proceso de Machine Learning**

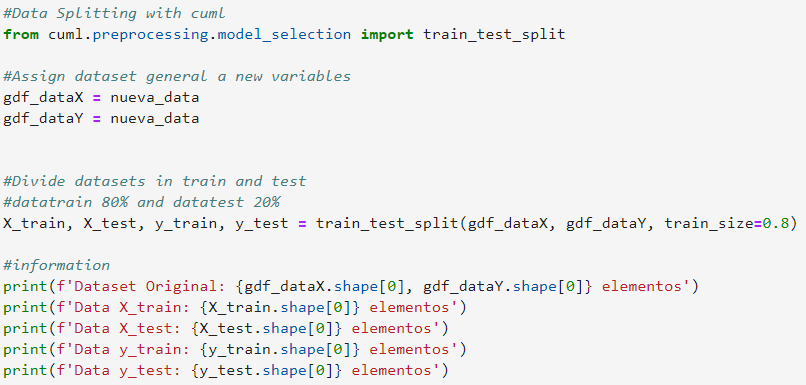
En el proceso de Machine Learning permite que un sistema, por sí mismo aprenda en forma automatizada, a descubrir patrones, tendencias y relaciones con los datos, se oferece mejores perspectivas, como tal, requiere de etapas que se deben cumplir, a continuación se explica los principales fases que se realizo para cumplir el proceso de Machine Learning utilizando la biblioteca cuml de Rapids, todo este proceso se realizo en la plataforma cloud BlazingSQL.

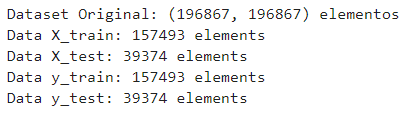
### *Dividir Dataset en Train y Test con la Biblioteca Cuml*

Está operación es común, es la división del conjunto de datos, una parte de entrenamiento que corresponderá a la mayor parte del dataset, se usa para entrenar el modelo y una parte de pruebas, es de menor tamaño, sobre el cual se evalúa el modelo entrenado, a continuación, se detalla la fase de división de la data que cumplio anteriormente el proceso de ETL.

**Figura 50**

Dividir la data en Train y Test





*Nota.* Dividir la data en Train y Test [Gráfico], por Autor, 2020.

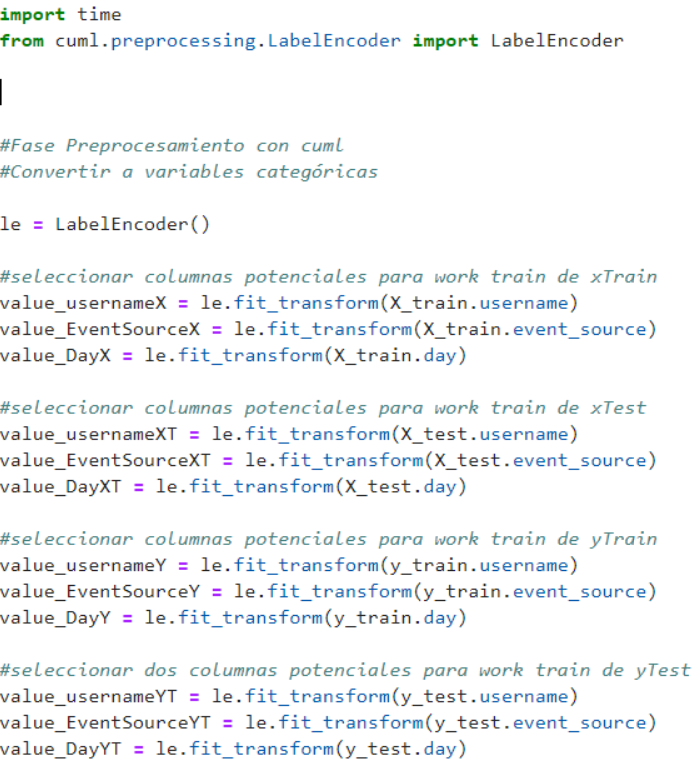
La figura 50 representa la división de todo el conjunto de datos, se utiliza las característica y clase de cuml que es preprocessing.model.selection e importamos train\_test\_split, esta función permite dividir el conjunto de datos en cuatro objetos intercalados imitando el train\_test\_split de skit-learn, se divide de tal manera, para que al momento de utilizar los algoritmos ML de cuml sea más preciso y rápido el entrenamiento, cabe recordar que Rapids trabaja de manera paralela, de tal forma es preciso la división en cuatro objetos, dos de Train a un 80% y dos de Test a un 20%, por lo general es el que más se utiliza, como se puede destacar en la imagen se detalla el resultado de toda la división en los cuatro objetos intercalados para entrenamiento y prueba.

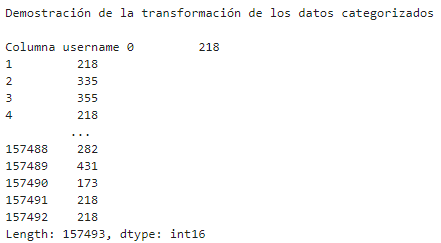
### *Preprocesamiento con Cuml*

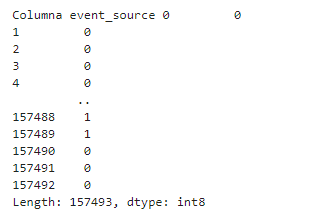
Este aparatado se enfoca en convertir a datos categóricos, escogiendo las columnas que se pretenden categorizar, utilizando la biblioteca cuml, esta conversión permite medir el nivel de entendimiento de lo datos, como también, una ayuda para poder utilizar algoritmos Machine Learning, se debe categorizar los datos porque cada algoritmo entiende valores numéricos o cuantitativos.

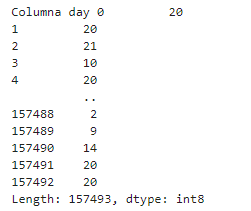
**Figura 51**

Transformar a datos categóricos









*Nota.* Transformar a datos categóricos [Gráfico], por Autor, 2020.

La figura 51 representa la conversión de datos normales a datos categóricos, se escogio tres columnas de la data de entrenamiento (username, event\_source, day) y dos columnas de la data de prueba (username, event\_source, day) utilizando cuml preprocessing.LabelEncoder, es una clase que permite realizar la conversión de las tres columnas seleccionadas a datos categorizados utilizando la función fit\_transform de LabelEncoder que pertenece a la funcionalidades de cuml.

A continuación una tabla que refleja el tipo de configuración y transformación a datos categóricos que podemos realizar utilizando la biblioteca cuml.

**Tabla 15**

*Características de tipos de transformación a datos categorizados*

|  |  |  |
| --- | --- | --- |
| **cudf** | **descripción** | **Rapids** |
| fit | Ajusta la instancia de LabelEncoder a un conjunto de datos categorizados | Stable 0.14 |
| fit\_transform | Ajusta y transforma simultaneamente una entrada (columna o dataframe) a datos categorizados y es la que se utiliza más | Stable 0.14 |
| inverse\_transform | Revertir la etiqueta ordinal a la etiqueta orginal | Stable 0.14 |
| transform | Transforma una entrada a sus claves categóricas | Stable 0.14 |

Nota: Características que permite convertir o transformar los datos a datos categorizados utilizando la biblioteca cuml de Rapids.

### *Aplicación de Algoritmos Machine Learning con la Biblioteca Cuml*

Para el proceso de Machine Learning utilizando la biblioteca cuml se ha analizado, estudiado e implementado dos tipos de algoritmos que pertenecen al conjunto de esta biblioteca, el grupo al que pertenecen es Regresión, dentro de ello se utilizo el algoritmo Regresión Logística y Regresión Lineal, en las siguientes secciones se detalla el proceso de construcción, configuración y entrenamiento de los algoritmos ya especificados.

#### **Algoritmo Regresión Lineal**

Una descripción corta de lo que permite realizar el algoritmo Regresión Lineal, se usa en tareas de regresión donde uno quiere predecir, por ejemplo, ventas o precios de vivienda, además, se usa en tareas de extrapoblación o series te tiempo y muchas otras tareas de aprendizaje automático.

Dentro del desarrollo de este algoritmo, se enfoca en crear el modelo de entrenamiento conjuntamente con los datos que se esta trabajando (OpenCampus) y verificar la aceleración del entrenamiento, para destacar la rapidez que tiene la biblioteca cuml, cuando se está entrenado un modelo con un gran volumen de información; se detallara la aceleración de las pruebas y resultados en el capítulo 5.

**Figura 52**

Utilizando el algoritmo de cuml Regresión Lineal



*Nota.* Utilizando el algoritmo de cuml Regresión Lineal [Gráfico], por Autor, 2020.

La figura 52 representa la demostración y utilización del algoritmo Regresión Lineal, en donde se verifica que, por medio de la biblioteca cuml se obtiene la facilidad de utilizar algoritmos Machine Learning. Se crea un nuevo DataFrame de cudf (cudf.DataFrame) que contiene los datos que se dividio, se categorizo y va a corresponder a un nuevo DataFrame de entrenamiento, seguidamente creamos un cudf Series (cudf.Series), en este caso corresponde a una Serie de data de prueba, este proceso se hace para poder entrenar el algoritmo de Regresión Lineal; se destaca la asociación de la librería cupy, dicha librería permite crear GPUs arrays a cada dato del nuevo DataFrame y Serie, esto permitirá que, el entrenamiento del algoritmo sea más rápido, por último, se llama al algoritmo y se incorpora las configuraciones de entrenamiento que dispone el mismo.

Cabe destacar que el tipo de dato de todo el Dataframe y Serie deben ser float32, si no tiene este tipo de dato en el transcurso del entrenamiento, el algoritmo no se ejecutara y lanzara errores. A continuación unas tablas en donde se detalla las configuraciones y las características que se utilizaron para el algoritmo Regresión Lineal de cuml.

**Tabla 16**

*Desarrollo y configuración para utilizar el algoritmo Regresión Lineal*

|  |  |  |  |
| --- | --- | --- | --- |
| **Biblioteca** | **Característica** | **Descripción** | **Rapids** |
| cuml | cuml.linear\_model | Modelo Lineal | Stable 0.14 |
| cuml | LinearRegression | Algoritmo Regresión Lineal de cuml | Stable 0.14 |
| cudf | cudf.DataFrame | Creación de un DataFrame | Stable 0.14 |
| cudf | cudf.Series | Creación de una Serie | Stable 0.14 |
| cupy | cp.asarray | Crear matrices de GPU arrays | Stable 0.14 |
| cupy | cp.float32 | Tipo dato del GPU array sea float32 | Stable 0.14 |

Nota: Asociación y utilización de las bibliotecas, librerías que se utilizaron para comenzar a entrenar el modelo con el algoritmo Regresión Lineal de cuml.

**Tabla 17**

*Características del algoritmo Regresión Lineal de cuml*

|  |  |  |
| --- | --- | --- |
| **Algoritmo RL** | **Descripción** | **Rapids** |
| fit\_intercept | Si fit\_intercept es true, el algoritmo intenta corregir la media global de la data de prueba. | Stable 0.14 |
| normalize | Los predictores de la data de entrenamiento se normalizarán, en este caso solo si normalize es True. | Stable 0.14 |
| algorithm = “eig” | eig usa una descomposición propia de la matriz de covarianza y es rápido cuando esta entrenando el algoritmo. | Stable 0.14 |
| coef\_ | Resultado de todos los coeficientes cuando ya termino de entrenar el algoritmo. | Stable 0.14 |
| intercept\_ | Resultado de las intercepciones que existe en la data de entrenamiento y pruebas. | Stable 0.14 |
| predict | Resultado de la predicción del algoritmo. | Stable 0.14 |

Nota: Características que se utilizaron en el entrenamiendo del Algoritmo de Regresión Lineal de cuml.

#### **Algoritmo Regresión Logística**

Una descripción corta de lo que permite realizar el algoritmo de Regresión Logística, es un algoritmo simple que interpreta fácilmente los resutados, se usa con más frecuencia ya que no sufre problemas de multicolinealidad, se utiliza en predicción de primas de seguros, análisis de bolsa y mucho más.

En el desarrollo del algoritmo Regresión Logística se crea un modelo de entrenamiento conjuntamente con los datos que se esta trabajando (OpenCampus) para determinar la aceleración del entrenamiento del algoritmo, con el fin de destacar la rapidez cuando se está entrenando un modelo con un gran volumen de información; se detallara la aceleración de las pruebas y resultados en el capítulo 5.

**Figura 53**

Utilizando el algoritmo de cuml Regresión Logística



*Nota.* Utilizando el algoritmo de cuml Regresión Logística [Gráfico], por Autor, 2020.

La figura 53 representa la demostración y utilización del algoritmo Regresión Logística, en donde se verifica que, por medio de la biblioteca cuml se puede utilizar algoritmos Machine Learning. Se crea un nuevo DataFrame de cudf (cudf.DataFrame) que contiene los datos que se dividio, se categorizo y va a corresponder a un nuevo DataFrame de entrenamiento, seguidamente creamos un cudf Series (cudf.Series), en este caso corresponde a una Serie de data de prueba, este proceso se realiza para poder entrenar el algoritmo de Regresión Logística; se destaca la asociación de la librería cupy, la librería permite crear GPUs arrays a cada dato del nuevo DataFrame y Serie, esto permitirá que, el entrenamiento del algoritmo sea más rápido, por último, se llama al algoritmo y se incorpora las configuraciones de entrenamiento que dispone el mismo.

A continuación unas tablas en donde se detalla las configuraciones y las características para utilizar el algoritmo Regresión Logística de cuml.

**Tabla 18**

*Desarrollo y configuración para utilizar el algoritmo Regresión Logística*

|  |  |  |  |
| --- | --- | --- | --- |
| **Biblioteca** | **Característica** | **Descripción** | **Rapids** |
| cuml | cuml.linear\_model | Modelo Lineal | Stable 0.14 |
| cuml | LogisticRegression | Algoritmo Regresión Logística de cuml | Stable 0.14 |
| cudf | cudf.DataFrame | Creación de un DataFrame | Stable 0.14 |
| cudf | cudf.Series | Creación de una Serie | Stable 0.14 |
| cupy | cp.asarray | Crear matrices de GPU arrays | Stable 0.14 |
| cupy | cp.float32 | Tipo dato del GPU array sea float32 | Stable 0.14 |

Nota: Asociación y utilización de las bibliotecas, librerías que se utilizaron para comenzar a entrenar el modelo con el algoritmo Regresión Logística de cuml.

**Tabla 19**

*Características del algoritmo Regresión Logística de cuml*

|  |  |  |
| --- | --- | --- |
| **Algoritmo RL** | **Descripción** | **Rapids** |
| fit\_intercept | Si fit\_intercept es true, el algoritmo intenta corregir la media global de la data de prueba. | Stable 0.14 |
| max\_iter | Número máximo de interaciones tomadas para que los solucionadores converjan. | Stable 0.14 |
| coef\_.to\_output(‘cupy’) | Resultado de todos los coeficientes cuando ya termino de entrenar el algoritmo, el resultado en formato de matriz GPU array. | Stable 0.14 |
| intercept\_.to\_output(‘cupy’) | Resultado de las intercepciones que existe en la data de entrenamiento y pruebas, el resultado en formato de matriz GPU array. | Stable 0.14 |
| predict | Resultados de la predicción del algoritmo. | Stable 0.14 |

Nota: Características que se utilizaron en el entrenamiendo del Algoritmo de Regresión Logística de cuml.

En el siguiente apartado de visualización, se dará una demostración de la biblioteca de Rapids cuxfilter, para mostrar los datos de forma gráfica.

## **Visualización de Gráficos**

La visualización permite que Rapids presente los datos como un gráfico interactivo para facilitar la identificación y poder comprender conceptos difíciles, los usuarios, desarrolladores, ingenieros y científicos de datos por médio de las gráficas pueden crear y desarrollar nuevas ideas. Dentro del proceso de desarrollo de los gráficos se ha analizado, estudiado e implementado dos tipos de gráficos estadísticos de barras y de líneas, a continuación en los siguientes puntos se detalla el proceso de desarrollo de los gráficos utilizando la biblioteca cuxfilter de Rapids.

### *Gráficos Estadísticos de Barras con Cuxfilter*

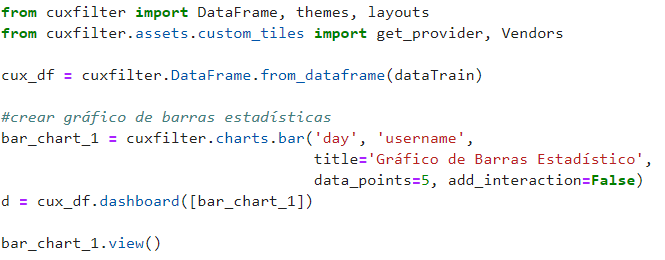
Permite poder representar de forma visual (en gráficos) las distintas interacciones de los datos con el usuario, examinar los mismos y definir nuevas ideas para generar un valor significativo a los datos que se representan de forma gráfica, el gráfico de barras o diagrama representa distintos tipos de datos, la altura es proporcional a la frecuencia de cada variable, las barras pueden ser horizontales y verticales, según si las variables se reflejan en el eje vertical o horizontal y está conformado por barras rectangulares.

Por medio de la biblioteca cuxfilter se puede crear, modificar y generar un gráfico de barras estadístico que sea dinámico e interactivo, permite representar los datos de manera acelerada, es decir, cuando se esta manejando grandes volúmenes de información y se requiere graficar dicha información dependiendo de las condiciones de cada usuario cuxfilter construye el gráfico de una forma muy rápida y las operaciones se realizan en la propia GPU.

Para el desarrollo del gráfico de barras aplicando la biblioteca cuxfilter se baso en la construcción y configuración utilizando los módulos de la biblioteca que fueron explicados en la sección 2.3.6.

**Figura 54**

Creación del gráfico estadístico de Barras



*Nota.* Creación del gráfico estadístico de Barras [Gráfico], por Autor, 2020.

La figura 54 representa la creación y desarrollo del gráfico estadístico de barras, importamos las clases y funcionalidades que van a permitir crear el gráfico, creamos un nuevo DataFrame cuxfilter de un DataFrame cudf, cabe recalcar que el DataFrame cudf ya realizo un proceso previo en donde cumplio la división de la data en train, test y un preprocesamiento utilizando la biblioteca cuml en la figura 53, seguidamente se construye el gráfico de barras especificando las características, los ejes X y Y que va a contener el mismo, por último se crea el gráfico y se reliza una vista previa.

A continuación en la tabla siguiente se detalla las características que se utilizaron para crear el gráfico de barras utilizando cuxfilter.

**Tabla 20**

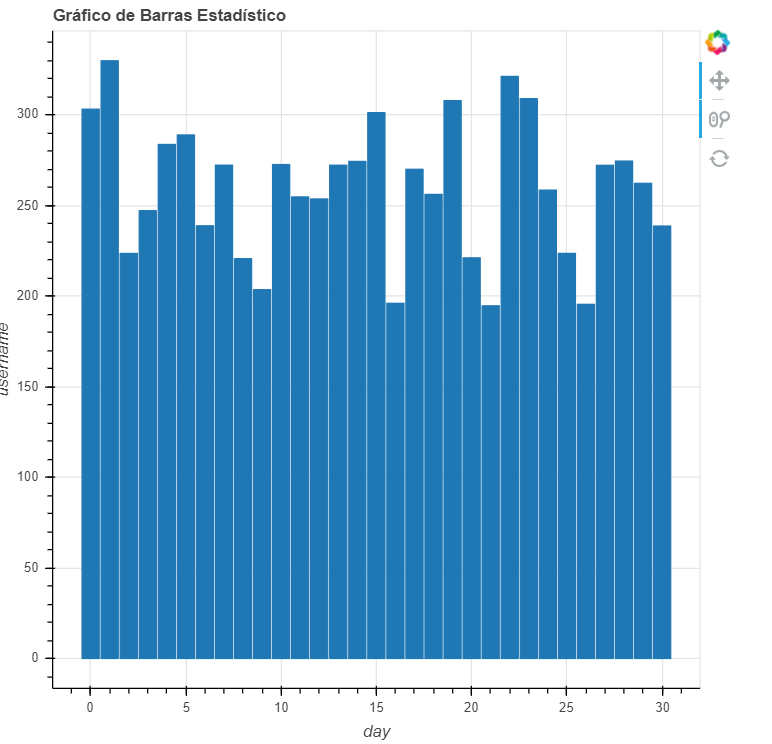
*Características de cuxfilter para la creación del gráfico de barras*

|  |  |  |
| --- | --- | --- |
| **Cuxfilter** | **Descripción** | **Rapids** |
| cuxfilter.DataFrame.from\_dataframe | Convertir de un DataFrame cudf a un nuevo DataFrame cuxfilter | Stable 0.14 |
| cuxfilter.charts.bar | Especificar el tipo de gráfico que se va a graficar | Stable 0.14 |
| dashboard | Construcción del gráfico | Stable 0.14 |
| view | Vista previa del gráfico creado. | Stable 0.14 |
| Title | Título que va a tener el gráfico | Stable 0.14 |
| data\_points | Puntos de datos que va a tener el gráfico | Stable 0.14 |
| add\_interaction | Agregar la interacción predeterminado debe ser valor True | Stable 0.14 |

Nota: Características que permite crear un gráfico estadístico de barras utilizando la biblioteca cuxfilter.

**Figura 55**

Gráfico estadístico de Barras



*Nota.* Gráfico estadístico de Barras [Gráfico], por Autor, 2020.

La figura 55 representa el gráfico de barras creado por cuxfilter, se puede visualizar la cantidad de usuarios conectados por día en la plataforma de OpenCampus.

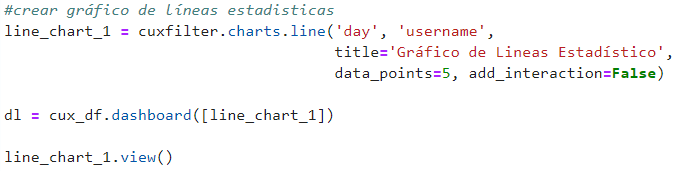
### *Gráficos Estadísticos de Líneas con Cuxfilter*

Permite poder representar de forma visual las interacciones de los datos con el usuario, examinar los mismos y definir nuevas ideas para generar un valor significativo a los datos, el gráfico de líneas se compone de una serie de datos representados por puntos y unidos por segmentos lineales.

La biblioteca cuxfilter permite crear, modificar y generar un gráfico de líneas estadístico que sea dinámico e interactivo, presenta los datos de forma gráfica con mucha rapidez, es decir, cuando se esta manejando grandes volúmenes de información y se requiere graficar dicha información en un gráfico estadístico construye el mismo de forma rápida y las operaciones se realizan en la propia GPU.

**Figura 56**

Creación del gráfico estadístico de Líneas



*Nota.* Creación del gráfico estadístico de Líneas [Gráfico], por Autor, 2020.

La figura 56 representa la creación y desarrollo del gráfico estadístico de líneas, se construye el gráfico de barras especificando las características, los ejes X y Y que va a contener el mismo, por último se crea el gráfico y se reliza una vista previa.

A continuación en la tabla siguiente se detalla las características que se utilizaron para crear el gráfico de líneas utilizando cuxfilter.

**Tabla 21**

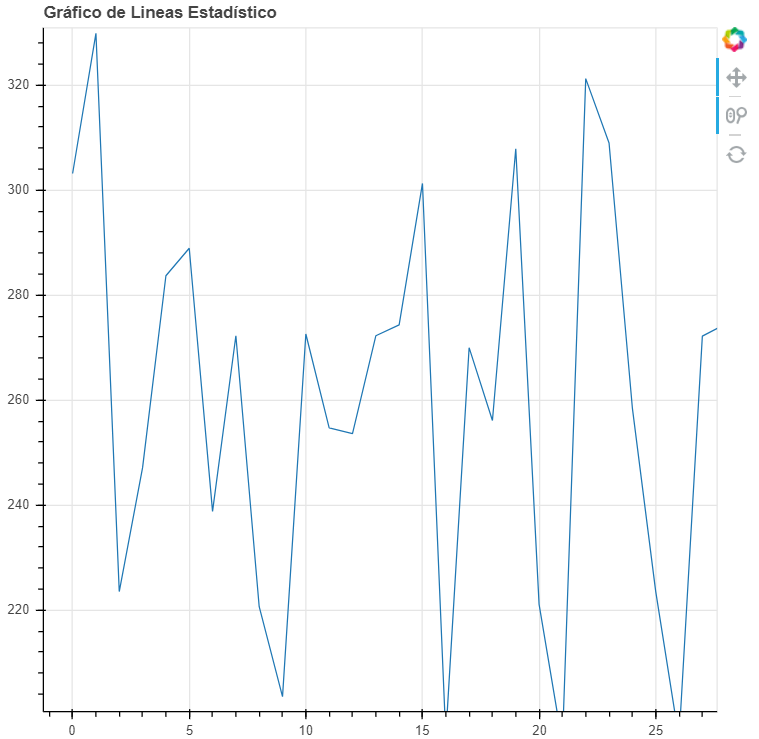
*Características de cuxfilter para la creación del gráfico de líneas*

|  |  |  |
| --- | --- | --- |
| **Cuxfilter** | **Descripción** | **Rapids** |
| cuxfilter.charts.line | Especificar el tipo de gráfico que se va a graficar en este caso es de líneas | Stable 0.14 |
| dashboard | Construcción del gráfico | Stable 0.14 |
| view | Vista previa del gráfico creado. | Stable 0.14 |
| title | Título que va a tener el gráfico | Stable 0.14 |
| data\_points | Puntos de datos que va a tener el gráfico | Stable 0.14 |
| add\_interaction | Agregar la interacción predeterminado debe ser valor True | Stable 0.14 |

Nota: Características y las funcionalidades para crear un gráfico estadístico de líneas utilizando la biblioteca cuxfilter.

**Figura 57**

Gráfico estadístico de Líneas



*Nota.* Gráfico estadístico de Líneas [Gráfico], por Autor, 2020.

La figura 57 representa el gráfico de líneas creado por cuxfilter, se puede visualizar la cantidad de usuarios conectados por día en la plataforma de OpenCampus.

## **Construcción y Desarrollo con Librerías Tradicionales para Ciencia de Datos.**

### *Implementación con Pandas*

### *Implementación con Scikit-Learn*

### *Implementación con Matplotlib*

# Conclusiones

Con el análisis expuesto de la presente investigación de Ciencia de Datos y Rapids, se ha destacado la importancia que tiene la GPU, revela detalles de trabajo que adopta en ambientes de proceso exploratorio de datos por medio de las bibliotecas que tiene la librería, además, apoya en mejorar el futuro proceso en ambientes de ciencias de datos. El objetivo es dar a conocer el mejor funcionamiento de Rapids, para poder cosechar toda su potencia de trabajo e implementación.

Rapids permite trabajar de manera rápida, eficaz, eficiente y de forma acelerada de extremo a extremo utilizando los beneficios de la GPU (aplicando las arquitectura de Nvidia), el conjunto de bibliotecas de Rapids perfecciona el flujo de analizar, procesar, trabajar y solucionar problemas; brinda a los desarrolladores y científicos de datos trabajar de manera acelerada y que los tiempos de cada proceso son rápidos cuando se está trabajando con grandes volúmnes de información, progresar en el estudio para aumentar el análisis exploratorio y trabajos con Inteligencia Artificial (Machine Learning y Deep Learning).

# Recomendaciones

Sugerencias para posibles investigaciones que surgieren del estudio realizado.

# Referencias

Aguerzame, A., Pelletier, B., & Waeselynck, F. (2019). GPU Acceleration of PySpark using RAPIDS AI. *DATA 2019 - Proceedings of the 8th International Conference on Data Science, Technology and Applications*, *Data*, 437–442. https://doi.org/10.5220/0008191404370442

Allen, R., & US, B. R. (2020). *Security Alert Analysis Using GPUs.* https://medium.com/rapids-ai/security-alert-analysis-using-gpus-1a31270aa85e

Alpaydin, E. (2009). *Introduction to machine learning*.

Aramburo, R. (2019). BlazingSQL Parte 1: El GPU DataFrame (GDF) y cuDF en RAPIDS AI. *CuDF — GPU Data Processing for GDFs*, *0*(0). https://blog.blazingdb.com/blazingsql-part-1-the-gpu-dataframe-gdf-and-cudf-in-rapids-ai-96ec15102240

Arrow, A. (2016). *The Apache Software Foundation*. https://arrow.apache.org/

B, S. L., Chu, R. S. W., Wang, X., & Luk, W. (2019). *Image Classification on FPGAs* (Vol. 1, Issue 16). Springer International Publishing. https://doi.org/10.1007/978-3-030-17227-5

Castro, J. A. (2018). *Metodología de reduccion de dimensión de tipo espectral con representación interactiva de datos*. 28. http://bdigital.unal.edu.co/64456/1/1127938442.2018.pdf

Chainer. (2019). *A flexible framework for neural networks*. https://chainer.org/

Crist, J. (2016). Dask & Numba: Simple libraries for optimizing scientific python code. *Proceedings - 2016 IEEE International Conference on Big Data, Big Data 2016*, 2342–2343. https://doi.org/10.1109/BigData.2016.7840867

El, E. N., Cuda, E., Aplicaciones, E. N., & Tecnolog, N. C. (2015). *Segunda Parte : TECNOLOGÍA CUDA*. 12–26.

Enemark, A. (2018). *Accelerating Cross Filtering with cuDF*. https://medium.com/rapids-ai/accelerating-cross-filtering-with-cudf-3b4c29c89292

Estrada, J. C. ., Silva, I. A. ., & Paéz, J. O. . (2018). Big Data: Ventajas y desventajas-aplicaciones y tecnologías para implementar el servicio. *COMITÉ CIENTÍFICO CICOM 2018*, *0*(0).

Garre, M., Cuadrado, J. J., Sicilia, M. A., Rodríguez, D., & Rejas, R. (2007). Comparación de diferentes algoritmos de clustering en la estimación de coste en el desarrollo de software. *Revista Espa？ola de Innovación, Calidad e Ingeniería Del Software*.

Guim, F., & Rodero, I. (2019). *Arquitecturas basadas en computación gráfica (GPU)*. *0*(0), 68.

Howard, K., Siddharth, S., & Sergei, V. (2010). Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms. *A Model of Computation for MapReduce*, *0*(0).

Huang, J., Lin, Z., Ma, C., & Yuan, X. (2013). *GPU SPEED-UP FOR THE IMPLICIT NAVIER-STOKES SOLVER*. *0*(0), 10.

Joyones Aguilar, L. (2016). *Big Data, Análisis de grandes volúmenes de datos en organizaciones*. https://books.google.com.ec/books?hl=es&lr=&id=1GywDAAAQBAJ&oi=fnd&pg=PT6&dq=Aguilar,+L.+J.+(2016).+Big+Data,+Análisis+de+grandes+volúmenes+de+datos+en+organizaciones.+Alfaomega+Grupo+Editor.&ots=\_WU9L37j\_Q&sig=YPC8eRaPaRv56HxrbY01I8IoRmk&redir\_esc=y#v=on

Keller, C. A., Clune, T. L., Thompson, M. A., Stroud, M. A., Evans, M. J., & Ronaghi, Z. (2019). *Accelerated simulation of air pollution using NVIDIA RAPIDS*. *November*, 4–6.

López García, D. (2012). Análisis de las posibilidades de uso de Big Data en las organizaciones. *Otros Conceptos Relacionados Con Big Data*, *0*(0), 18–62.

Mansingh, G., Osei-Bryson, K. M., Rao, L., & McNaughton, M. (2017). Data preparation: Art or science? *Proceedings of the 2016 International Conference on Data Science and Engineering, ICDSE 2016*. https://doi.org/10.1109/ICDSE.2016.7823936

Matsuoka, S., Aoki, T., Endo, T., Nukada, A., Kato, T., & Hasegawa, A. (2009). GPU accelerated computing–from hype to mainstream, the rebirth of vector computing. *Commoditization of HPC and Its Acceleration, but Niche Still Remains .*, 11.

MXNET. (2019). *A FLEXIBLE AND EFFICIENT LIBRARY FOR DEEP LEARNING*. https://mxnet.apache.org/

Nolet, C. (2019). *Combining Speed & Scale to Accelerate K-Means in RAPIDS cuML*. https://medium.com/rapids-ai/combining-speed-scale-to-accelerate-k-means-in-rapids-cuml-8d45e5ce39f5

Numba. (2019). *Numba makes Python code fast*. https://numba.pydata.org/

NVIDIA. (2009). NIVDIA CUDA Architecture. *NVIDIA CUDA Architecture Introduction & Overview*, *0*(0), 9.

Owens, J. D., Houston, M., Luebke, D., Green, S., Stone, J. E., & Phillips, J. C. (2008). *GPU Computing*. *0*(0), 18.

Patterson, J. (2019). *The Platform Inside and Out*.

Peltenburg, J., van Straten, J., Wijtemans, L., van Leeuwen, L., Al-Ars, Z., & Hofstee, P. (2019). *Fletcher: A Framework to Efficiently Integrate FPGA Accelerators with Apache Arrow*. 270–277. https://doi.org/10.1109/fpl.2019.00051

Pérez Represa, C., Cámara Nebreda, J. M., & Sánchez Ortega, P. L. (2016). INTRODUCCIÓN A LA PROGRAMACIÓN EN CUDA. *INTRODUCCIÓN A LA PROGRAMACIÓN EN CUDA*, *0*(0), 74.

Puneet, G. (2018). Speed Up your Algorithms Part 2— Numba. *Get C++/Fortran like Speed for Your Functions with Numba*. https://towardsdatascience.com/speed-up-your-algorithms-part-2-numba-293e554c5cc1

PyTorch. (2019). *FROM RESEARCH TO PRODUCTION*. https://pytorch.org/

Rabhi, S., Sun, W., Perez, J., Kristensen, M. ., Liu, J., & Oldridge, E. (2019). Accelerating recommender system training 15x with RAPIDS. *In Proceedings of the Workshop on ACM Recommender Systems Challenge.*

RAPIDS Development Team. (2018). *RAPIDS: Collection of Libraries for End to End GPU Data Science*. https://rapids.ai

Rees, B. (2019). *RAPIDS cuGraph*. https://medium.com/rapids-ai/rapids-cugraph-1ab2d9a39ec6

Rhodes, B. (2019). Análisis acelerado de GPU de Cyber ​​Log con RAPIDS. *Las Operaciones de Seguridad (SecOps) y Los Departamentos de TI Están Recopilando, Gestionando e Intentando Analizar Más Datos Que Nunca. Es Probable Que Los Empleados Conecten Sus Propios Dispositivos a Las Redes Corporativas, Ampliando Aún Más Una Super*. https://medium.com/rapids-ai/gpu-accelerated-cyber-log-parsing-with-rapids-10896f57eee9

Shilpi, S., & Saurabh, G. (2001). Practical Real-Time Data Processing and Analytics. In *Assiut Journal of Environmental Studies*.

Shohei, H. (2016). Complex neural networks made easy by Chainer. *A Define-by-Run Approach Allows for Flexibility and Simplicity When Building Deep Learning Networks.* https://www.oreilly.com/content/complex-neural-networks-made-easy-by-chainer/

Srinath, A., & Kraus, K. (2019). RAPIDS and cuDF: Accelerating DataFrames on GPUs. *Hadoop Processing, Reading from Disk. Spark In-Memory Processing. Traditional GPU Processing. RAPIDS.*, *0*(0), 46.

Team, D. D. (2016). *Dask: Library for dynamic task scheduling*. https://dask.org

Thi Yen, P., Deok-Young, L., & Jeong-Gun, L. (2017). Impacts of optimization strategies on performance, power/energy consumption of a GPU based parallel reduction. *GPU Architecture and CUDA*, *0*(0), 14.

Turner, J. (2011). *Hadoop: What it is, how it works, and what it can do*. *0*(0), 2.

Vishal, M. (2019). *Accelerating Random Forests up to 45x using cuML*. https://medium.com/rapids-ai/accelerating-random-forests-up-to-45x-using-cuml-dfb782a31bea

White, T. (2015). *Hadoop The Definitive Guide Fourth Edition* (Vol. 0, Issue 0).

# Apéndice

Se incluye de acuerdo al orden citado en el cuerpo del Trabajo de Titulación.

**Apéndice 1:**

**Apéndice 2:**

**Apéndice 3:**

1. Machine Learning: Aprendizaje Automático [↑](#footnote-ref-1)
2. Big Data: Grandes Datos [↑](#footnote-ref-2)
3. IBM: International Business Machines [↑](#footnote-ref-3)
4. HTML: HyperText Markup Language [↑](#footnote-ref-4)
5. XML: Extensible Markup Language [↑](#footnote-ref-5)
6. Apache Lucene: Una api de código abierto para la recuperación de información. [↑](#footnote-ref-6)
7. Apache Nutch: Es un robot y motor de búsqueda. [↑](#footnote-ref-7)
8. UPS: Sistema de alimentación Interrumpida [↑](#footnote-ref-8)
9. HDFS Read: Archivos de lectura. [↑](#footnote-ref-9)
10. Query: Sentencias SQl. [↑](#footnote-ref-10)
11. HDSF Write: Archivos de escritura. [↑](#footnote-ref-11)
12. RDD: Conjunto de datos distribuidos. [↑](#footnote-ref-12)
13. DAG: Gráfico acíclico dirigido. [↑](#footnote-ref-13)
14. PC: Computadora Personal [↑](#footnote-ref-14)
15. GPU Read: Leer archivos en la GPU. [↑](#footnote-ref-15)
16. CPU Write: Escribir archivos en la CPU. [↑](#footnote-ref-16)
17. 3D: Tridimensional. [↑](#footnote-ref-17)
18. 3dfx Interactive: Empresa especializada en la manufactura de procesadores gráficos 3D. [↑](#footnote-ref-18)
19. Many-core:Procesadores Manycore. [↑](#footnote-ref-19)
20. SFU: Unidades de funciones especiales. [↑](#footnote-ref-20)
21. DDR: Tipo de memoria RAM. [↑](#footnote-ref-21)
22. Speedup: La aceleración mide el rendimiento relativo de dos sistemas que procesan el mismo problema [↑](#footnote-ref-22)
23. IEEE: Instituto de Ingeniería Eléctrica y Electrónica [↑](#footnote-ref-23)
24. Nvidia Cuda: Compute Unified Device Architecture [↑](#footnote-ref-24)
25. Deep Learning: Aprendizaje Profundo [↑](#footnote-ref-25)
26. API: Interfaz de progrmación de Aplicaciones [↑](#footnote-ref-26)
27. GoAi: GPU Open Analytics Initiative. [↑](#footnote-ref-27)
28. FPS: Fotogramas por Segundo. [↑](#footnote-ref-28)
29. SMD: Instrucción Única, datos múltiples [↑](#footnote-ref-29)
30. Word2Vec: Grupo de modelos de relaciones que se utilizan para producir incrustaciones de palabras. [↑](#footnote-ref-30)
31. NetworkX: Biblioteca de Python para el estudio de gráficos y análisis de redes. [↑](#footnote-ref-31)
32. OpenUCX: Marco de Comunicaciones. [↑](#footnote-ref-32)
33. Gluon: Bosón de interacción nuclear fuerte. [↑](#footnote-ref-33)
34. PNL: Programación Neurolingüistica. [↑](#footnote-ref-34)
35. LLVM: Infraestrutura para desarrollar compiladores. [↑](#footnote-ref-35)
36. Intel Xeon Hasweel: Microarquitectura de procesadores desarrollado por Intel. [↑](#footnote-ref-36)
37. XGBoost: Librería de Software de código abierto que se implementa en minería de datos para clasificar y pronosticar sobre uma variable Y. [↑](#footnote-ref-37)
38. BullSequana: Servidores. [↑](#footnote-ref-38)
39. CLX: Biblioteca que ofrece llamadas de bajo nivel. [↑](#footnote-ref-39)
40. Regex: Denotan expresiones regulares que se utilizán en la informática teórica. [↑](#footnote-ref-40)
41. Splunk: Software para buscar, monitorizar y analizar macrodatos generados por máquinas de aplicaciones, sistemas e infraestructura IT a través de una interfaz web. [↑](#footnote-ref-41)