# Implementación de un Ambiente de Ciencia de Datos a traves de la Librería Open Source RAPIDS

1st Guarnizo Romero José Alberto
*Departamento De Ciencias de la Computación y Telecomunicaciones*
*Universidad Técnica Particular de Loja*
Loja, Ecuador
jaguarnizo4@utpl.edu.ec

2nd Elizalde Solano René Rolando
*Departamento De Ciencias de la Computación y Telecomunicaciones*
*Universidad Técnica Particular de Loja*
Loja, Ecuador
rrelizalde@utpl.edu.ec

***Abstract*—RAPIDS ejecuta canalizaciones de datos y análisis de extremo a extremo utilizando las principales características de la GPU de NVIDIA, la librería maneja grandes volúmenes de información en tiempo real y su principal enfoque es realizar una analítica de la información, utilizando el proceso de extracción, transformación y carga e implementar un conjunto de algoritmos Machine Learning; se da realce a sus principales funcionalidades y su conjunto de bibliotecas, además, se utilizó la plataforma gratuita cloud BlazingSQL para la implementación, donde resalta el proceso de construcción de un ambiente de ciencia de datos utilizando RAPIDS, se implemento una contraparte utilizando librerías que aún no tiene soporte para GPU como Pandas, Scikit-learn y Matplotlib, la comparación con las librerías tradicionales permite resaltar la aceleración que tiene RAPIDS cuando trabaja con grandes volúmenes de información.**

***Index Terms*—RAPIDS, GPU, ETL, Machine Learning, NVIDIA**

## I. Introducción

Este documento se enfoca en la implementación de un ambiente de ciencia de datos a través de la Librería Open Source RAPIDS para analizar un conjunto de datos abiertos utilizando las principales características de la librería. Mediante un análisis completo de RAPIDS permitirá manejar grandes volúmenes de datos en tiempo real y hacer uso de las características más importantes de la misma que permiten: tiempos mínimos de cada proceso y capacidad para el análisis, procesamiento e implementación de algoritmos Machine Learning y que los tiempos de cada proceso sean acelerados.

## II. Marco Teórico

### A. Librería Open Source RAPIDS

La librería RAPIDS brinda la capacidad de ejecutar canalizaciones de datos y análisis de extremo a extremo en GPU (Unidad de procesamiento gráfico), se enfoca en tareas comunes de preparación de datos y ciencia de datos.

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.

A continuación se datalla las principales bibliotecas de RAPIDS que se utilizaron para la implementación de la librería, dentro de ello se especifica las deficiones de cada una.

*1) Biblioteca cuDF:* cuDF es la biblioteca de manipulación de DataFrames basada en Apache Arrow, está enfocada en la preparación de datos, acelera la carga, el filtrado y la manipulación de la información, contiene la definición del modelo de memoria gdf [1].

Permite la manipulación de datos como concatenación, funciones aritméticas y clasificación, unir, agrupar, agregar y operaciones adicionales con los usuarios que están familiarizados con Pandas, Numpy o SQL.

*2) Biblioteca cuML:* cuML es una biblioteca que implementan algoritmos machine learning aceleradas por GPU, permite a los científicos de datos ejecutar tareas tradicionales de machine learning en GPU, cumpliendo un proceso de división de datos, entrenamiento y prueba.

*3) Biblioteca Cuxfilter:* Es un framework de RAPIDS que conecta visualizaciones web a filtros cruzados por GPU, realiza un filtrado interactivo y rápido de más de 100 millones de conjuntos de datos a través de cuDF.

Cuxfilter permite que los datos se mantengan 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 [2].

### B. Servicios desplegados en la nube con RAPIDS

La librería RAPIDS ahora se implementa en los servicios cloud de pago, lo que permite poder aprovechar los aumentos de velocidad y reducciones de tiempos para los procesos de ETL, ML y visualización.

La implementación cloud requiere que se utilice instancias GPU, como cada proveedor cloud tiene instancias GPU esto

facilita poder utilizar RAPIDS ya que las mismas son compatibles con la librería en diferentes capacidades y precios. Los principales provedores donde se puede utilizar RAPIDS son: Amazon Web Services, Microsoft Azure y Google Cloud.

## III. Implementación con RAPIDS

### A. Análisis de la Arquitectura RAPIDS

RAPIDS trabaja de forma rápida, eficiente y eficaz, debe 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 exploratorio y procesamiento de los datos.
- Una vez realizado el proceso de ETL los datos ya se encuentran totalmente limpios; se procede a realizar el proceso ML que consiste en dividir todo el dataset en data de entrenamiento y data de prueba, se utiliza la biblioteca cuML, seguidamente se realiza un proceso de preprocesamiento de los datos, es decir, cambiar a datos categóricos, con el fin de poder utilizar algoritmos machine learning.
- Para los algoritmos machine learning lo realiza la biblioteca cuML, 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 sea más rápido trabajando de manera paralela.
- Visualización, cuxfilter para la respectiva representación gráfica en forma de gráficos estadísticos.

### B. Arquitectura RAPIDS

La arquitectura de RAPIDS propuesta está en un ambiente dedicado al entorno de ciencia de datos (la misma se especificó en el punto anterior), está utilizando la plataforma gratuita BlazingSQL que es un medio para utilizar la librería, cumple el flujo de trabajo para los procesos de ETL, ML, visualización y que bibliotecas de RAPIDS cumple cada uno de estos procesos.

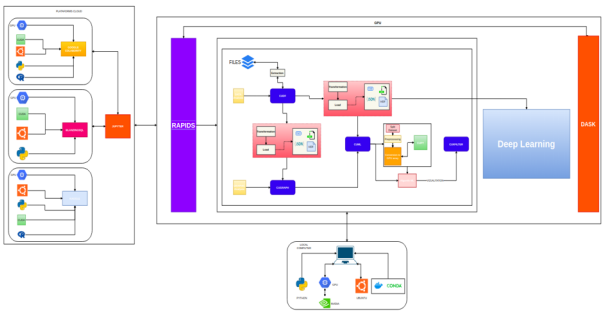


Fig. 1. Arquitectura RAPIDS.

En el siguiente punto se detalla el proceso de instalación en la plataforma cloud BlazingSQL.

*1) Plataforma BlazingSQL:* BlazingSQL ejecuta demostraciones libres que se encuentra en un entorno de trabajo JupyterLab y es de uso gratuito, permite ejecutar rapidamente BlazinSQL + RAPIDS, acelera el procesamiento de datos, es un motor SQL acelerado por GPU construido 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. El procedimento para utilizar la plataforma BlazingSQL, es el siguiente:

- Ingresar a su página principal: https://blazingsql.com/
- Seguidamente logearse con una cuenta de Gmail.
- Por último se procede a selecionar el cuaderno de trabajo de RAPIDS, es importante utilizar los cuadernos estables que actualmente se encuentra en la versión 0.15.

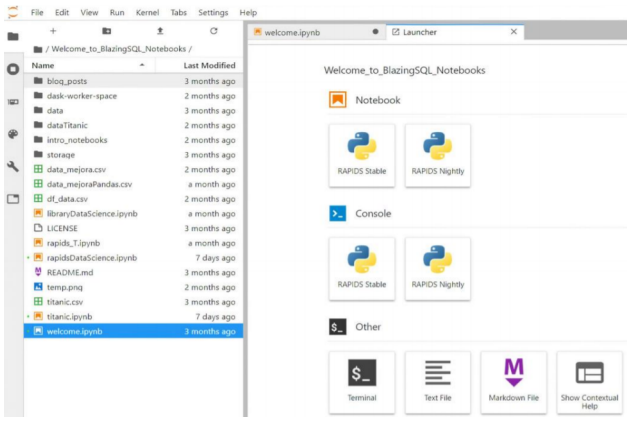


Fig. 2. Plataforma BlazingSQL + RAPIDS.

Los beneficios de utilizar BlazingSQL es que no se necesita realizar una instalación de paquetes de RAPIDS, permite escribir, ejecutar código en python en un navegador como google chrome, la plataforma ya tiene preinstalada toda la librería, conjuntamente con otras librerías adicionales como pandas, dask, cupy, entre otras.

BlazingSQL obtiene cuadernos de ejemplos Jupyter utilizando RAPIDS que ayuda a entender el funcionamento de la librería. Proporciona una GPU Tesla T4 integrada que ayuda a mejorar el rendimiento en los flujos de trabajo para ETL, ML y visualización de los datos.

Cabe recalcar que durante los dos últimos años desde que se construyo RAPIDS (se desarrollo en el 2018) ha fomentado un ecosistema en constante crecimiento de bibliotecas y herramientas todo esto acelerado por GPU, es asombroso ver la evolución de RAPIDS gracias a la gran comunidad que contribuye con el proyecto Open Source.

Se destaca que RAPIDS y BlazingSQL ya estan asociados en su totalidad (se puede instalar de manera local en el computador personal) todo esto para mejorar la experiencia

de usuario, con el fin de perfeccionar la estabilidad, velocidad y rendimiento de la librería RAPIDS.

*2) Proceso de Extracción, Transformación y Carga:* En la fase de extracción se trabajo con datos de OpenCampus (Comunidad global de aprendizaje de cursos abiertos), la data contiene 200.000 filas de información, los mismos no contenían un esquema fijo, no tenían ningun tratamiento y no era entendible.

| | u | n | o |
|---|---|---|---|
| 0 | /edx/var/log/tracking/tracking.log-20171110-15... | null | null |
| 1 | /edx/var/log/tracking/tracking.log-20171110-15... | null | null |
| 2 | /edx/var/log/tracking/tracking.log-20171110-15... | null | null |
| 3 | /edx/var/log/tracking/tracking.log-20171110-15... | null | null |
| 4 | /edx/var/log/tracking/tracking.log-20171110-15... | null | null |
| ... | ... | ... | ... |
| 199995 | /edx/var/log/tracking/tracking.log-20180110-15... | null | null |
| 199996 | /edx/var/log/tracking/tracking.log-20180110-15... | null | null |
| 199997 | /edx/var/log/tracking/tracking.log-20180110-15... | null | null |
| 199998 | /edx/var/log/tracking/tracking.log-20180110-15... | null | null |
| 199999 | /edx/var/log/tracking/tracking.log-20180110-15... | null | null |

200000 rows × 3 columns

Fig. 3. Salida de los datos extraídos y leídos.

La fase de transformación de los datos realiza un análisis exploratorio de toda la información, donde se procede 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.

| | username | event_source | host | referer | ip | l | year | month | day | hour |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | Amada1012 | server | opencampus.utpl.edu.ec | http://opencampus.utpl.edu.ec/courses/course-v... | 190.57.140.77 | 9844 | 2017 | 12 | 19 | 04:23 |
| 1 | Amada1012 | browser | opencampus.utpl.edu.ec | http://opencampus.utpl.edu.ec/courses/course-v | 190.57.140.77 | 9844 | 2017 | 12 | 19 | 04:23 |

Fig. 4. Visualización de la Data Transformada.

*3) Proceso Machine Learning:* El proceso machine learning permite que un sistema, por sí mismo aprenda en forma automatizada, a descubrir patrones, tendencias y relaciones con los datos, ofrece mejores perspectivas, a continuación se explica por medio de gráficos el proceso de machine learning utilizando la biblioteca cuML de RAPIDS, todo el proceso se realizó en la plataforma cloud BlazingSQL.

```
# Fase dividir Dataset
from cuml.preprocessing.model_selection import train_test_split

# Asignar dataset general a nuevas variables
gdf_dataX = DataMejorada
gdf_dataY = DataMejorada

# Dividir dataset en train y test
# datatrain 80% y datatest 20%
X_train, X_test, y_train, y_test = train_test_split(gdf_dataX,
gdf_dataY,

train_size=0.8)
```

```
rom cuml.preprocessing.LabelEncoder import LabelEncoder
e = LabelEncoder()

Convertir a variables categóricas
seleccionar columnas potenciales para train de xTrain
alue_usernameX = le.fit_transform(X_train.username)
alue_EventSourceX = le.fit_transform(X_train.event_source)
alue_DayX = le.fit_transform(X_train.day)

seleccionar columnas potenciales para test de xTest
alue_usernameXT = le.fit_transform(X_test.username)
alue_EventSourceXT = le.fit_transform(X_test.event_source)
alue_DayXT = le.fit_transform(X_test.day)

seleccionar columnas potenciales para train de yTrain
alue_usernameY = le.fit_transform(y_train.username)
alue_EventSourceY = le.fit_transform(y_train.event_source)
value_DayY = le.fit_transform(y_train.day)

# seleccionar dos columnas potenciales para test de yTest
value_usernameYT = le.fit_transform(y_test.username)
value_EventSourceYT = le.fit_transform(y_test.event_source)
value_DayYT = le.fit_transform(y_test.day)
```

Fig. 5. Dividir la data en Train, Test y Transformar a datos categóricos.

```
# Fase Uso de los algoritmos ML de cuml: Algoritmo
LinearRegression
import cupy as cp
from cuml.linear_model import LinearRegression

# Creación de un nuevo dataframe Train
dataTrain = cudf.DataFrame()
dataTrain['username'] = cp.asarray(value_usernameX,
dtype=cp.float32)
dataTrain['day'] = cp.asarray(value_DayX, dtype=cp.float32)

# Creación de un nuevo dataframe Test
new_value = cp.asarray(value_usernameY, dtype=cp.float32)
dataTest = cudf.Series(new_value)

# Entrenando el algoritmo
lr = LinearRegression(fit_intercept = True, normalize =
False,algorithm = "eig")
reg = lr.fit(dataTrain, dataTest)
```

Fig. 6. Algoritmo Regresión Lineal de cuML.

*4) Construcción de gráficos con RAPIDS:* La visualización permite que RAPIDS presente los datos como un gráfico interactivo para facilitar la identificación y poder comprender conceptos difíciles; desarrolladores, ingenieros y científicos de datos por medio de las gráficas pueden crear y desarrollar nuevas ideas.

La figura 7 representa el ingreso a la plataforma OpenCampus en donde un número de usuarios han ingresado mediante el año, se destaca que en el año 2018 a tenido mayor demanda de ingreso por parte de los usuarios.

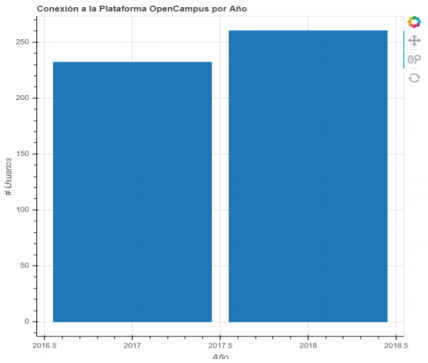


Fig. 7. Ingreso de usuarios a la plataforma OpenCampus por año.

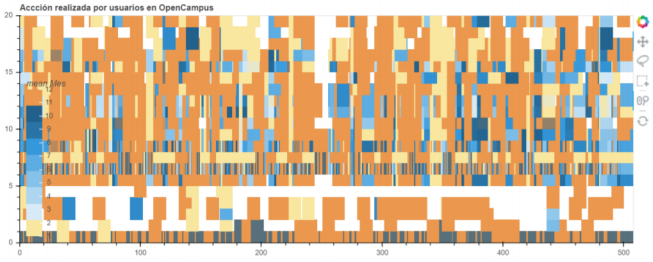


Fig. 8. Acción realizada por los Usuarios en la plataforma OpenCampus.

La figura 8 representa la mayor acción (reproducir un video, pausar un vídeo, cargar un video) que han realizado los usuarios dentro de la plataforma OpenCampus, se traza el resultado en un gráfico de mapa de calor para determinar

la mayor acción que han realizado y los meses en donde se registra más acciones realizadas, cabe destacar que en el mes de noviembre y diciembre tanto para los años 2017 y 2018 los usuarios han reproducido más vídeos y se representa de color anaranjado.

### *C. Implementación con Librerías Tradicionales para Ciencia de Datos.*

- Pandas: Se desarrollo todo el proceso de ETL utilizando el entorno de trabajo de la plataforma BlazingSQL y representa la contraparte de la biblioteca cuDF de RAPIDS.
- Scikit-Learn: La librería trabaja en ambiente CPU no tiene soporte para GPU, por medio de esta implementación con Scikit-Learn da un realce como contraparte de la biblioteca cuML de RAPIDS.
- Matplotlib: La librería trabaja en el ambiente CPU no tiene soporte para GPU, esta implementación da un realce como contraparte para la biblioteca cuxfilter de RAPIDS.

## IV. Pruebas y Resultados

Las pruebas y resultados obtenidos del presente detalla el poder de RAPIDS, se enfoca en acelerar todos los procesos mencionados con anterioridad (ETL, ML y visualización) a un dataset de la plataforma OpenCampus, la información comprende un registro de eventos que realizan los usuarios dentro de los años 2017 y 2018, la misma corresponde a información que no tienen formatos fijos (logs) y están compuestos por diferentes piezas de información.

TABLE I
Comparativa de Aceleración ETL con RAPIDS y Pandas

| Fase | Tiempo (s) | Datos |
|---|---|---|
| Extracción (cuDF-RAPIDS) | 0.70 s | 200.000 |
| Transformación (cuDF-RAPIDS) | 7.24 s | 200.000 |
| Carga (cuDF-RAPIDS) | 0.51 s | 200.000 |
| Extracción (PANDAS) | 3.17 s | 200.000 |
| Transformación (PANDAS) | 8.14 s | 200.000 |
| Carga (PANDAS) | 2.10 s | 200.000 |

TABLE II
Comparativa de Aceleración ML de preprocesamiento de datos RAPIDS y Scikit-Learn

| Fase | Tiempo (s) | Datos |
|---|---|---|
| Preprocesamiento (cuML-RAPIDS) | 0.13 s | 196.867 |
| Preprocesamiento (Scikit-Learn) | 0.14 s | 196.867 |

TABLE III
Comparativa de Aceleración algoritmos de entrenamiento ML con RAPIDS y Scikit-Learn

| Algoritmo ML | Tiempo (s) | Datos |
|---|---|---|
| Regresión Lineal (cuML-RAPIDS) | 0.009 s | 196.867 |
| Regresión Lineal (Scikit-Learn) | 0.018 s | 196.867 |

TABLE IV
Comparativa de Aceleración para visualización de gráficos con RAPIDS y Matplotlib

| Algoritmo ML | Tiempo (s) | Datos |
|---|---|---|
| Barras (cuxfilter-RAPIDS) | 4.25 s | 196.867 |
| Líneas (cuxfilter-RAPIDS) | 0.38 s | 196.867 |
| Barras (Matplotlib) | 191.22 s | 196.867 |
| Líneas (Matplotlib) | 3.38 s | 196.867 |

### *A. Explicación de los Resultados de Aceleración*

En base a los escenarios propuestos comparando los tiempos de aceleración de RAPIDS juntamente con las librerías tradicionales para ciencia de datos tanto en los procesos ETL, machine learning y visualización, destaca RAPIDS, acelera el flujo de extremo a extremo y cumple su principal objetivo de trabajar de forma acelerada utilizando las características de las GPUs de NVIDIA (GPU Tesla T4), esto da un realce significativo para utilizar la librería RAPIDS cuando se va a trabajar con grandes volúmenes de información.

Cabe destacar que RAPIDS siempre debe trabajar con gran número de información para que actúe la GPU con el fin de acelerar el flujo de trabajo principalmente para todos los procesos enfocados a la ciencia de datos como análisis exploratorio de la información, machine learning y visualización.

## References


[1] Rodrigo. A (2019, Feb 25). BlazingSQL Parte 1: El GPU DataFrame (GDF) y cuDF en RAPIDS AI (1nd ed.). [Online]. Available: https://blog.blazingdb.com/blazingsql-part-1-the-gpu-dataframe-gdf-and-cudf-in-rapids-ai-96ec15102240

[2] RAPIDS Development Team (2018). RAPIDS: Collection of Libraries for End to End GPU Data Science. [Online]. Available: https://rapids.ai