# Bosón de Higgs

Maximiliano de Jesús Galindo Hernández[1] and Alejandro Mendoza Puig[2]

[1] *Posgrado en Ciencia e Ingeniería de la Computación, Universidad Nacional Autónoma de México, Ciudad Universitaria Unidad de Posgrado, Edificio "C" 1er. nivel, Delegación Coyoacán, C.P. 04510, CDMX.*

Fecha de publicación: 28/11/2023

**Resumen**— En el ámbito de la física de partículas y altas energías, la ingente cantidad de datos generados en centros de investigación como el CERN suele plantear desafíos en su procesamiento y análisis. Este estudio se centró en comparar el procesamiento de datos de forma concurrente y secuencial, utilizando un conjunto emocionante de datos vinculados a los procesos relacionados con el descubrimiento del bosón de Higgs. Además, llevamos a cabo un análisis para estudiar el comportamiento de ciertas variables en presencia y ausencia del bosón de Higgs.

**Palabras Clave**— Bosón de Higgs, Concurrencia

## Introducción

El bosón de Higgs es una partícula fundamental en la teoría del Modelo Estándar de la física de partículas. Propuesta en la década de 1960 por Peter Higgs y otros físicos, su existencia explicaría cómo otras partículas elementales adquieren masa. Esta partícula es crucial para completar nuestra comprensión de las fuerzas fundamentales que rigen el universo.

En el Gran Colisionador de Hadrones (LHC) del CERN, ubicado cerca de Ginebra, Suiza, se llevaron a cabo experimentos para detectar el bosón de Higgs. Los científicos realizaron colisiones de protones a altas energías dentro del LHC, generando condiciones similares a las que se cree existían poco después del Big Bang. Tras años de análisis de datos provenientes de estas colisiones, en 2012 se anunció la detección del bosón de Higgs, un logro significativo para la física de partículas.

La detección del bosón de Higgs fue un hito científico crucial, pero su estudio continuo es fundamental para comprender mejor sus propiedades y su papel en el universo. Los investigadores en el CERN y otros lugares del mundo continúan analizando datos, utilizando técnicas avanzadas de análisis de datos y herramientas computacionales para estudiar con mayor detalle las características y comportamientos del bosón de Higgs.

El procesamiento de grandes volúmenes de datos provenientes de las colisiones en el LHC es un desafío monumental. Los métodos de análisis de datos se han vuelto cada vez más sofisticados, empleando técnicas de programación concurrente y algoritmos avanzados para identificar patrones que distingan entre eventos que producen un bosón de Higgs y aquellos que no lo hacen. Este enfoque combina la física teórica con la informática y la estadística para desentrañar los secretos del universo a nivel subatómico.

## Desarrollo

Para el desarrollo de nuestro proyecto seguimos los siguientes pasos:

### *Recolección de Datos*

Nuestra base de datos fue obtenida de la siguiente liga: https://archive.ics.uci.edu/dataset/280/higgs. Los datos han sido generados mediante simulaciones de Monte Carlo y contienen información sobre propiedades cinemáticas medidas por detectores de partículas en un acelerador, donde las primeras 21 características son de este tipo. Las últimas siete características son construidas a partir de las primeras 21 para ayudar a distinguir entre dos clases (detección de un bosón de Higgs o alguna otra partícula).

La primera columna es la etiqueta de clase (1 para señal, 0 para fondo), seguida por las 28 características (21 características de bajo nivel y luego 7 características de alto nivel): *pT_del_leptón*, *eta_del_leptón*, *phi_del_leptón*, *magnitud_de_energía_faltante*, *phi_de_energía_faltante*, *pt_del_jet_1*, *eta_del_jet_1*, *phi_del_jet_1*, *etiqueta_b_del_jet_1*, *pt_del_jet_2*, *eta_del_jet_2*, *phi_del_jet_2*, *etiqueta_b_del_jet_2*, *pt_del_jet_3*, *eta_del_jet_3*, *phi_del_jet_3*, *etiqueta_b_del_jet_3*, *pt_del_jet_4*, *eta_del_jet_4*, *phi_del_jet_4*, *etiqueta_b_del_jet_4*, $m_{jj}$, $m_{jjj}$, $m_{lv}$, $m_{jlv}$, $m_{bb}$, $m_{wbb}$, $m_{wwbb}$.

## Creación del Diseño del Programa

El diseño de las clases se muestra en el diagrama 1

### *BosonHiggs.java*

Clase principal que contiene el método `main` y la interfaz para que el usuario defina los parámetros de entrada.

### *CSVHandler.java*

Clase encargada de hacer operaciones con los CSV y sus directorios. Crear/borrar directorios, contar número de líneas, dividir el archivo en *n* partes.

### *Manager.java*

Clase encargada de crear el pool de workers, asignarles un CSV y esperar a que terminen para hacer `join` y unir los

resultados en un solo archivo.

### *Worker.java*

Clase que extiende de `Runnable` para los Workers concurrentes. Cada worker lee el CSV asignado, filtra los registros y los escribe en una variable `data` compartida entre los workers. Al momento de entrar a escribir en `data`, cada worker aplica un candado para evitar que otro worker escriba al mismo tiempo.

### *datosFiltrados.java*

Clase que contiene la variable `data` donde se escriben los registros filtrados por cada worker. Esta variable es `static` para que sea compartida entre los workers y el worker.

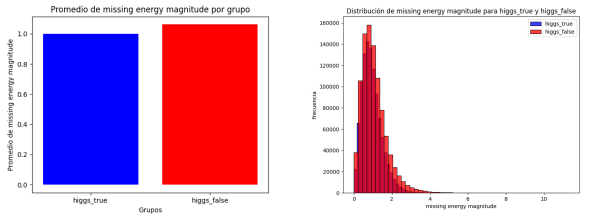
## ANÁLISIS DE DATOS

Para el análisis de datos, elegimos utilizar Python, un lenguaje de programación diferente a Java. Nos sentimos más cómodos trabajando con Python debido a nuestra familiaridad con él y a las excelentes herramientas que ofrece para este tipo de análisis.

Nuestro análisis se enfoca en obtener distribuciones, promedios y otros aspectos de las variables para comprender el grado de cambio en un proceso que involucra un bosón de Higgs en comparación con aquellos en los que no está presente.

Como era de esperar, el análisis de este tipo de datos, simulados debido a la naturaleza del experimento, es complicado. Sin embargo, mediante un procedimiento básico para obtener promedios y distribuciones, obtuvimos un par de resultados interesantes.

En las imágenes siguientes, mostramos las gráficas relacionadas con el promedio y las distribuciones de una de las variables más significativas en nuestro conjunto de datos, denominada "*missing_energy_magnitude*", la cual cuantifica la energía perdida en cada uno de los experimentos.



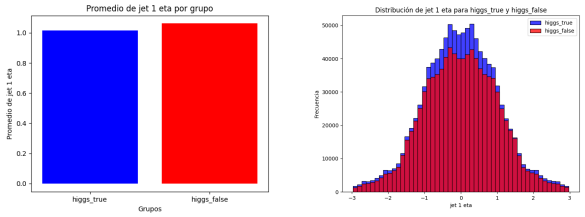

**(a)** Promedios **(b)** Distribución

**Fig. 2:** Energía perdida

En las imagenes anteriores mostramos las gráficas relacionadas al promedio y las distribuciones de una las variables más significativas en nuestro dataset que es "*missing_energy_magnitud*"la cual cuantifica la energía perdida en cada uno de los experimentos.

Los promedios indican que en procesos donde está presente un bosón de Higgs, se pierde menos energía que en aquellos donde el bosón no es el de Higgs. Esto se debe a que en procesos con un bosón de Higgs se producen algunas partículas que pueden ser registradas por los detectores, mientras que en procesos sin un bosón de Higgs no sucede así. La imagen de la distribución confirma lo mencionado, aunque los datos están distribuidos de manera similar en ambos casos, las frecuencias difieren, mostrando un comportamiento consistente en casi todas las variables.

A continuación, presentamos un análisis similar para la variable "*jet_1_eta*", que representa el momento $\eta$ de uno de los jets generados en una colisión.




**(a)** Promedios **(b)** Distribución

**Fig. 3:** Energía perdida

De manera similar a la variable "*missing_energy_magnitude*", el promedio muestra que se tienen valores más altos del momento del jet en procesos donde no se genera un bosón de Higgs. Sin embargo, en la distribución observamos que para procesos con un bosón de Higgs, hay un pico más prominente. En este caso, la distribución está centrada en cero, lo que indica que en procesos con un bosón de Higgs, el momento del jet tiende a ser muy bajo en comparación con los procesos donde no está presente.

## RESULTADOS

A continuación se muestran los resultados midiendo el tiempo de filtrado entre correr el procesamiento de manera secuencial o concurrente. El tiempo mostrado se captura desde que el/los workers toman su CSV correspondiente y lo filtran según los criterios. No se contempla el tiempo que toma abrir el archivo por primera vez para poderlo dividir en multiples CSV ya que esto se realiza de manera secuencial.

A lo largo de las 20 ejecuciones del programa que se realizaron (10 en concurrente y 10 en serial) podemos ver que el promedio del tiempo de ejecución es de 2.6 por lo cual podemos generalizar que el tiempo de ejecucion se reduce en menos de la mitad cuando usamos programación concurrente. Es importante mencionar que la ejecución del programa se realizó en

## CONCLUSIÓN

Basados en los resultados podemos ver que la mejora en el tiempo de ejecución es notable. El procesamiento de los 2 millones de renglones se recorta a la mitad al momento de utilizar la programación concurrente. En el desarrollo de este proyecto pudimos ver que en casos como este el hacer el código de manera concurrente no agrega demasiada complejidad al código pero si brinda una mejora en desempeño notable, por lo cual si se sabe que es un programa crítico que estará ejecutando archivos grandes o múltiples veces al día, desarrollarlo en concurrente deberia ser una de las primeras consideraciones a tener.

Algo que pudimos notar durante la comparación entre ejecuciones es que cuando corre en concurrente el orden del archivo filtrado varía un poco, por lo cual si se requiere que los datos mantengan el mismo orden se tendría que hacer alguna

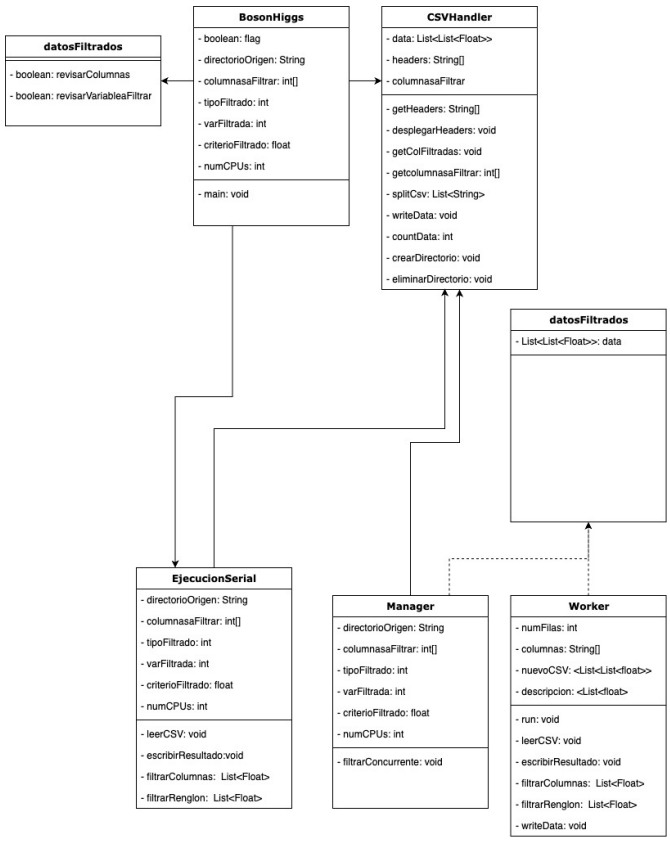



**Fig. 1:** Diagrama de clases de secciones responsables para el filtrado

modificación para asegurarse de mantener el orden.

## ANEXOS

**TABLA 1:** 20 EJECUCIONES DE FILTRADO, 10 EN SECUENCIAL Y 10 CONCURRENTE.

| Type | Original Size | Filtered Size | Filter Criterion | Execution Time (s) | Speedup |
|---|---|---|---|---|---|
| Concurrent | (25, 2,000,000) | (10: 324,342) | Col[1] <= 0.5 | 4.6 | 2.2 |
| Serial | (25, 2,000,000) | (10: 324,342) | Col[1] <= 0.5 | 10.2 | |
| Concurrent | (25, 2,000,000) | (20: 1,058,818) | Col[0] == 1 | 4.3 | 2.4 |
| Serial | (25, 2,000,000) | (20: 1,058,818) | Col[0] == 1 | 10.5 | |
| Concurrent | (25, 2,000,000) | (12: 18,941) | Col[10] <= 0.3 | 2.9 | 3.4 |
| Serial | (25, 2,000,000) | (12: 18,941) | Col[10] <= 0.3 | 9.8 | |
| Concurrent | (25, 2,000,000) | (6: 425,850) | Col[20] >=1 | 3.4 | 2.9 |
| Serial | (25, 2,000,000) | (6: 425,850) | Col[20] >=1 | 10 | |
| Concurrent | (25, 2,000,000) | (16: 924,453) | Col[11]>0.1 | 5.1 | 2.6 |
| Serial | (25, 2,000,000) | (16: 924,453) | Col[11]>0.1 | 13.4 | |
| Concurrent | (25, 2,000,000) | (21: 941,182) | Col[0] == 0 | 4.7 | 2.6 |
| Serial | (25, 2,000,000) | (21: 941,182) | Col[0] == 0 | 12 | |
| Concurrent | (25, 2,000,000) | (21: 2000,000) | No filter | 5.8 | 2.6 |
| Serial | (25, 2,000,000) | (21: 2000,000) | No filter | 15.1 | |
| Concurrent | (25, 2,000,000) | (17: 5,425) | Col[15] <0.8 | 5.4 | 2.4 |
| Serial | (25, 2,000,000) | (17: 5,425) | Col[15] <0.8 | 12.9 | |
| Concurrent | (25, 2,000,000) | (6: 1,213,483) | Col[2] <= 0.3 | 5 | 2.3 |
| Serial | (25, 2,000,000) | (6: 1,213,483) | Col[2] <= 0.3 | 11.3 | |
| Concurrent | (25, 2,000,000) | (24: 551,445) | Col[22] >1 | 4.2 | 2.7 |
| Serial | (25, 2,000,000) | (24: 551,445) | Col[22] >1 | 11.4 | |

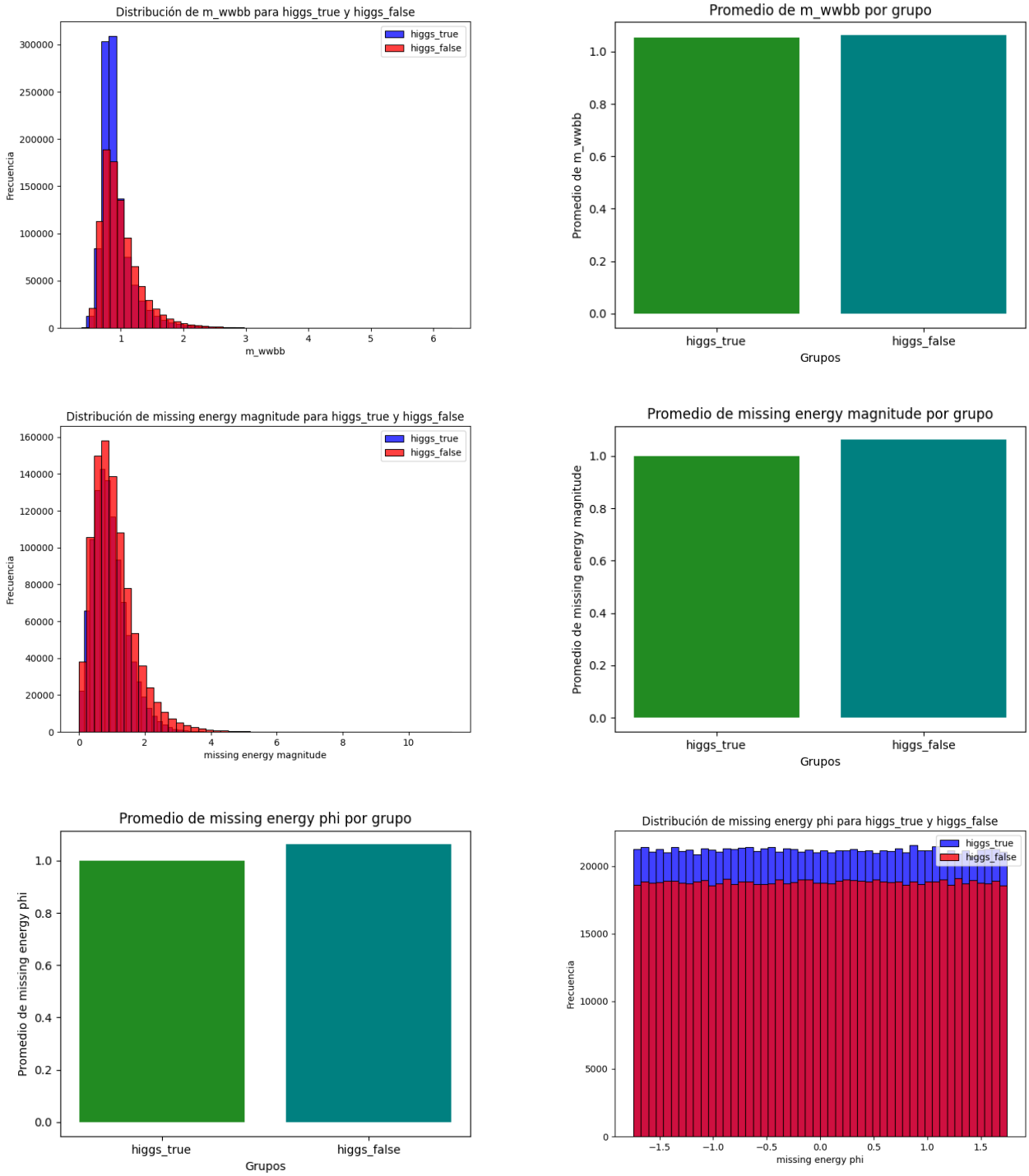



**Fig. 4:** Variables que muestran diferencia en las distribuciones/promedios. La gráfica b) muestra el comportamiento que mostraron otras variables, casi identicos.