Celda 1: Dividir archivo

using Pkg

Pkg.add("CSV")

Pkg.add("DataFrames")

using CSV

using DataFrames

# Cargar el archivo CSV

df = CSV.File("ratings.csv") |> DataFrame

# Definir el número de partes en las que quieres dividir el archivo

N = 10

# Calcular el tamaño de cada parte

num\_rows = nrow(df)

rows\_per\_part = div(num\_rows, N)

# Crear un loop para dividir el DataFrame y guardar cada parte

for i in 0:(N-1)

start\_row = i \* rows\_per\_part + 1

end\_row = (i == N - 1) ? num\_rows : (start\_row + rows\_per\_part - 1)

# Crear un DataFrame para la parte actual

part\_df = df[start\_row:end\_row, :]

# Guardar la parte como un nuevo archivo CSV

CSV.write("ratings\_part\_$(i+1).csv", part\_df)

end

println("El archivo ha sido dividido en $N partes.")

Celda 2

using CSV

using DataFrames

using Base.Threads

using Printf

# Asegúrate de que el número de hilos esté configurado

println("Número de hilos disponibles: ", nthreads())

# Función para cargar los archivos CSV y contar calificaciones por género

function cargar\_y\_contar(i, movies)

ratings = CSV.read("ratings\_part\_" \* string(i) \* ".csv", DataFrame)

return count\_ratings\_by\_genre(ratings, movies)

end

# Función para contar las calificaciones por género

function count\_ratings\_by\_genre(ratings, movies)

# Unir las tablas por 'movieId'

data = innerjoin(ratings, movies, on=:movieId)

# Inicializar un diccionario para contar las calificaciones por género

genre\_count = Dict{String, Int}()

for row in eachrow(data)

genres = split(row.genres, "|") # Dividir los géneros por '|'

for genre in genres

genre\_count[genre] = get(genre\_count, genre, 0) + 1

end

end

return genre\_count

end

# Cargar la lista de películas

movies = CSV.read("movies.csv", DataFrame)

# Arreglo para almacenar los resultados de cada hilo

results = Vector{Dict{String, Int}}(undef, 10)

# Usar Threads.@threads para ejecutar en paralelo

Threads.@threads for i in 1:10

results[i] = cargar\_y\_contar(i, movies)

end

# Combinar los resultados de todos los hilos

final\_result = Dict{String, Int}()

for result in results

for (genre, count) in result

final\_result[genre] = get(final\_result, genre, 0) + count

end

end

function format\_with\_commas(n::Int)

s = string(n)

len = length(s)

# Si la longitud es menor o igual a 3, solo retorna el número

if len <= 3

return s

end

# Crear una cadena con comas

result = ""

for (i, digit) in enumerate(reverse(s))

if i > 1 && (i - 1) % 3 == 0

result \*= ","

end

result \*= digit

end

return reverse(result)

end

# Iterar sobre el diccionario e imprimir el resultado formateado

for (genre, count) in final\_result

formated\_count = format\_with\_commas(count)

println("$genre: \t $formated\_count calificaciones")

end