Hiện tại tao đã chép file /root/data\_sample/citizens\_data\_small.csv tới cho các server chạy worker rồi. Nhưng sao vẫn chậm

**ChatGPT said:**

ChatGPT

Nếu bạn đã sao chép file CSV đến tất cả các server chạy worker nhưng hiệu suất vẫn chậm, có thể có các vấn đề khác liên quan đến cấu hình hoặc cách Spark xử lý dữ liệu. Dưới đây là các lý do và cách kiểm tra:

**1. Phân phối và đọc dữ liệu (File locality)**

**Lý do:**

* Mặc dù file CSV đã được sao chép, Spark có thể không tối ưu hóa việc đọc file dựa trên vị trí của dữ liệu.
* Trong Spark, các worker không tự động ưu tiên đọc file từ local disk của chính nó. Thay vào đó, nó có thể vẫn giao tiếp qua mạng với các worker khác, gây overhead.

**Cách khắc phục:**

1. **Xác định đường dẫn file trên mỗi worker**:
   * Đảm bảo file\_path trong code Spark sử dụng đường dẫn **local** trên từng worker.

python

Copy code

file\_path = "file:///root/data\_sample/citizens\_data\_small.csv"

1. **Sử dụng Broadcast**:
   * Nếu file nhỏ, bạn có thể sử dụng Spark **Broadcast** để gửi dữ liệu tới tất cả các worker:

python

Copy code

from pyspark.sql import SparkSession

from pyspark.sql.functions import col

from pyspark.broadcast import Broadcast

spark = SparkSession.builder \

.appName("Optimized Read") \

.master("spark://172.30.2.98:7077") \

.getOrCreate()

# Đọc file vào driver trước

with open('/root/data\_sample/citizens\_data\_small.csv', 'r') as f:

data = f.read()

# Broadcast dữ liệu tới các worker

broadcast\_data = spark.sparkContext.broadcast(data)

# Chuyển broadcast thành RDD hoặc DataFrame

rdd = spark.sparkContext.parallelize(broadcast\_data.value.splitlines())

df = spark.read.csv(rdd)

**2. Kích thước partition không phù hợp**

**Lý do:**

* Spark tự động chia file CSV thành các partition. Nếu file nhỏ, việc có quá nhiều partition sẽ gây thêm overhead:
  + Nhiều partition → Nhiều task nhỏ → Thời gian khởi tạo task và giao tiếp tăng.

**Cách khắc phục:**

1. **Giảm số lượng partition**:
   * Tối ưu số lượng partition sao cho gần với số core khả dụng:

python

Copy code

df = spark.read.csv(file\_path, header=True, inferSchema=True).repartition(2)

1. **Kiểm tra số partition**:
   * Xem số lượng partition mặc định mà Spark tạo:

python

Copy code

print(df.rdd.getNumPartitions())

**3. Shuffle dữ liệu không cần thiết**

**Lý do:**

* Nếu bạn sử dụng các thao tác như filter, Spark có thể thực hiện **shuffle** dữ liệu (di chuyển dữ liệu giữa các worker).

**Cách khắc phục:**

1. **Giữ dữ liệu cục bộ**:
   * Thực hiện tối đa các phép tính trong mỗi partition trước khi giảm dữ liệu toàn cục.
2. **Kích hoạt Broadcast Join (nếu cần)**:
   * Nếu bạn đang thực hiện join hoặc filter phức tạp, bạn có thể bật **Broadcast Join**:

python

Copy code

spark.conf.set("spark.sql.autoBroadcastJoinThreshold", -1)

**4. Hiệu suất đọc/ghi của hệ thống tệp**

**Lý do:**

* Nếu các worker sử dụng hệ thống tệp chậm hoặc bị tranh chấp (NFS hoặc HDD thay vì SSD), hiệu suất đọc/ghi dữ liệu sẽ giảm đáng kể.

**Cách khắc phục:**

1. **Đảm bảo file CSV trên SSD**:
   * Kiểm tra tốc độ truy cập local disk trên mỗi worker:

bash

Copy code

dd if=/root/data\_sample/citizens\_data\_small.csv of=/dev/null bs=1M count=100

1. **Cấu hình Spark để ưu tiên locality**:
   * Bật tùy chọn cho phép worker ưu tiên dữ liệu local:

bash

Copy code

spark.locality.wait=0

**5. Phân bổ tài nguyên không tối ưu**

**Lý do:**

* Nếu mỗi worker được cấu hình với ít core hoặc RAM, Spark không thể tận dụng hết khả năng xử lý song song.

**Cách khắc phục:**

1. **Kiểm tra tài nguyên của worker**:
   * Đảm bảo mỗi worker có đủ core và RAM:

bash

Copy code

--executor-cores 2 --executor-memory 4G

1. **Giảm số lượng worker nếu cần**:
   * Với file nhỏ, việc giảm số lượng worker (hoặc sử dụng tất cả tài nguyên trên một worker) có thể cải thiện hiệu suất.

**6. Kiểm tra trong Spark UI**

* Truy cập Spark Web UI (http://<master-host>:8080) để kiểm tra:
  + **Thời gian thực thi mỗi task**.
  + **Chi phí Shuffle** (read/write).
  + **Số lượng partition và dữ liệu mỗi partition**.

**Tóm tắt**

Dù file đã được sao chép tới worker, thời gian xử lý chậm có thể do:

1. Chi phí đồng bộ (file locality không được tối ưu).
2. Partition quá nhiều hoặc không phù hợp.
3. Shuffle dữ liệu không cần thiết.
4. Hệ thống tệp hoặc I/O bị giới hạn.
5. Phân bổ tài nguyên Spark chưa tối ưu.

**Hành động tiếp theo**

1. Xác minh đường dẫn file là local (file:///...).
2. Kiểm tra số partition và điều chỉnh nếu cần.
3. Sử dụng Broadcast nếu file nhỏ.
4. Xem xét việc giảm số lượng worker hoặc tăng tài nguyên cho mỗi worker.

Nếu vẫn chưa rõ, có thể gửi thêm log hoặc cấu hình cụm Spark để phân tích chi tiết hơn! 😊