Caching and persist.

Why Everything Is Showing 0.0 B?

This typically means:
No caching was used or enabled for the data being read.
Without caching no memory storage

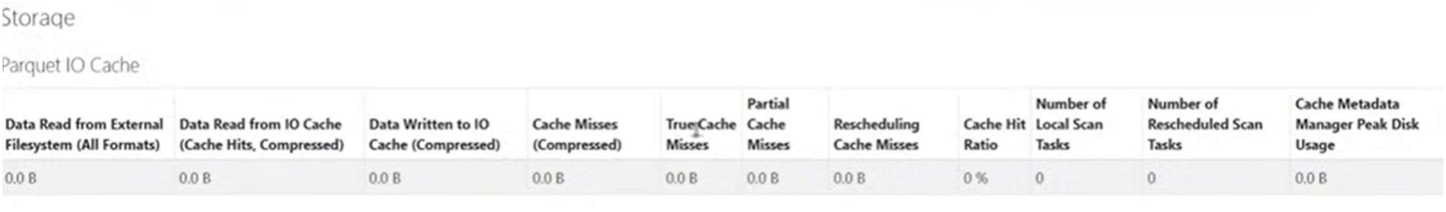
Spark Memory Usage: With vs Without Cache

1. Without .cache()

Each action (e.g., .count(), .show()) **recomputes the entire lineage**.
Data is **not stored in memory** between actions.
Memory usage stays low, but CPU and IO cost is high due to repeated computation.
Useful when the DataFrame is used **only once**.

2. With .cache() or .persist()

Data is **stored in memory** (default: deserialized form).
First action triggers execution and **loads data into memory**.
Subsequent actions **reuse the in-memory data**, avoiding recomputation.
Increases **memory usage** (visible in Spark UI under "Storage" tab).
Useful when the same DataFrame is used **multiple times**.



Metric	Without Cache	With Cache
Memory Used (Storage)	0 KiB	e.g., 406.6 KiB
Cache Hit Ratio	0%	100%
RDD/DF Recomputation	Happens every time	Only first time
Performance (Multiple Actions)	Slower	Faster

Spark Commands

```python
# Without cache
df = spark.read.csv("/path/to/file.csv", header=True, inferSchema=True)
df.count()  # full computation
df.select("column").show()  # recomputes again

# With cache
df = spark.read.csv("/path/to/file.csv", header=True, inferSchema=True)
df.cache()
df.count()  # triggers caching
df.select("column").show()  # uses cached memory
```

Caching and StorageLevel in PySpark – Short Notes

Import Required

from pyspark import StorageLevel

Caching Methods

.cache() → Shortcut for .persist(StorageLevel.MEMORY_AND_DISK)

.persist(StorageLevel.X) → Gives control over storage behavior

Common Storage Levels

StorageLevel	Description
MEMORY_ONLY	Stores only in memory, fails if not enough RAM
MEMORY_AND_DISK	Stores in memory, spills to disk if needed
DISK_ONLY	Caches only on disk, skips memory
MEMORY_ONLY_SER	Serialized storage in memory, saves space, uses more CPU
MEMORY_AND_DISK_SER	Serialized in memory, spills to disk if memory full

Example Usage

df = spark.read.csv("/path/to/file.csv", header=True, inferSchema=True)

df.persist(StorageLevel.MEMORY_AND_DISK) # Can also use other levels

df.count() # Triggers caching

View Cached Info

Go to **Spark UI → Storage tab**

Shows:

Storage Level (e.g., Disk Memory Deserialized)

Size in Memory

Cached Partitions

Fraction Cached