Pyspark - Partition

18 December 2025 13:51

PARTITION → logical division of data, help spark to process data parallel.

file read.

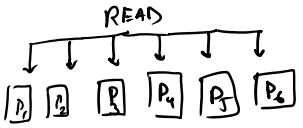
Spark / Pyspark → distributed computing.

→ Read

→ file operation → operation

Partition ← No. of input file, file size

① Default partition (READ) → csv / json / parquet → block size = 128 MB



② Default partition (operation) → Dataframe / Spark sql → [200 shuffle partition]

- groupBy
- join
- distinct
- orderBy
- window function

Ex- [1000 files — each size 64 MB]

How many partition

file 1 — 64MB
file 2 — 64MB
file 3 — 64MB
file 4 — 64MB
⋮
file 1000 — 64MB

≈ 128 MB

64MB < 128 MB

Spark – by default, one partition per file

Ex-2. file size = 198 MB — 1000 file.

2 partition ← file 1. — 198 MB
file 2 — 198 MB
⋮
file 1000 — 198 MB

198 MB > 128 MB

128 MB partition 1 | 70 MB partition 2

1000 file × 2 partition (each file)

= 2000 partition

2. Default partition (operation) → Dataframe / Spark sql → [200 shuffle partition]

groupBy ✓
Join
distinct
orderBy
window function

[only one partitioning scheme is active per stage]

Read → 1000 file → df → GroupBy → 200 Shuffle partition → Result

64MB file

1000 partition

Eg df = 5TB / 1GB

↳ default = 200 shuffle partition.

5TB ÷ 200 = 25 GB per partition

↳ very large shuffle partition.
→ long running task.
→ poor parallelism

No. of partition depend upon data size

of 128MB

Solution — partition size decrease

(+ increase - decrease) repartition → 200

(- decrease) Coalesce. Use.

Ex - 100GB [GROUP BY] / 200 (shuffle partition)

repartition = 0.5GB ≈ 512 MB → computation fast (parallel).

100GB / 800 ≈ 128MB

finance, Health, IT, SPACE

100 GB → groupBy → DEPT

4

P1 P2 P3 P4 P5 ..... P800

result.write.parquet(path) → file 1, file 2, ..., file 800

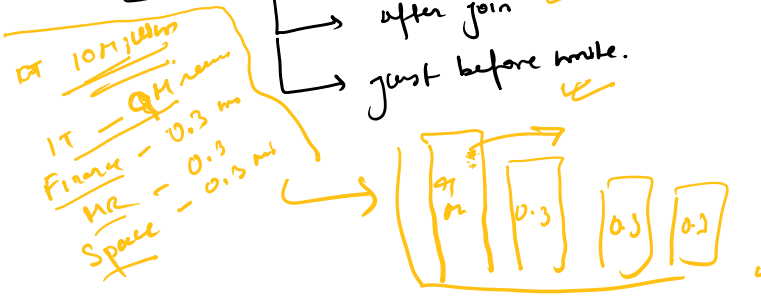
spark.conf.set("spark.sql.shuffle.partition", 800)
result = df.groupBy("dep").count()

result.coalesce(4).write.parquet(path) → 4 files

→ result.coalesce(4).write.parquet(path)

800 partition —merge→ 4 output partition file
4 output partition file → one file per department

COALESCE
- after groupBy
- after join
- just before write.



100GB (df)
↓
groupBy (800 shuffle partition) → Computation optimized
↓
Coalesce(4) → storage optimized
↓
4 clean output file