RDD Persistence

Tushar B. Kute,
http://tusharkute.com

RDD Persistence

- Spark RDD persistence is an optimization technique in which saves the result of RDD evaluation.
- Using this we save the intermediate result so that we can use it further if required.
- It reduces the computation overhead.
- We can make persisted RDD through cache() and persist() methods. When we use the cache() method we can store all the RDD in-memory.
- We can persist the RDD in memory and use it efficiently across parallel operations.

RDD Persistence

- The difference between cache() and persist() is that using cache() the default storage level is MEMORY_ONLY while using persist() we can use various storage levels.
- It is a key tool for an interactive algorithm. Because, when we persist RDD each node stores any partition of it that it computes in memory and makes it reusable for future use.
- This process speeds up the further computation ten times.

RDD Persistence

- When the RDD is computed for the first time, it is kept in memory on the node.
- The cache memory of the Spark is fault tolerant so whenever any partition of RDD is lost, it can be recovered by transformation Operation that originally created it.

RDD Persistence : Why?

- In Spark, we can use some RDD's multiple times. If honestly, we repeat the same process of RDD evaluation each time it required or brought into action.
- This task can be time and memory consuming, especially for iterative algorithms that look at data multiple times.
- To solve the problem of repeated computation the technique of persistence came into the picture.

RDD Persistence : Benefits

- There are some advantages of RDD caching and persistence mechanism in spark.
- It makes the whole system
  - Time efficient
  - Cost efficient
  - Lessen the execution time.

RDD Persistence : Storage Levels

- Using persist() we can use various storage levels to Store Persisted RDDs in Apache Spark. Let's discuss each RDD storage level one by one-
- MEMORY_ONLY
  - In this storage level, RDD is stored as deserialized Java object in the JVM. If the size of RDD is greater than memory, It will not cache some partition and recompute them next time whenever needed.
  - In this level the space used for storage is very high, the CPU computation time is low, the data is stored in-memory. It does not make use of the disk.

RDD Persistence : Storage Levels

- MEMORY_AND_DISK
  - In this level, RDD is stored as deserialized Java object in the JVM.
  - When the size of RDD is greater than the size of memory, it stores the excess partition on the disk, and retrieve from disk whenever required.
  - In this level the space used for storage is high, the CPU computation time is medium, it makes use of both in-memory and on disk storage.

RDD Persistence : Storage Levels

- MEMORY_ONLY_SER
  - This level of Spark store the RDD as serialized Java object (one-byte array per partition). It is more space efficient as compared to deserialized objects, especially when it uses fast serializer.
  - But it increases the overhead on CPU. In this level the storage space is low, the CPU computation time is high and the data is stored in-memory.
  - It does not make use of the disk.

RDD Persistence : Storage Levels

- MEMORY_AND_DISK_SER
  - It is similar to MEMORY_ONLY_SER, but it drops the partition that does not fits into memory to disk, rather than recomputing each time it is needed.
  - In this storage level, The space used for storage is low, the CPU computation time is high, it makes use of both in-memory and on disk storage.

RDD Persistence : Storage Levels

- DISK_ONLY
  - In this storage level, RDD is stored only on disk.
  - The space used for storage is low, the CPU computation time is high and it makes use of on disk storage.

RDD Unpersistence

- Spark monitor the cache of each node automatically and drop out the old data partition in the LRU (least recently used) fashion.
- LRU is an algorithm which ensures the least frequently used data.
- It spills out that data from the cache. We can also remove the cache manually using RDD.unpersist() method.

Conclusion

- Hence, Caching or persistence are the optimization techniques for interactive and iterative Spark computations.
- It helps to save intermediate results so we can reuse them in subsequent stages.
- These intermediate results as RDDs are thus kept in memory (default) or more solid storages like disk and/or replicated.

Thank you

This presentation is created using LibreOffice Impress 5.1.6.2, can be used freely as per GNU General Public License

@mitu_skillologies

/mITuSkillologies

@mitu_group

/company/mitu-skillologies

MITUSkillologies

Web Resources
https://mitu.co.in
http://tusharkute.com

contact@mitu.co.in

tushar@tusharkute.com