Paper Evaluation

Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing

Michael Lanthier

This paper introduces the idea of Resilient Distributed Datasets (RDDs) which are used to perform in memory distributed batch computations across large clusters of machines. RDDs implemented in their system, Spark, are able to provide fault tolerance by tracking the operations performed on the data allowing for quick re-computation in the case of failure or stragglers. These improvements not only remove the need for replication but also increase performance when compared to similar systems like Mapreduce.

Questions/Comments

1. If your transformations still cannot reduce the data size to fit in memory how does this compare to Mapreduce? Does it degrade to similar performance except you gain some better fault tolerance and recovery? Do you just end up running at disk speed depending on what operations are happening?
2. How does the overhead of checkpointing compare to not checkpointing at all and just letting recovery happen via recomputation? I assume checkpoints can be done asynchronously so maybe it is not super expensive but some overhead must be expected. If your failure rates are very low on nodes it may make sense to just avoid checkpoining to avoid the overhead.
3. Since fine grained reads are capable in all of the nodes, could a query language be thrown in front of this to allow for easier analysis of data after computation has finished?