Spark: Cluster Computing with Working Sets

Kartikeya Upasani (kuu2101)

Paper Review, 31 October 2016

1 Motivation

Certain applications such as machine learning or data analytics have computational needs that involve cyclic workflow of large scale data. However, existing cluster computing libraries like MapReduce are designed primarily for acyclic data flow.

2 Goal

Developing a scalable and fault-tolerant cluster computing API for distributed tasks operating on cyclic workflows of large-scale data.

3 Key Idea

Storing data in a 'lazy' fashion, in memory, instead of writing all data to persistent storage. Fast access of data in memory is traded for cost of having to recompute a chunk on node failure.

4 Approach

A resilient distributed dataset (RDD) is a read-only collection of objects partitioned across a set of machines that can be rebuilt if a partition is lost. Spark provides cache operation that hints if data is going to be reused and must be kept in memory, and alternatively a save operation if data is not going to be used in near future. Parallel operations like reduce, collect and foreach are provided to operate on RDDs. Spark also provides shared variables that prevent same data from being copied redundantly when being passed to functions. Counters are abstracted as special variables called Accumulators that support fast increment and set operations.

5 Results

The Spark implementation was tested on Logistic Regression, Alternating Least Squares, and querying a large dump of Wikipedia. It proved to be 10x, 2.5x and 30x faster than Hadoop for each experiment respectively.

6 Conclusion

Spark provides a way to express certain problems in simple and computationally efficient ways that leads to reduces execution times and easier coding.

7 Comments

The paper talks about Spark at an early stage of development. Spark today has several interesting optimizations such as lazy evaluation of queries. Moreover, the level of detail of approach is low.