Spark – Review

Yixing Chen, yc3094

The paper *Spark: Cluster Computing with Working Sets* introduced Spark – an open source cluster computing framework. MapReduce and its variants have been highly successful in implementing large-scale data-intensive applications on commodity clusters. However, most of these systems are built around an acyclic data flow model that is not suitable for other popular applications. We propose a new framework called Spark that supports these applications while retaining the scalability and fault tolerance of MapReduce. To achieve these goals, Spark introduces an abstraction called resilient distributed datasets (RDDs).

Compare with Hadoop's two-stage disk-based MapReduce paradigm, Spark's multi-stage in-memory primitives provides performance up to 100 times faster for certain applications. By allowing user programs to load data into a cluster's memory and query it repeatedly, Spark is well-suited to machine learning algorithms.

Spark requires a cluster manager and a distributed storage system. For cluster management, Spark supports standalone (native Spark cluster), Hadoop YARN, or Apache Mesos. For distributed storage, Spark can interface with a wide variety, including Hadoop Distributed File System (HDFS), Cassandra, OpenStack Swift, Amazon S3, Kudu, or a costume solution can be implemented. Spark also supports a pseudo-distributed local mode, usually used only for development or testing purpose, where distributed storage is not required and the local file system can be used instead. In such kind of scenario, Spark is run on a single machine with one executor per CPU core.

The results of experiments the researchers ran shows that Spark has great performance of conducting logistic regression jobs, alternating least squares and query interactively.

The last part is the future work, it mainly includes data abstraction for programming clusters using core idea behind RDDs. Spark provides three simple data abstractions for programming clusters: resilient distributed datasets (RDDs), and two restricted types of shared variables: broadcast variables and accumulators. While these abstractions are limited, so in the future the writers plan to focus on four parts: formally characterize the properties of RDDs, enhance the RDD abstraction, define new operation to transform RDDs and prove higher-level interactive interfaces.