Summary Section 2.4 Extensions to MapReduce

This section introduces extensions and modifications of Hadoop MapReduce system. The most popular systems include UC Berkeley’s Spark., Google’s TensorFlow, and a graph model of data, Google’s Pregel.

MapReduce paradigm consists of a simple two step structure, Map and Reduce. It can solve most massive data processing problems. However, MapReduce model has a few limitations when deal with complicated tasks. MapReduce save intermediate results on local file system of Map and Reduce workers. And in complicate cases, one output is often input to another MapReduce task. This will require repeated read from and write to disks. It will then require more job completions time for run through multiple steps and multiple jobs.

Spark, TensorFlow, Pregel and other MapReduce extensions are all use a “Workflow” architecture. They share three major characteristics with MapReduce.

1. Build on a distributed file system.
2. Manage tremendous tasks, whereas only need to write small number of functions.
3. Handle failures occur during execution without restart job all over.

Workflow systems improved MapReduce by using an acyclic graph to deal with any collection of functions. Workflow systems use a master controller for dividing the works among the tasks by hashing inputs. The output of function f will be passed as inputs of f’s successors g and i. Workflow systems use effectively cascades of MapReduce jobs, that can significantly reduce communication cost that read and save to local files between chaining job tasks.

Workflow inherits MapReduce’s blocking property by only deliver completed output. If a task fails, its master control can easily kill the failed task at that node and restart that task.

Diagram

Description automatically generated

## Spark

Spark improves many MapReduce drawbacks, while keeps many benefits.

Spark Implementation is different from MapReduce in many of aspects:

* Performance

Spark could drastically speed up large scale of big data tasks, because it utilises RAM to process data in memory, while MapReduce persists data back to the disk after each Map-Reduce task. This allows Spark save communication time between tasks.

Same as MapReduce, with workflow architecture, Spark breaks down large dataset and process them in parallel. However, Spark works well for smaller data sets that can all fit into a server's RAM.

* Ease of use

Spark has a faster learning curve than MapReduce. Spark provide pre-built APIs for Java, Scala, Python, and R, etc. It is easy to program user-defined functions for different developers. Whereas MapReduce is written in Java.

Spark includes a core data processing engine, as well as libraries for SQL, machine learning, and stream processing.

* Compatibility

Hadoop focus on process key-value pairs as inputs and outputs. Whereas Spark is more flexible. Spark is compatible with all of Hadoop’s data sources and file formats. In addition, Spark use a Resilient Distributed Dataset (RDD), that is distributed and fault-tolerant and not restricted only for key-value pairs in the MapReduce. Spark use transformation and action operations that apply one RDD to produce another RDD such as Map, Flatmap, and Filter operations.

* Data processing

Hadoop MapReduce is great for batch processing. Whereas Spark can do much more. Spark can do real-time processing due to its high performance. Spark is capable to process graphs and deal with machine learning tasks. Spark offers a "one size fits all" platform that you can use rather than splitting tasks across different platforms.

TensorFlow

TensorFlow is another workflow system and use a multidimensional matrix instead of RDD in Spark model. It supports machine-learning with an easy-to-use built-in operation.

Recursive Extensions to MapReduce

Another main stream of extensions to MapReduce adopt recursion approach. It recursively use MapReduce job for a unknown steps until the result of two consecutive iterations are close enough. A few classic uses of recursive algorithm include PageRank and gradient descent.

However, recursions approach has limitations for failure recovery. Three different approaches have been used: Iterated MapReduce, Spark Approach and Bulk Synchronous Systems.

Pregel Bulk Synchronous Systems

Google’s Pregel system is a graph-based, bulk-synchronous system that consider its data as a graph. Each node is viewed as a task, and generate outputs as the inputs for other graph nodes. Computations are grouping as supersteps, where all messages received by any nodes at previous supersteps are processed and then generate new messages to destination nodes. This grouping message will make communication great but very short.

Pregel failure management is designed as checkpoints at certain supersteps, so that will not restart failed tasks.