Hadoop Distributed File System: Introduction

This lesson gives a brief introduction to the Hadoop Distributed File System.

We'll cover the following

Goal#

Design a distributed system that can store huge files (terabyte and larger). The system should be scalable, reliable, and highly available.

What is Hadoop Distributed File System (HDFS)?#

HDFS is a distributed file system and was built to store unstructured data. It is designed to store huge files reliably and stream those files at high bandwidth to user applications.

HDFS is a variant and a simplified version of the Google File System (GFS). A lot of HDFS architectural decisions are inspired by GFS design. HDFS is

A lot of HDFS architectural decisions are inspired by GFS design. HDFS is built around the idea that the most efficient data processing pattern is a

write-once, read-many-times pattern.

Background#

Apache Hadoop (https://hadoop.apache.org/) is a software framework that provides a distributed file storage system and distributed computing for analyzing and transforming very large data sets using the MapReduce (https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html) programming model. HDFS is the default file storage system in Hadoop. It is designed to be a **distributed**, **scalable**, **fault-tolerant** file system that primarily caters to the needs of the **MapReduce** paradigm.

Both HDFS and GFS were built to store very large files and scale to store petabytes of storage. Both were built for handling batch processing on huge data sets and were designed for data-intensive applications and not for end-users. Like GFS, HDFS is also not POSIX-compliant and is not a mountable file system on its own. It is typically accessed via HDFS clients or by using application programming interface (API) calls from the Hadoop libraries.

Given the current HDFS design, the following types of applications are not a good fit for HDFS:

1. **Low-latency data access:**
   HDFS is optimized for high throughput (which may come at the expense of latency). Therefore, applications that need low-latency data access will not work well with HDFS.

2. **Lots of small files:**
   HDFS has a central server called NameNode, which holds all the filesystem metadata in memory. This limits the number of files in the filesystem by the amount of memory on the NameNode. Although storing millions of files is feasible, billions are beyond the capability of the current hardware.

of the current hardware.

3. **No concurrent writers and arbitrary file modifications:** Contrary to GFS, multiple writers cannot concurrently write to an HDFS file. Furthermore, writes are always made at the end of the file, in an append-only fashion; **there is no support for modifications at arbitrary offsets in a file**.

APIs#

HDFS does not provide standard POSIX-like APIs. Instead, it exposes user-level APIs. In HDFS, files are organized hierarchically in directories and identified by their pathnames. HDFS supports the usual file system operations, e.g., files and directories can be **created**, **deleted**, **renamed**, **moved**, and **symbolic links** can be created. All **read** and **write** operations are done in an append-only fashion.

← Back

Mock Interview: GFS

Next →

High-level Architecture

Mark as Completed

Report an Issue