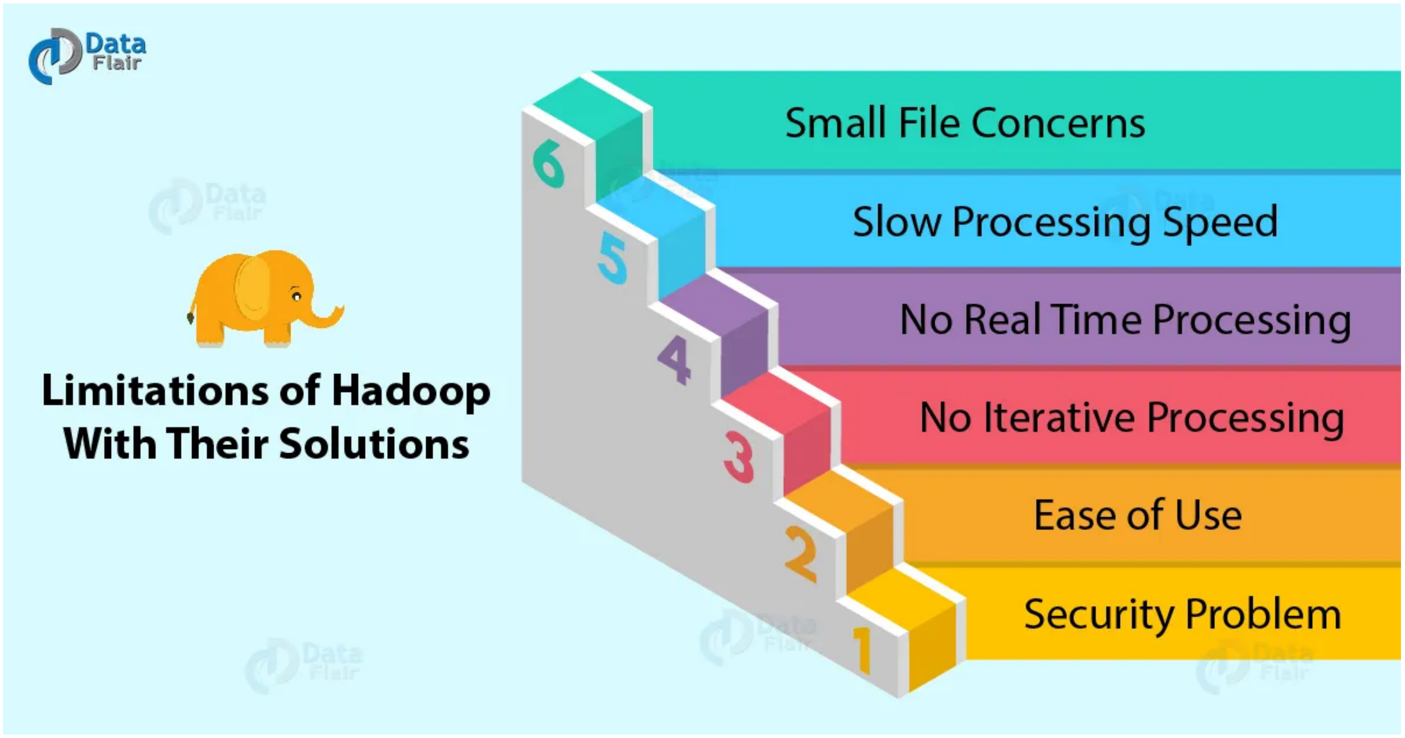
Bharvi Vyas Follow
Feb 24, 2019 · 6 min read · Listen

Save

6 Major Hadoop Limitations With Their Solutions


Data Flair
Limitations of Hadoop With Their Solutions
6 Small File Concerns
5 Slow Processing Speed
4 No Real Time Processing
3 No Iterative Processing
2 Ease of Use
1 Security Problem

Major Hadoop Limitations

Below are some of the Hadoop drawbacks with solutions, how can you overcome a problem. Read all the Hadoop limitations and its solutions, it will definitely help you for working smoothly in Hadoop.

- Ease of Use
- Security Problem

Let's discuss these Hadoop Limitations in detail -

1. Small File Concerns

The idea behind Hadoop was to have a small number of large files. But if you have many small files then Hadoop cannot manage it. Small files are those files whose size is quite less than the block size in Hadoop. Each file, directory, and block occupies a memory element inside NameNode's memory. As a **rule of thumb**, this memory element is about 150 bytes. So if you have 10 million files each using a block then it would occupy 1.39 GB of memory. Scaling beyond this level is not possible with current hardware. Also Retrieving small files is very inefficient in Hadoop. At the back-end, it causes many disks to seeks and hopping from one datanode to another. It incurs a lot of time.

Solution

Hadoop Archives or HAR files is one of the solutions to small files problem. Hadoop archives act as another layer of the file system over Hadoop. With Hadoop archive command we can build HAR files. This command runs a **map-reduce job** at the backend to pack the archived files into a small number of HDFS files. But again reading through HAR files is not much efficient than reading through HDFS. This is because it requires to access two index files and then finally the data file.

Sequence file is another solution to small file problem. In this, we write a program to merge a number of small files into one sequence file. Then we process this sequence file in a streaming fashion. Map-reduce can break this sequence files into chunks and process it in parallel as we can split the sequence file.

2. Slow Processing Speed

In Hadoop, the **MapReduce reads and writes the data** to and from the disk. For every stage in processing the data gets read from the disk and written to the disk. This disk seeks takes time thereby making the whole process very slow. If Hadoop processes data in small volume, it is very

Spark is the solution for the slow processing speed of map-reduce. It does in-memory calculations which makes it a hundred times faster than Hadoop. Spark while processing reads the data from RAM and writes the data to RAM thereby making it a fast processing tool.

Flink is one more technology which is faster than Hadoop map-reduce as it does in-memory calculations. **Flink is even faster than Spark**. This is due to the **stream processing engine** at the core as opposed to Spark which has batch processing engine.

3. No Real Time Processing

Hadoop with its **core Map-Reduce framework** is unable to process real-time data. Hadoop process data in batches. First, the user loads the file into HDFS. Then the user runs **map-reduce job** with the file as input. It follows the ETL cycle of processing. The user extracts the data from the source. Then the data gets transformed to meet the business requirements. And finally loaded into the data warehouse. The users can generate insights from this data. The companies use these insights for the betterment of their business.

Solution

Spark has come up as a solution to the above problem. Spark supports real-time processing. It processes the incoming streams of data by forming micro-batches and then applying computations on these micro-batches.

Flink is also one more solution for slow processing speed. It is even much faster than Spark as it has a stream processing engine at the core. Flink is a true streaming engine with adjustable latency and throughput. It has a rich set of APIs exploiting streaming runtime.

4. No Iterative Processing

Core Hadoop does not support iterative processing. Iterative processing requires a cyclic data flow. In this output of a previous stage serves as an input to the next stage. **Hadoop map-reduce** is capable of batch processing. It works on the principle of write-once-read-many. The data gets written on the disk once. And then read multiple times to get insights. The Map-reduce of Hadoop has a batch processing engine at its core. It is not able to iterate through data.

Solution

reusability of RDDs. The iterative algorithms apply operations repeatedly over data. Thus they benefit from RDDs caching across iterations.

Flink also supports iterative processing. Flink iterates data using streaming architecture. We can instruct Flink to process only the data which gets changed thereby improving the performance. Flink implements iterative algorithms by defining a step function. It embeds the step functions into special iteration operator. The two variants of this operator are — iterate and delta iterate. Both these operators apply the step function over and over again until they meet a terminating condition.

5. Ease of Use

In Hadoop, we have to hand code each and every operation. This has two drawbacks first it is difficult to use. And second, it increases the number of lines to code. There is no interactive mode available with Hadoop Map-Reduce. This also makes it difficult to debug as it runs in the batch mode. In this mode, we have to specify the jar file, the input as well as the location of the output file. If the program fails in between, it is difficult to find the culprit code.

Solution

Spark is easy for the user as compared to Hadoop. This is because it has many APIs for Java, Scala, Python, and **Spark SQL**. Spark performs batch processing, stream processing and machine learning on the same cluster. This makes life easy for users. They can use the same infrastructure for various workloads.

In Flink, the number of high-level operators is available. This reduces the number of lines of code to achieve the same result.

6. Security Problem

Hadoop does not implement encryption-decryption at the storage as well as network levels. Thus it is not much secure. For security, **Hadoop** adopts Kerberos authentication which is difficult to maintain.

Solution

Spark encrypts temporary data written to local disk. It does not support encryption of output data

Summary

The limitations of Hadoop and its MapReduce engine led to the invention of Spark and Flink. These both provide for stream and iterative processing apart from batch processing. They do this with lightning speed as they do in-memory calculations. Hence industry is now **switching to Big Data technologies** like Apache Spark and Apache Flink. But these technologies use Hadoop's HDFS for backend as it still provides a robust data storage system. Furthermore, if you have any query regarding Hadoop Limitations, ask in the comment section.

Big Data Hadoop Limitations Hadoop Solutions Disadvantages Of Hadoop Hadoop