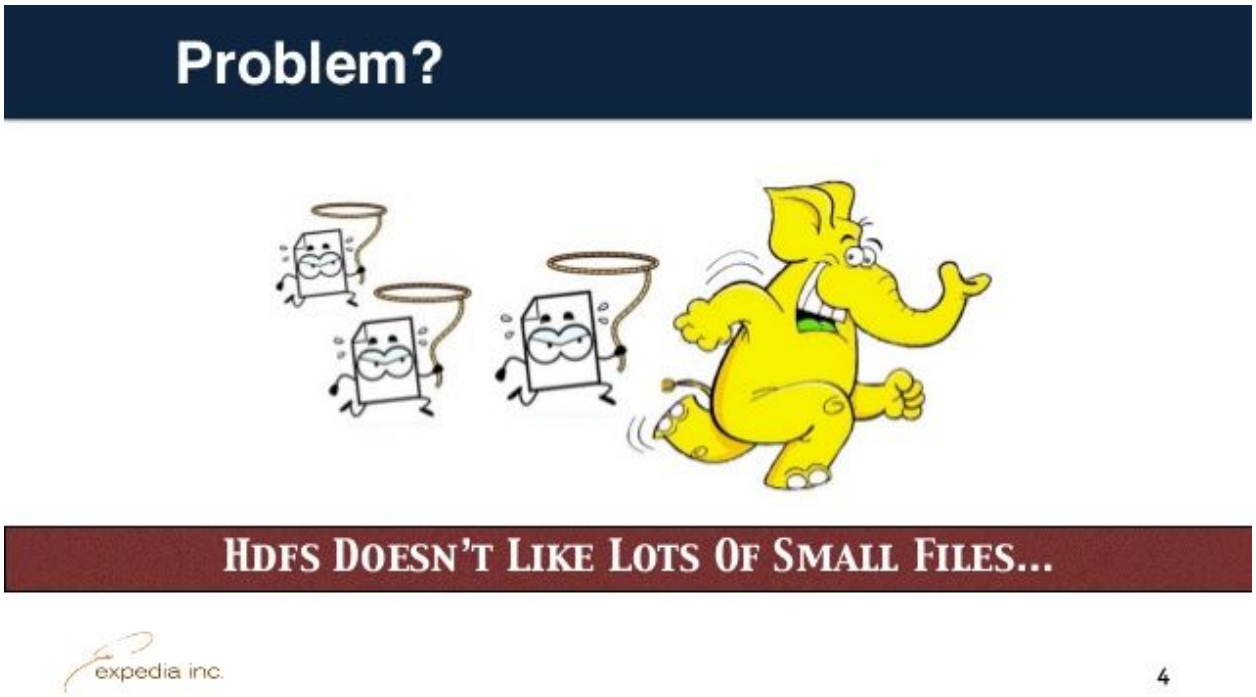


Problem Statement:

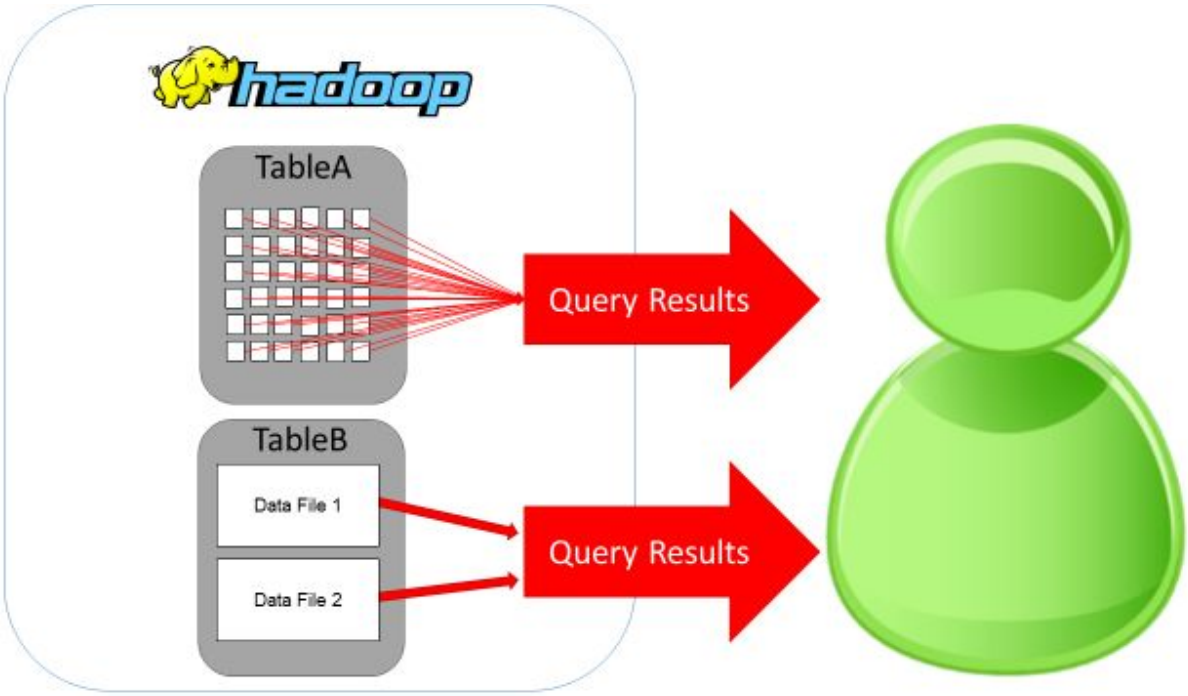
A problem as old as Hadoop itself, see this blog post from 2009:
https://blog.cloudera.com/blog/2009/02/the-small-files-problem/

So why is this an issue?

- Puts pressure on the NameNode i.e. constantly requiring more Heap Space
    - 1 Million files = 1 GB of Heap
    - As files and clusters grow the NameNode needs to grow with it
- Small files will throttle performance as many small files will require excessive disk I/O

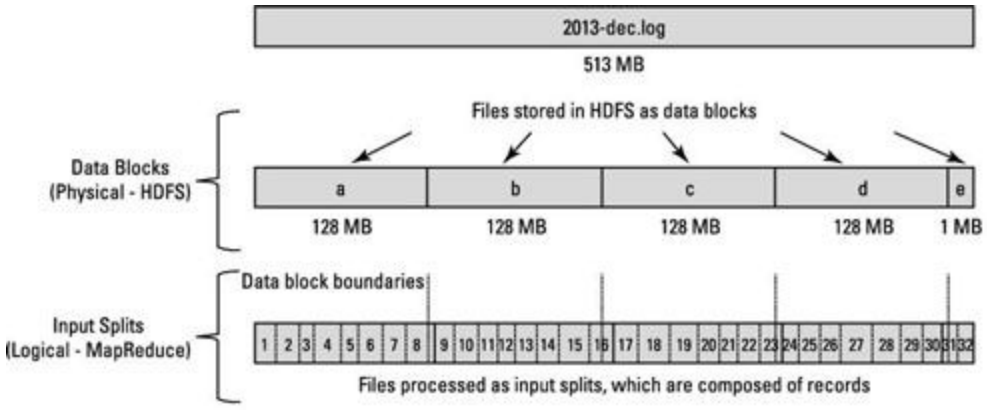
Disk I/O Performance Example:

Queries from the first table will require ~20x more time for Disk I/O



How HDFS Stores Files:

To understand the small file problem further it helps to also understand how files are stored across HDFS and how the metadata is tracked.



map: (K1, V1) → list(K2, V2)
reduce: (K2, list(V2)) → list(K3, V3)

Storage Examples:

This is such a common problem that Cloudera has a section dedicated to ensuring this issue is thought about and addressed in the proper manner.

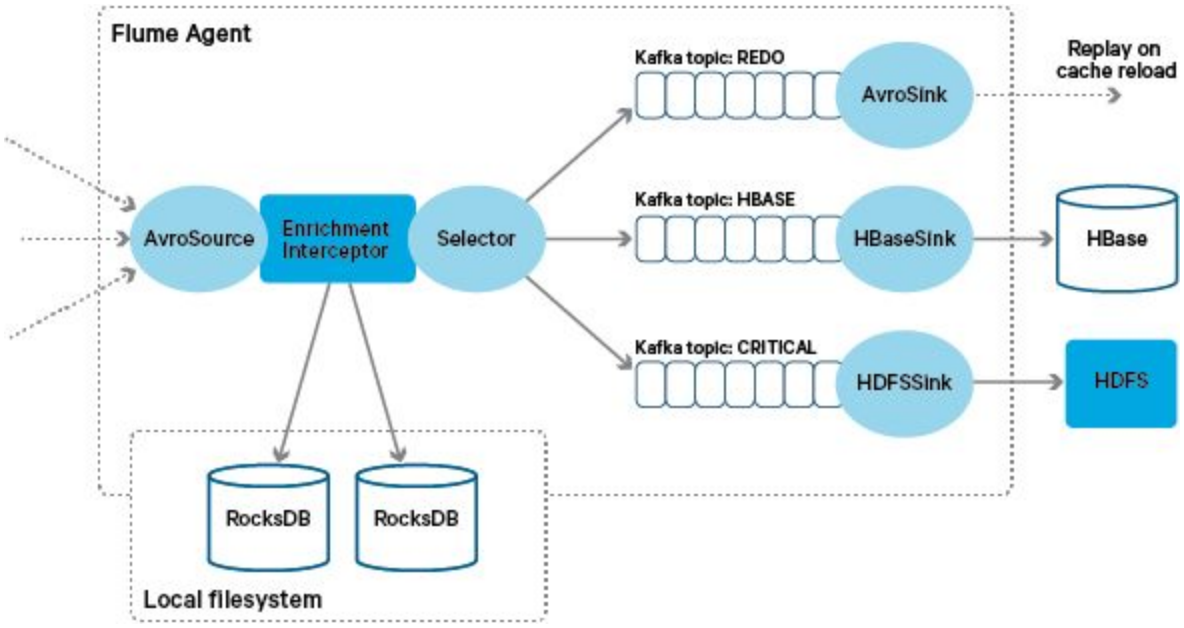
https://www.cloudera.com/documentation/enterprise/latest/topics/admin_nn_memory_config.html#concept_oyk_bdq_mv

How Small Files Typically Occur:

- Frequent stream of data (Flume)
- Over Partitioning
- Excessive Mappers or Reducers or Spark Executors
    - The last step in a Map-Reduce job determines the number of output files
    - The executor number in the last stage of Spark execution determines the final number of output files
- Combination of the above
- No Compression
    - Not a direct contributor to small files, but if a uncompressed file is 300 MB it consumes 3 HDFS block whereas compressing the file down to 128 MB will consume 1 HDFS block

Flume Example:

A typical Flume example may look like the following:



When Flume uses the HDFS Sink it will write to disk (close a file and open a new one) at specified flush times. Typically this is no longer than 10 seconds and usually is no bigger that a couple of MB's. Here is what it can typically look like:

Size	Last Modified	Replication	Block Size	Name
76.39 KB	2017-6-7 12:26:19	3	128 MB	FlumeData.1496834774363
75.07 KB	2017-6-7 12:26:24	3	128 MB	FlumeData.1496834774364
79.7 KB	2017-6-7 12:26:27	3	128 MB	FlumeData.1496834774365
74.67 KB	2017-6-7 12:26:31	3	128 MB	FlumeData.1496834774366
68.58 KB	2017-6-7 12:26:35	3	128 MB	FlumeData.1496834774367
69.9 KB	2017-6-7 12:26:39	3	128 MB	FlumeData.1496834774368
74.64 KB	2017-6-7 12:26:43	3	128 MB	FlumeData.1496834774369
79.11 KB	2017-6-7 12:26:46	3	128 MB	FlumeData.1496834774370
69.21 KB	2017-6-7 12:26:51	3	128 MB	FlumeData.1496834774371
20.29 KB	2017-6-7 12:26:57	3	128 MB	FlumeData.1496834774372

This can quickly put pressure on the namenode from a file and block count perspective since these files are being closed close to every 5 seconds. This simple example spans almost 40 seconds and has consumed 30 files and 30 blocks of storage. Over a day this extrapolates to 21,600 files and blocks and in a week it's 151,200 files and blocks. This needs to be handled properly to ensure everyone using the cluster is not affected, this is a multi-tenant environment and one bad actor can and will affect everyone from an HDFS perspective.

Potential Impact:

- Disk I/O Overhead
- NameNode Health (i.e. Cluster Health)
- Application (Spark, Impala, Hive, etc) performance slow down due to the following
    - Disk I/O
    - When querying wide tables, ensure serialization is being used

Compaction Options:

There are a few ways to adhere to this and depending on your requirements (always remember to pick the right tool for the job) you can, at a minimum, achieve efficient file consumption via scheduled cleanup jobs.

- Hive Compaction
    - Insert Overwrite Table on a schedule
- Hive Dynamic Compaction
    - Creates extra MR stage to clean up final file write

    - Settings: hive.merge.mapfiles, hive.merge.mapredfiles, hive.merge.size.per.task, hive.merge.smallfiles.avgsize
- Spark Compaction
- FileCrusher
- Hadoop Archive Record (HAR) - old

Future Considerations:

- HDFS 3 Erasure Encoding
- Kudu for streaming

Demo:

- Hive Compaction
- Spark Compaction

Whatever action you choose, try to keep the cluster happy and healthy!