LIT2018026

Question 1

Answer In Hadoop, When one of the machines i.e. part of pipeline which has datanode process running fails. Hadoop has an inbuilt functionality to handle this scenario. When a datanode fails while data is being written to it, the following actions are taken, which are transparent to client writing the data.

- first, the pipeline is closed, and any packets in ack queue are added to front of data queue so that datanode that are downstream from failed node will not miss any packets.
- The current block on good datanode is given a new identity, which is communicated to namenode, so that partial block on failed datanode will be deleted if the failed datanode recovery later on.
- The datanode that failed removed from pipeline, & then the remainder of block's data is written to two good datanodes in ~~picture~~ pipeline.

- The namenode notices that the block is under-replicated & it arranges for further replica to created on another node. Then it treats subsequent blocks as normal.

It is possible, but unlikely that multiple datanodes fail when client writes a block. As long as, it write dfs.replication.min replicas (which default to 1) the write will be successfull, & the block will asynchronously replicate across the cluster until it achives target replication factor. failover process is same as data write operation, data read operation is fault tolerance.

⇒ Apart from this Hadoop framework faces problem in reading & writing of small files because lack of ability to efficient support the random reading of small files because of high capacity design.

LIT2018026

Question → 2

Answer

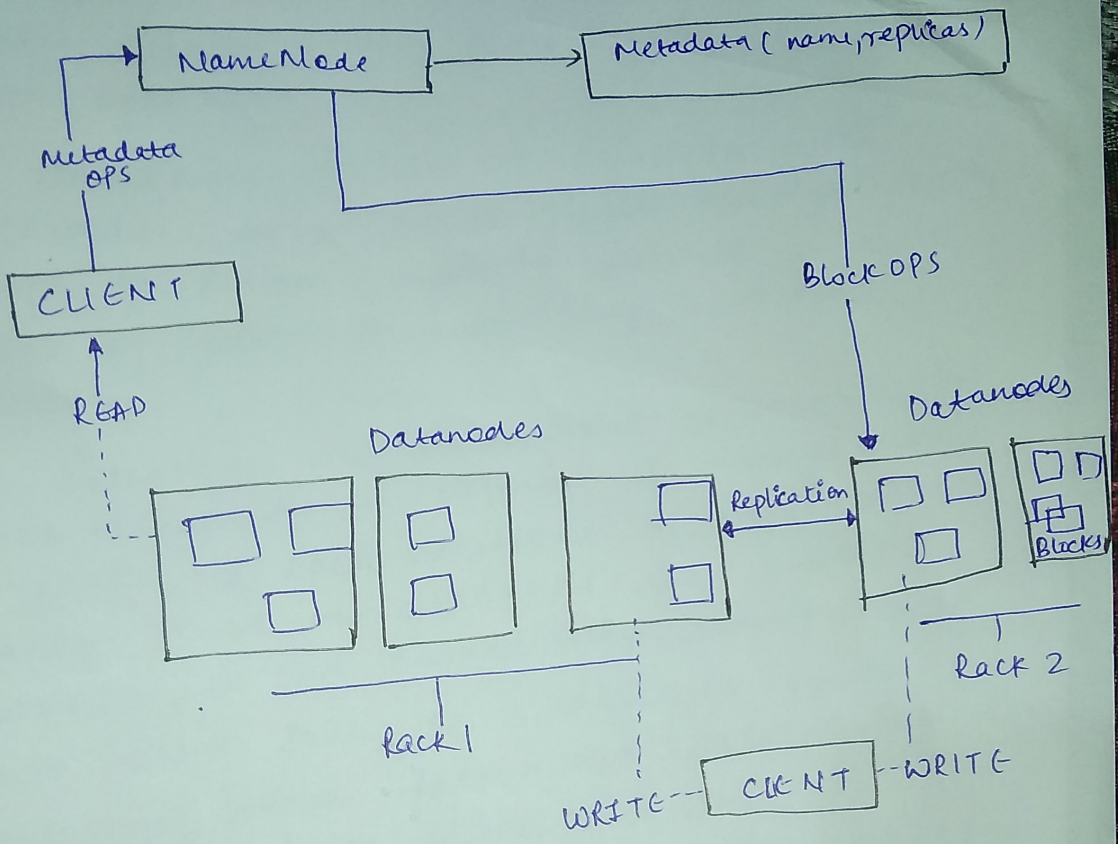
Working of HDFS :-

HDFS enables the rapid transfer of data between compute nodes. At its outset, it was closely coupled with MapReduce, a framework for data processing that filters & divides up work among node in a cluster, and it organizes & condenses the result into a cohesive answer to a query. Similarly, when HDFS takes in data, it breaks the information down into seperate blocks & distribute them to different nodes in cluster.

With HDFS, data is written in server once, read and reused numerous time after that. HDFS has a primary namenode, which keep track of where file data is kept in cluster.

HDFS has also multiple Datanode on a commodity hardware cluster - typically one per node in a cluster. The Datanodes are generally organized within the same rack in data center. Data is broken down into seperate blocks & distributed among various datanode, for storage. Blocks are also replicated across nodes, enabling highly efficient parallel processing.

The NameNode knows which DataNode



HDFS Architecture

contain which blocks & where the DataNodes reside within machine cluster. The Namenode also manages access to files including reads, writes, creates, deletes & data block replication across DataNodes.

The Namenode operates in conjuction with DataNodes. As a result cluster can dynamically adapt to server capacity demand in real time by adding or subtracting nodes as necessary.

The DataNodes are in constant communication with Namenode to determine if always aware of status of each DataNode. If Namenode realizes that one datanode isn't working properly, it can immediately reassign that Data node's task to a different node containing same data block. DataNodes also communicate with each other, which enables them to cooperate during normal file operation.

Advantages of HDFS: -

1) Handling oversized file ones: - Here usually refer to files with size of hundreds of MB & a setting of hundreds of TB. Currently in practical applications, HDFS can already be used to store & manage petabytes of data.

2) Streaming access to data: - The design of HDFS is based on responding more to write once, read, & write tasks. This means that once a data set is generated by datasource, it will be copied & distributed to different storage nodes.

3) Cost effectiveness: - Datanodes that stores data rely on inexpensive off-the-self hardware, which cuts storage costs. Also because HDFS is open source, there's no licesing fee

4) Fast recovery from hardware failure: - HDFS is designed to detect faults & automatically recover on its own.

5) Portability: - HDFS is portable across all compatible with several OS, including hardware platforms, & its window, Linux & Mac OS/X.

Disadvantages of HDFS :-

1) Not suitable for low-latency data access :-

If you want to process some low latency applic. requests that users require a relatively short time, HDFS is not suitable.

2) Unable to efficiently store a large number of small files :-

Because Namenode puts the metadata of file system in memory, the number of files that file system can accomodate is determined by size of Namenodes' memory.

3) Multi-user writing & arbitary file modification are not supported :-

There is only one writer in HDFS file ; the write operation can only be completed at the end of file, that is, only append operation can be performed. Currently HDFS doesn't support multiple users writing to same file & modifying the file anywhere.

LIT2018026

Question → 3 Primary NameNode:-

Answer:- The Namenode is centerpiece of HDFS file system. It keeps the directory tree of all files in the file system & track where across the cluster the file data is kept. It does not store the data of these file itself.

Client applications talk to the Name node when they wish to locate a file or when they want to add/copy/move/delete a file. The Namenode responds the successful requests by returning a list of relevant Datanode server where data lives.

The Namenode is single point of failure for HDFS cluster. HDFS is not currently a high availability system. When the Namenode goes down, the file system goes offline. There is an optional secondary Namenode that can be hosted on separate Machine.

⇒ Below are the main features performed by primary NameNode:-

1) Store metadata of actual data. E.g. filename, path, No. of data blocks, Block IDs, Block location, No. of replicas, Slave related configurati

2) Manages file system Namespace
3) Regulates client access request for actual file data file
4) Assign work to slaves
5) Executes files system namespace operation like opening/closing files, renaming files, & directories
6) As Namenode keep metadata in memory for fast retrieval, the huge amount of memory is required for its operation. This should be hosted on reliable hardware.

Question → 4 Secondary NameNode:- LIT2018026

Answer Secondary Namenode in Hadoop is more of a helper to Namenode. It is not backup namenode server which can quickly takeover in case of Namenode failure.

The secondary namenode job is to periodically merge the image file with edit log file.

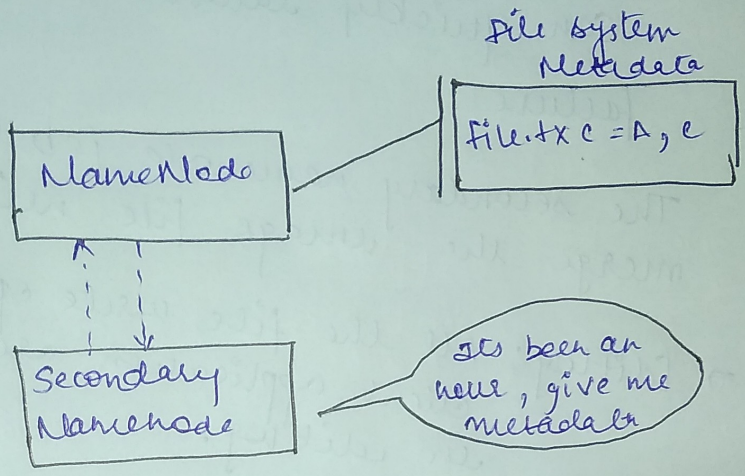
⇒ Editlog: All the file write operation done by client application are first recorded in edit log.

⇒ FSImage: This file has complete information about file system metadata when namenode starts. All operation after that are recorded in editlog.

This merge operation is to avoid the edit log file from growing into large file. The secondary namenode runs on separate machine & copies the image & edit log files periodically. The secondary namenode require as much as memory as primary namenode & bring the primary ~~the state~~ of ~~secondary~~ namenode up. ~~However~~ namenode crashes, then we can use the copied image & edit log files from secondary namenode & bring the primary namenode up. However the state of

LIT2018026

Secondary namenode lags from primary namenode. So incase of namenode failure, the data loss is obivous.



Secondary NameNode.

- Not a hot standby for Name Node
- Connects to Namenode ever hour*
- Housekeeping, backup of NameNode metadata
- Saved metadata can rebuild a failed name Node

LIT2018026

Question→5

Answer fencing Mechanism:-

With High Availability configured for HDFS, one NameNode server as active while other is serving as standby node. If active NameNode goes down, standby takes over the responsibility & configured proxy would direct request to new active Namenode. Now there would be only 1 Namenode which is in active state. If at later point of time, the first namenode join the cluster network again it wouldn't aware of the cluster state & would try to gain lock on metadata files. Also the proxy would not be able to direct the request. As only one Namenode (active) is allowed to change state of metadata, two concurrently active Namenode cause Split Brain Scenio. To prevent this HDFS provide fencing mechanism with following 2 options:-

1) sshfence:- The method provides a mechanism to ssh to active namenode & kill the process. for this it uses fuser to kill the process running on specified TCP port. It also require passwordless access to host machine. To enable this fencing mechanism, user need to set/added the following properties in hdfs-site.xml

```
<property>
<name> dfs.ha.fencing.methods </name>
<value> sshfence </value>
</property>

<property>
<name> dfs.ha.fensing.ssh.private-key-files </name>
<value> /home/exampluser/.ssh/id_esa </value>
</property>
```

2) shell:- This method provides a mechanism to run a custom script or shell command. It could be configured as:

```
<property>
<name> dfs.ha.fencing method </name>
<value> shell (/path/to/my/script.sh arg1 arg2:.)
</value>
</property>
```

This would run script.sh with specified args.

If the shell command returns an exit code of 0,
the fencing is determined to be successful.
If it returns any other exit code,
the fencing was unsuccessful.

LIT2018026

Question 6 :-

Answer ⇒ In this question, I am planning to design a music/radio website. This website will provides several services to its users, including free streams of songs by famous artists. It also recommend music & events to their userbase, offer them personalized charts & much more. I am taking assumption that million of people will use this website so every month it will generate large quantities of data that have to be processed.

⇒ Why this website will use Hadoop :-

As I assumed that due to million of people data &, It has to use a way to handle that much amount of data. So, that's why I am using Hadoop Architecture. This website is intends to store, manage & process the large data.

⇒ Hadoop at this website (Assumptions)

1) Approx 200 Nodes
2) 5 cores per node
3) 25 GB memory per node
4) Hive integration to run optimized SQL queries for analysis.

⇒ Things that I am planning to do with Hadoop

1) site stats & metrics
2) Charts
3) Reporting
4) Metadata corrections
5) Neighbours
6) Recommendations
7) Indexing for search
8) Evaluation
9) Data insights

* Charts for a single user can be shown in real time & are computed on fly. But computing overall charts is pretty big job & is done on Hadoop.
* overall visualizations also typically require Hadoop for getting the data they visualize

LIT2018026

⇒ Let us explore few advantages that's the why we should use Hadoop →

1) The Hadoop's component HDFS allowed keeping redundant backups for file (use streaming data, webblogs etc) saved on distribute filesystem in cost-effective way.

2) As Hadoop runs on commodity Hardware (affordable computer hardware), it will be easy for website to achieve scalability

3) As Hadoop is freely accersible, it doesn't mess with company's financial constraints. The Apache Hadoop community is highly active, & the code for it is open source, it will be ease for us to customize the framework according to the needs.

4) Hadoop supports a flexible framework for excuting allocated computing algorithms with relatively simple learning curve.

5) open source code & active community groups mean that the website can freely modify Hadoop to custom features & patches.

LIT2018026

→ Hive :-

The Apache Hive data warehouse software facilitates query & managing large datasets residing in distributed storage. Hive provides a mechanism to project structure onto this data & query the data using SQL-like language called HiveQL.

→ Sqoop:-

We will be used sqoop to transfer data between Hadoop & relational databases.

→ SQL on Hadoop:-

for running quick, ad-hoc query of all of that data sitting on huge cluster, we can write a new hadoop job which would take a bit of time.

By using all of these component, this website is able to & handle large data & can also with help of Hadoop & some other API's can make intelligent taste & compatible decision for generation recommendation.