Apache Hadoop HDFS Architecture

Recommended by 217 users

Ashish Bakshi (https://www.edureka.co/blog/author/ashishbedureka-co/) | Jul 17,2018

Apache Hadoop HDFS Architecture

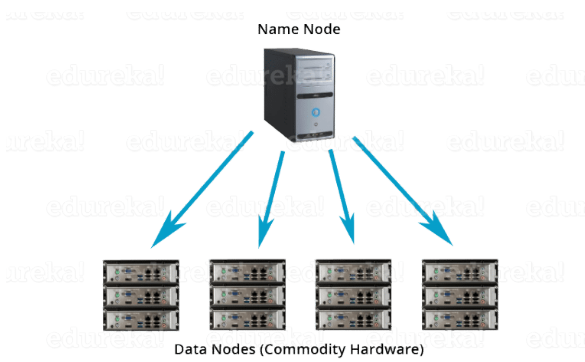
Introduction:

In this blog, I am going to talk about Apache Hadoop HDFS Architecture. From my ***previous blog (https://www.edureka.co/blog/hdfs-tutorial)***, you already know that HDFS is a distributed file system which is deployed on low cost commodity hardware. So, it's high time that we should take a deep dive into Apache Hadoop HDFS Architecture and unlock its beauty.

The topics that will be covered in this blog on Apache Hadoop HDFS Architecture are as following:

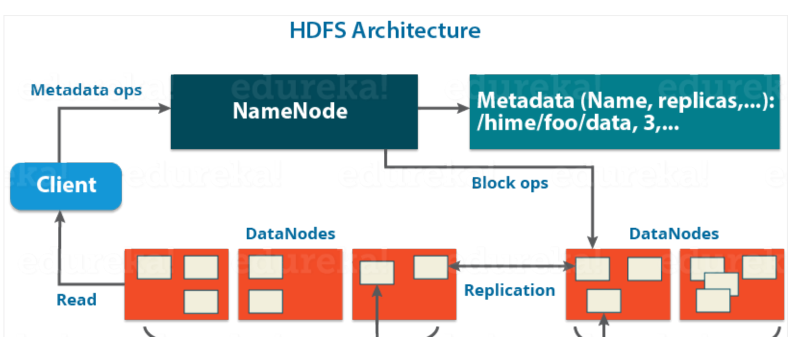
- **HDFS Master/Slave Topology**
- **NameNode, DataNode and Secondary NameNode**
- **What is a block?**
- **Replication Management**
- **Rack Awareness**
- **HDFS Read/Write – Behind the scenes**

HDFS Architecture:



Apache HDFS or **Hadoop Distributed File System** is a block-structured file system where each file is divided into blocks of a pre-determined size. These blocks are stored across a cluster of one or several machines. Apache Hadoop HDFS Architecture follows a *Master/Slave Architecture*, where a cluster comprises of a single NameNode (Master node) and all the other nodes are DataNodes (Slave nodes). HDFS can be deployed on a broad spectrum of machines that support Java. Though one can run several DataNodes on a single machine, but in the practical world, these DataNodes are spread across various machines.

NameNode:



NameNode is the master node in the Apache Hadoop HDFS Architecture that maintains and manages the blocks present on the DataNodes (slave nodes). NameNode is a very highly available server that manages the File System Namespace and controls access to files by clients. I will be discussing this High Availability feature of Apache Hadoop HDFS in my next blog. The HDFS architecture is built in such a way that the user data never resides on the NameNode. The data resides on DataNodes only.

Functions of NameNode:

- It is the master daemon that maintains and manages the DataNodes (slave nodes)
- It records the metadata of all the files stored in the cluster, e.g. The location of blocks stored, the size of the files, permissions, hierarchy, etc. There are two files associated with the metadata:
  - **FsImage:** It contains the complete state of the file system namespace since the start of the NameNode.
  - **EditLogs:** It contains all the recent modifications made to the file system with respect to the most recent FsImage.
- It records each change that takes place to the file system metadata. For example, if a file is deleted in HDFS, the NameNode will immediately record this in the EditLog.
- It regularly receives a Heartbeat and a block report from all the DataNodes in the cluster to ensure that the DataNodes are live.
- It keeps a record of all the blocks in HDFS and in which nodes these blocks are located.
- The NameNode is also responsible to take care of the **replication factor** of all the blocks which we will discuss in detail later in this HDFS tutorial blog.
- In **case of the DataNode failure**, the NameNode chooses new DataNodes for new replicas, balance disk usage and manages the communication traffic to the DataNodes.

DataNode:

DataNodes are the slave nodes in HDFS. Unlike NameNode, DataNode is a commodity hardware, that is, a non-expensive system which is not of high quality or high-availability. The DataNode is a block server that stores the data in the local file ext3 or ext4.

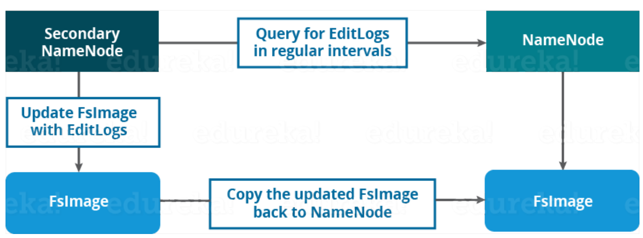
Functions of DataNode:

- These are slave daemons or process which runs on each slave machine.
- The actual data is stored on DataNodes.
- The DataNodes perform the low-level read and write requests from the file system's clients.
- They send heartbeats to the NameNode periodically to report the overall health of HDFS, by default, this frequency is set to 3 seconds.

Till now, you must have realized that the NameNode is pretty much important to us. If it fails, we are doomed. But don't worry, we will be talking about how Hadoop solved this single point of failure problem in the next Apache Hadoop HDFS Architecture blog. So, just relax for now and let's take one step at a time.

Secondary NameNode:

Apart from these two daemons, there is a third daemon or a process called Secondary NameNode. The Secondary NameNode works concurrently with the primary NameNode as a **helper daemon.** And don't be confused about the Secondary NameNode being a **backup NameNode because it is not.**



Functions of Secondary NameNode:

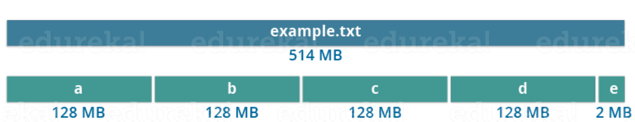
- The Secondary NameNode is one which constantly reads all the file systems and metadata from the RAM of the NameNode and writes it into the hard disk or the file system.
- It is responsible for combining the EditLogs with FsImage from the NameNode.
- It downloads the EditLogs from the NameNode at regular intervals and applies to FsImage. The new FsImage is copied back to the NameNode, which is used whenever the NameNode is started the next time.

Hence, Secondary NameNode performs regular checkpoints in HDFS. Therefore, it is also called CheckpointNode.

Blocks:

Now, as we know that the data in HDFS is scattered across the DataNodes as blocks. **Let's have a look at what is a block and how is it formed?**

Blocks are the nothing but the smallest continuous location on your hard drive where data is stored. In general, in any of the File System, you store the data as a collection of blocks. Similarly, HDFS stores each file as blocks which are scattered throughout the Apache Hadoop cluster. The default size of each block is 128 MB in Apache Hadoop 2.x (64 MB in Apache Hadoop 1.x) which you can configure as per your requirement.



It is not necessary that in HDFS, each file is stored in exact multiple of the configured block size (128 MB, 256 MB etc.). Let's take an example where I have a file "example.txt" of size 514 MB as shown in above figure. Suppose that we are using the default configuration of block size, which is 128 MB. Then, how many blocks will be created? 5, Right. The first four blocks will be of 128 MB. But, the last block will be of 2 MB size only.

Now, you must be thinking why we need to have such a huge blocks size i.e. 128 MB?

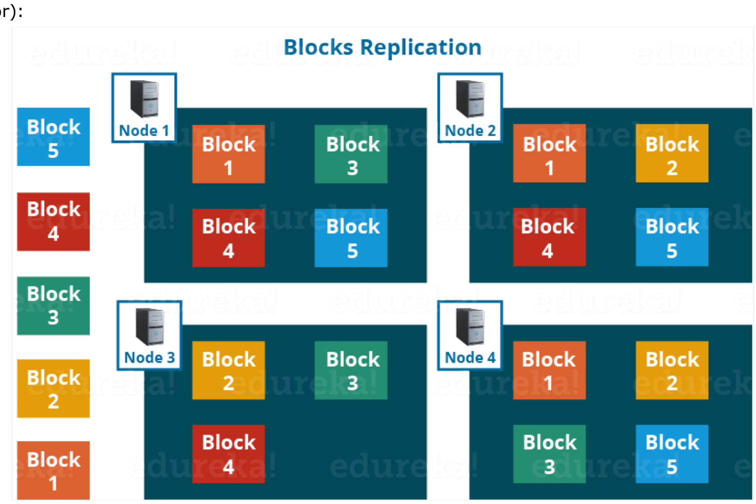
Well, whenever we talk about HDFS, we talk about huge data sets, i.e. Terabytes and Petabytes of data. So, if we had a block size of let's say of 4 KB, as in Linux file system, we would be having too many blocks and therefore too much of the metadata. So, managing these no. of blocks and metadata will create huge overhead, which is something, we don't want.

*As you understood **what a block is**, let us understand how the replication of these blocks takes place in the next section of this HDFS Architecture. Meanwhile, you may check out this video tutorial on HDFS Architecture where all the HDFS Architecture concepts has been discussed in detail:*

HDFS Architecture Tutorial Video | Edureka

Replication Management:

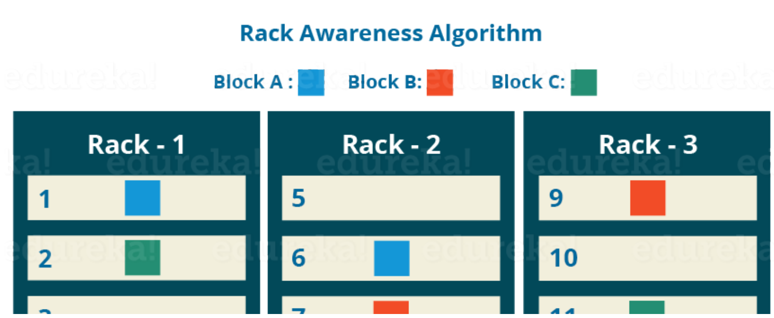
HDFS provides a reliable way to store huge data in a distributed environment as data blocks. The blocks are also replicated to provide fault tolerance. The default replication factor is 3 which is again configurable. So, as you can see in the figure below where each block is replicated three times and stored on different DataNodes (considering the default replication factor):



Therefore, if you are storing a file of 128 MB in HDFS using the default configuration, you will end up occupying a space of 384 MB (3*128 MB) as the blocks will be replicated three times and each replica will be residing on a different DataNode.

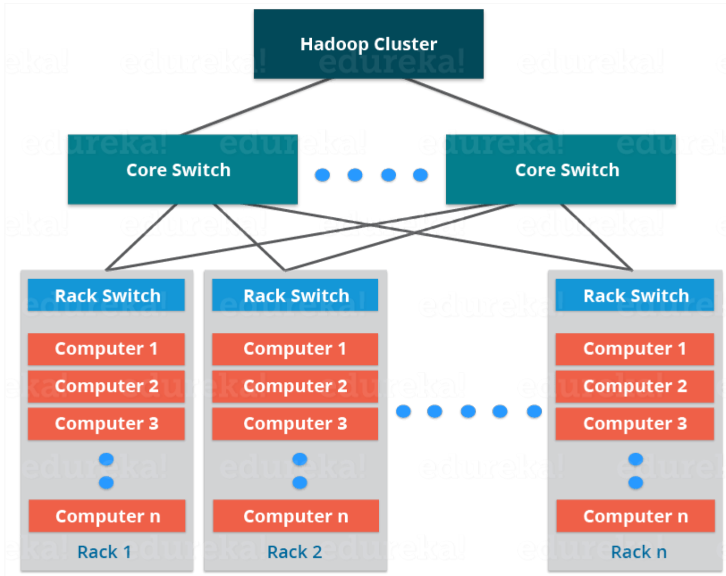
Note: The NameNode collects block report from DataNode periodically to maintain the replication factor. Therefore, whenever a block is over-replicated or under-replicated the NameNode deletes or add replicas as needed.

Rack Awareness:



Anyways, moving ahead, let's talk more about how HDFS places replica and what is rack awareness? Again, the NameNode also ensures that all the replicas are not stored on the same rack or a single rack. It follows an in-built Rack Awareness Algorithm to reduce latency as well as provide fault tolerance. Considering the replication factor is 3, the Rack Awareness Algorithm says that the first replica of a block will be stored on a local rack and the next two replicas will be stored on a different (remote) rack but, on a different DataNode within that (remote) rack as shown in the figure above. If you have more replicas, the rest of the replicas will be placed on random DataNodes provided not more than two replicas reside on the same rack, if possible.

This is how an actual Hadoop production cluster looks like. Here, you have multiple racks populated with DataNodes:



Advantages of Rack Awareness:

So, now you will be thinking why do we need a Rack Awareness algorithm? The reasons are:

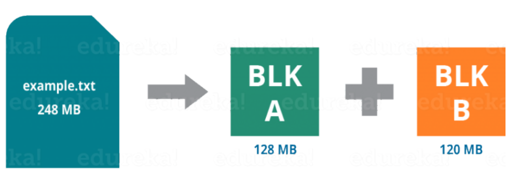
- **To improve the network performance:** The communication between nodes residing on different racks is directed via switch. In general, you will find *greater network bandwidth* between machines in the same rack than the machines residing in different rack. So, the Rack Awareness helps you to have reduce write traffic in between different racks and thus providing a better write performance. Also, you will be gaining increased read performance because you are using the bandwidth of multiple racks.
- **To prevent loss of data:** We don't have to worry about the data even if an entire rack fails because of the switch failure or power failure. And if you think about it, it will make sense, as it is said that *never put all your eggs in the same basket.*

HDFS Read/ Write Architecture:

Now let's talk about how the data read/write operations are performed on HDFS. HDFS follows Write Once – Read Many Philosophy. So, you can't edit files already stored in HDFS. But, you can append new data by re-opening the file.

HDFS Write Architecture:

Suppose a situation where an HDFS client, wants to write a file named "example.txt" of size 248 MB.



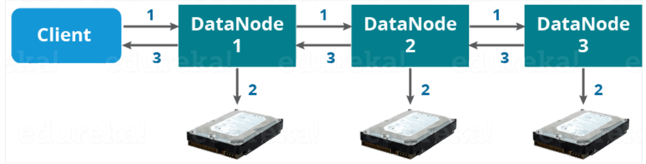
Assume that the system block size is configured for 128 MB (default). So, the client will be dividing the file "example.txt" into 2 blocks – one of 128 MB (Block A) and the other of 120 MB (block B).

Now, the following protocol will be followed whenever the data is written into HDFS:

- At first, the HDFS client will reach out to the NameNode for a Write Request against the two blocks, say, Block A & Block B.
- The NameNode will then grant the client the write permission and will provide the IP addresses of the DataNodes where the file blocks will be copied eventually.
- The selection of IP addresses of DataNodes is purely randomized based on availability, replication factor and rack awareness that we have discussed earlier.
- Let's say the replication factor is set to default i.e. 3. Therefore, for each block the NameNode will be providing the client a list of (3) IP addresses of DataNodes.

The list will be unique for each block.

- Suppose, the NameNode provided following lists of IP addresses to the client:
  - For Block A, list A = {IP of DataNode 1, IP of DataNode 4, IP of DataNode 6}
  - For Block B, set B = {IP of DataNode 3, IP of DataNode 7, IP of DataNode 9}
- Each block will be copied in three different DataNodes to maintain the replication factor consistent throughout the cluster.
- Now the whole data copy process will happen in three stages:

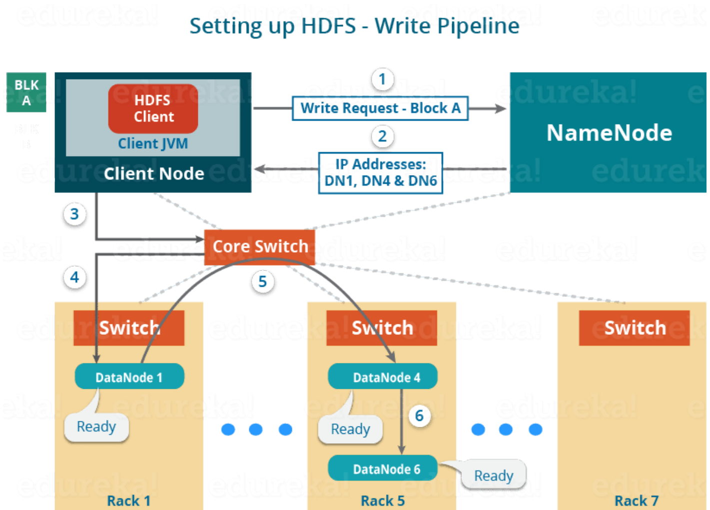


1. Set up of Pipeline
2. Data streaming and replication
3. Shutdown of Pipeline (Acknowledgement stage)

1. Set up of Pipeline:

Before writing the blocks, the client confirms whether the DataNodes, present in each of the list of IPs, are ready to receive the data or not. In doing so, the client creates a pipeline for each of the blocks by connecting the individual DataNodes in the respective list for that block. Let us consider Block A. The list of DataNodes provided by the NameNode is:

For Block A, list A = {IP of DataNode 1, IP of DataNode 4, IP of DataNode 6}.

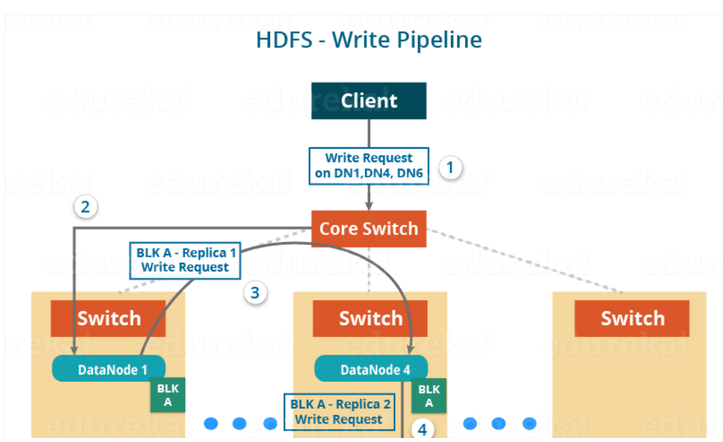


So, for block A, the client will be performing the following steps to create a pipeline:

- The client will choose the first DataNode in the list (DataNode IPs for Block A) which is DataNode 1 and will establish a TCP/IP connection.
- The client will inform DataNode 1 to be ready to receive the block. It will also provide the IPs of next two DataNodes (4 and 6) to the DataNode 1 where the block is supposed to be replicated.
- The DataNode 1 will connect to DataNode 4. The DataNode 1 will inform DataNode 4 to be ready to receive the block and will give it the IP of DataNode 6. Then, DataNode 4 will tell DataNode 6 to be ready for receiving the data.
- Next, the acknowledgement of readiness will follow the reverse sequence, i.e. From the DataNode 6 to 4 and then to 1.
- At last DataNode 1 will inform the client that all the DataNodes are ready and a pipeline will be formed between the client, DataNode 1, 4 and 6.
- Now pipeline set up is complete and the client will finally begin the data copy or streaming process.

2. Data Streaming:

As the pipeline has been created, the client will push the data into the pipeline. Now, don't forget that in HDFS, data is replicated based on replication factor. So, here Block A will be stored to three DataNodes as the assumed replication factor is 3. Moving ahead, the client will copy the block (A) to DataNode 1 only. The replication is always done by DataNodes sequentially.



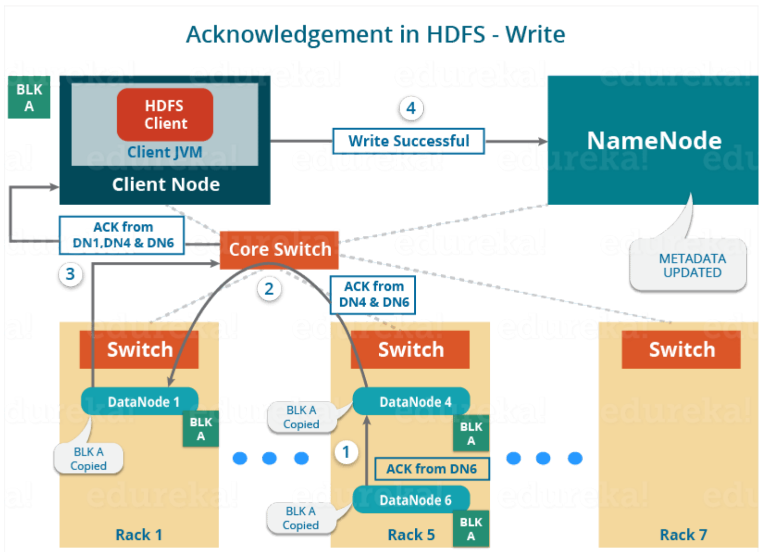
So, the following steps will take place during replication:

- Once the block has been written to DataNode 1 by the client, DataNode 1 will connect to DataNode 4.
- Then, DataNode 1 will push the block in the pipeline and data will be copied to DataNode 4.
- Again, DataNode 4 will connect to DataNode 6 and will copy the last replica of the block.

3. Shutdown of Pipeline or Acknowledgement stage:

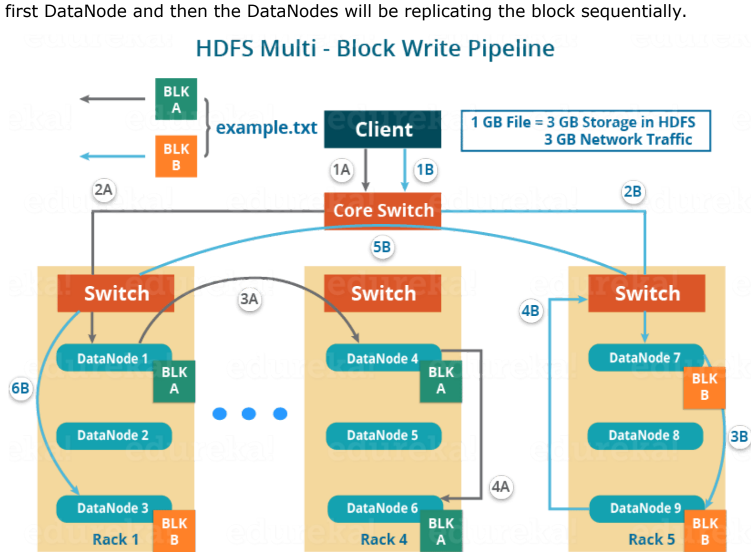
Once the block has been copied into all the three DataNodes, a series of acknowledgements will take place to ensure the client and NameNode that the data has been written successfully. Then, the client will finally close the pipeline to end the TCP session.

As shown in the figure below, the acknowledgement happens in the reverse sequence i.e. from DataNode 6 to 4 and then to 1. Finally, the DataNode 1 will push three acknowledgements (including its own) into the pipeline and send it to the client. The client will inform NameNode that data has been written successfully. The NameNode will update its metadata and the client will shut down the pipeline.



Similarly, Block B will also be copied into the DataNodes in parallel with Block A. So, the following things are to be noticed here:

- The client will copy Block A and Block B to the first DataNode **simultaneously**.
- Therefore, in our case, two pipelines will be formed for each of the block and all the process discussed above will happen in parallel in these two pipelines.
- The client writes the block into the first DataNode and then the DataNodes will be replicating the block sequentially.

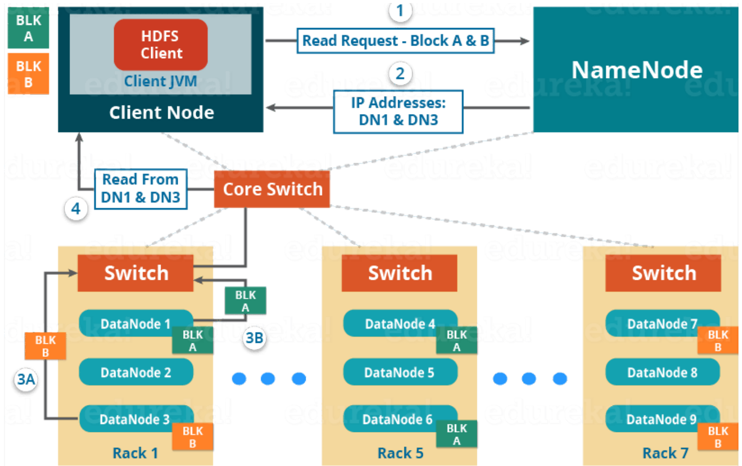


As you can see in the above image, there are two pipelines formed for each block (A and B). Following is the flow of operations that is taking place for each block in their respective pipelines:

- For Block A: 1A -> 2A -> 3A -> 4A
- For Block B: 1B -> 2B -> 3B -> 4B -> 5B -> 6B

HDFS Read Architecture:

HDFS Read architecture is comparatively easy to understand. Let’s take the above example again where the HDFS client wants to read the file “example.txt” now.



Now, following steps will be taking place while reading the file:

- The client will reach out to NameNode asking for the block metadata for the file "example.txt".
- The NameNode will return the list of DataNodes where each block (Block A and B) are stored.
- After that client, will connect to the DataNodes where the blocks are stored.
- The client starts reading data parallel from the DataNodes (Block A from DataNode 1 and Block B from DataNode 3).
- Once the client gets all the required file blocks, it will combine these blocks to form a file.

While serving read request of the client, HDFS selects the replica which is closest to the client. This reduces the read latency and the bandwidth consumption. Therefore, that replica is selected which resides on the same rack as the reader node, if possible.

Now, you should have a pretty good idea about Apache Hadoop HDFS Architecture. I understand that there is a lot of information here and it may not be easy to get it in one go. I would suggest you to go through it again and I am sure you will find it easier this time. Now, in my next blog, I will be talking about Apache Hadoop HDFS Federation and High Availability Architecture.

*Now that you have understood Hadoop architecture, check out the **Hadoop training (https://www.edureka.co/big-data-and-hadoop/)** by Edureka, a trusted online learning company with a network of more than 250,000 satisfied learners spread across the globe. The Edureka Big Data Hadoop Certification Training course helps learners become expert in HDFS, Yarn, MapReduce, Pig, Hive, HBase, Oozie, Flume and Sqoop using real-time use cases on Retail, Social Media, Aviation, Tourism, Finance domain.*

Got a question for us? Please mention it in the comments section and we will get back to you.

About Ashish Bakshi (11 Posts (https://www.edureka.co/blog/author/ashishbedureka-co/))

Share on

PREVIOUS

NEXT

Got your brain cells running?
Stay tuned to latest technology updates

Enter your Email Address

SUBSCRIBE

Related Posts

Introduction to Apache MapReduce and HDFS
28.7K
(https://www.edureka.co/blog/introduction-to-apache-hadoop-hdfs/)

10 Reasons Why Big Data Analytics is the Best Career Move
251.4K
(https://www.edureka.co/blog/10-reasons-why-big-data-analytics-is-the-best-career-move)

5 Reasons to Learn Hadoop
140.4K
(https://www.edureka.co/blog/5-reasons-to-learn-hadoop)

Comments

42 Comments

39 Comments https://www.edureka.co/blog/ Rajiv Chaudhuri

Recommend 3 Share Sort by Best

Join the discussion…

Shweta Vasagadekar • a year ago
Very precise explanation !! Thanks
2 • Reply • Share ›

EdurekaSupport Mod → Shweta Vasagadekar • a year ago
Hey Shweta, thanks for checking out our blog. We're glad you found it useful.
You might also like our YouTube tutorials; check them out here: https://www.youtube.com/edu....
Do subscribe to stay posted on upcoming blogs. Cheers!
• Reply • Share ›

Nam Nguyen • 2 years ago
Thank you so much.
2 • Reply • Share ›

eshank gupta • 9 months ago
In HDFS multi block write pipeline figure, can't we take 2B path same as 2A path, inorder to shorten the pipeline(no. of flow operations) for Block B ??
1 • Reply • Share ›

somu • 2 years ago
Excellent write-up ! Simple enough for a layman to understand and that is what we need.
1 • Reply • Share ›

EdurekaSupport Mod → somu • 2 years ago
Thanks for the wonderful feedback, Somu! Do check out some of our other HDFS blogs here: https://www.edureka.co/blog.... Cheers!
• Reply • Share ›

Hareesh@Disqus • 3 years ago
The following i have questions regarding HDFS and MR
1.Is it possible to store multiple files in HDFS with different block sizes?
2.Is it possible to give whole file as input to mapper?

Thanks
Hareesh A
1 • Reply • Share ›

EdurekaSupport Mod → Hareesh@Disqus • 3 years ago
Hi Hareesha,
Thank you for reaching out to us.
Yes it is possible in both situations but it will depend on the data blocks as well as the way in which they are applied. You can get in touch with us for further clarification by contacting our sales team on +91-8880862004 (India) or 1800 275 9730 (US toll free). You can also mail us on sales@edureka.co.
• Reply • Share ›

Kaushik Vankayala • 3 months ago

Hi Team,

I am stuck at the logstash installation step.

bin\logstash -e 'input { stdin { } } output { stdout {} }'

this is not working as expected. I am getting the error as follows;

Reply • Share ›

Anthony Brida • 3 months ago

Great explanation. Where can I download the code example?

Reply • Share ›

Sasi Diamond • 3 months ago

it was nice explanation.Thanks can i get the source code..??

Reply • Share ›

mohit gawanmde • 3 months ago

please send me code and dataset for predicting cat or dog

Reply • Share ›

Deepanshu Sharma • 3 months ago

It is absolutely well explained and precise but I wish to learn from the scrap then how would I be able to do it, Suggest me please. Thank You!

Reply • Share ›

Audrey • 5 months ago

I have an error: FATAL namenode.NameNode: Failed to start namenode.
Please help

Reply • Share ›

EdurekaSupport Mod → Audrey • 5 months ago

Hey Audrey, we'd like to help you out. Can you elaborate a little on the error? Thanks :)

Reply • Share ›

Sagar balai • 6 months ago

Excellent and very precise blog !!!

Reply • Share ›

EdurekaSupport Mod → Sagar balai • 6 months ago

Hey Sagar,
Thank you for appreciating our work. Do browse through our other blogs and let us know how you liked it.
Cheers :)

Reply • Share ›

Fadoua Mounir Chaoub • 8 months ago

a good and detailed explanation :) Thanks

Reply • Share ›

EdurekaSupport Mod → Fadoua Mounir Chaoub • 7 months ago

Hey Fadoua! Thank you for appreciating our work. Do check out our Hadoop certification course here:
https://www.edureka.co/big-...
Hope you find this useful as well :)

Reply • Share ›

SacTiw • 8 months ago

What happens if during write operation one of the datanode in the pipeline goes down?

I could imagine following possible scenarios:
- datanode goes down while waiting for data block
- datanode goes down while accepting the datablock
- datanode goes down while while waiting for acknowledgement (i.e. datablock has been copied and but further replication is still going on)

(by going down you can assume to be a machine crash or shutdown)

Reply • Share ›

Uma Mahesh yadav C • a year ago

Hi Ashish, Thanks for explaining very clearly.. Splitting file in to data bcks can be done by HDFS client? If so, mvFromLOcal, put commands also will spilt the file in to data blocks ? can you explain how this will happen please

• Reply • Share ›

EdurekaSupport Mod → Uma Mahesh yadav C • a year ago

Hey Uma Mahesh, thanks for checking out out blog. We're glad you found it useful.
The moment we execute the copyFromLocal command. The Hadoop environment will fetch the file from the provided path and split it into blocks .
Hope this helps. Cheers!

• Reply • Share ›

Rishav Kumar • 2 years ago

Very well explained, the sequence of explaining is too good.
I wanted to know if Hadoop uses any compression techniques to cope up with increased disk space requirement (default: 3 times) associated with data replication.

• Reply • Share ›

EdurekaSupport Mod → Rishav Kumar • 2 years ago

Hey Rishav, thanks for checking out the blog. First of all, the HDFS is deployed on low cost commodity hardware which is bound to fail. This is the most important reason why data replication is done i.e. to make the system Fault Tolerant and Reliable. And yes, Hadoop supports many codec utilities like gzip, bzip2, Snappy etc. But, there is always a tradeoff between compression ratio and compress/decompress speed. Also, the data are stored as blocks in HDFS, you can't apply those codec utilities where decompression of a block can't take place without having other blocks of the same file (residing on other DataNodes). In other words they need the whole file for decompression. These codecs are called non -splittable codecs. At last, HDFS cluster is scalable i.e. you can add more nodes to the cluster to increase the storage capacity
Hope this helps. Cheers!

• Reply • Share ›

Tanmay Jambavlikar • 2 years ago

If the NameNode fails what are the typical steps after addressing the relevant hardware problem to bring the name node online. I am asking this question from the fact that Fsimage must be the last up-to-date copy of the Meta-Data critical for hadoop cluster to operate and there is no automatic fail-over capability. So do we some how restore this copy on NameNode and then start the all the necessary daemons on the namenode? Will the cluster take this is Fsimage file as a valid input and then start its operations normally?

• Reply • Share ›

EdurekaSupport Mod → Tanmay Jambavlikar • 2 years ago

Hey Tanmay, thanks for checking out the blog. Let us understand this NameNode recovery process by taking an example where I am a Hadoop Admin and I have a situation where the NameNode has crashed in my HDFS cluster. So, the following steps will be taken by me to make the cluster up and running:
1. I will use the file system metadata replica (FsImage) to start a new NameNode.
2. Then, I will configure the DataNodes and clients so that they can acknowledge this new NameNode that I have started.
3. Now the new NameNode will start serving the client after it has completed loading the last checkpointed FsImage (for meta data information) and received enough block reports from the DataNodes to leave the safe mode.
This takes 30 minutes on an average. On large Hadoop clusters this NameNode recovery process may consume a lot of time and this becomes even a greater challenge in the case of the routine maintenance. This is why we have HDFS HA Architecture and HDFS Federation Architecture which is covered in a separate blog here: http://www.edureka.co/blog/.... Hope this helps. Cheers!

• Reply • Share ›

Bhupendra Pandey • 3 years ago

What is blockreport? Why datanodes need to send it to Namenode at regular interval? Doesn't namenode keep store metadata and block details in namespace at the time of file write?

• Reply • Share ›

Shaheer Kidwai → Bhupendra Pandey • 3 years ago

In some interval of time, the DataNode sends a block report to the
NameNode. The block report allows the NameNode to repair any divergence that may have occurred between the replica information on the NameNode and on the DataNodes. The Block and Replica Management may use this revised information to enqueue block replication or deletion commands for this or other DataNodes.

During normal operation DataNodes send heartbeats to the NameNode to confirm that the DataNode is operating and the block replicas it hosts are available. The default heartbeat interval is three seconds. If the NameNode does not receive a heartbeat from a DataNode in ten minutes the NameNode considers the DataNode to be out of service and the block replicas hosted by that DataNode to be unavailable. The NameNode then
schedules creation of new replicas of those blocks on other DataNodes.

Heartbeats from a DataNode also carry information about total storage capacity, fraction of storage in use, and the number of data transfers currently in progress. These statistics are used for the NameNode's block allocation and load balancing decisions.

• Reply • Share ›

Reply • Share ›

Ujwala ➦ Shaheer Kidwai • 3 years ago

if the default heartbeat interval is three seconds, isnt ten minutes too long to conclude that data node is out of service? is it the default number? can this be configured?

Reply • Share ›

EdurekaSupport Mod ➦ Ujwala • 3 years ago

Hi Ujwala, the default interval of time is 10 minutes and we can't change it. The Namenode wait for the heartbeat from the Datanode till the interval of time mentioned and if it doesn't receive the heartbeat then it consider that particular Datanode to be out of service and creates new replicas of those blocks on other Datanodes.

Reply • Share ›

EdurekaSupport Mod ➦ Shaheer Kidwai • 3 years ago

Thanks for responding to this question Shaheer.

Reply • Share ›

Deven Kalra • 4 years ago

When writing the data into physical blocks in the nodes, if one node fails, does it stop the writing process goes back to the name node, name node re writes nodes to write or I am wrong?

Reply • Share ›

EdurekaSupport Mod ➦ Deven Kalra • 4 years ago

Hi Deven, when writing the data into physical blocks in the nodes, namenode receives heart beat(a kind of signal) from the datanodes which indicates if the node is alive or not. If the namenode does not receive heartbeat within the specific time period then it assumes that the datanode has failed and then writes the data to a different data block.

Hope this helps!!

1 • Reply • Share ›

Biswa Bihari panda ➦ EdurekaSupport • 3 years ago

So it will write from the begining ?

Reply • Share ›

EdurekaSupport Mod ➦ Biswa Bihari panda • 3 years ago

Yes, you are right Biswa!!!

Reply • Share ›

rishikhurana17 • 4 years ago

very well explained .. thanx

Reply • Share ›

EdurekaSupport Mod ➦ rishikhurana17 • 4 years ago

Thanks Rishi!! Feel free to go through our other blog posts as well: http://www.edureka.co/blog/...

Reply • Share ›

Praveen Sharp • 5 years ago

I've read similar things on other blogs. Ill take your word for it. Stay solid! your pal.
Hmm, that is some compelling information youve got going! Makes me scratch my head and think. Keep up the good writing!

Reply • Share ›

Jojo De La Cruz • a year ago

Hi, I need help to further understand this statement from Rack Awareness vs the block B sequence in HDFS Multi - Block Write Pipeline diagram..

Rack Awareness:
"Considering the replication factor is 3, the Rack Awareness Algorithm says that the first replica of a block will be stored on a local rack and the next two replicas will be stored on a different (remote) rack but, on a different DataNode within that (remote) rack as shown in the figure above. If you have more replicas, the rest of the replicas will be placed on random DataNodes provided not more than two replicas reside on the same rack, if possible."

HDFS Multi - Block Write Pipeline:
The block B first replica was written in Node 7 in rack 7, then written in Node 9 still in rack 7 and then to Node 3 in rack 1

My understanding, the first replica will be in the local and that the subsequent write will be on another node in a different rack and then to another node to that within that same rack, but the "block B" shows otherwise. Can you help explain this further please?

Thanks you very much in advance.

Reply • Share ›

What is Robotic Process Automation? – An Introduction to RPA

2 comments • 5 months ago

Avatar **Lucas Ramalho Salata** — Hello, you. Very cool!Could you please correct the text below "COMPLIANCE"? It's wrong, I guess. It's the same text as …

7 Reasons to Choose Edureka Online Courses

4 comments • 6 months ago

Avatar **Gokhan Dedeoglu** — My registered e-mail adress is gdedeoglu@gmail.com

DevOps Roles: Which Of Them Is Your Dream?

2 comments • 7 months ago

Avatar **EdurekaSupport** — Hey Sharan, we are glad that you liked our blog. Do check out our DevOps course as well. We provide live online instructor led …

Top 55 Blockchain Interview Questions You Must Prepare In 2018

4 comments • 5 months ago

Avatar **Maximilian Fischer** — A blockchain (database) only contains a record of all changes that were ever made to it. Value is ascribed to the fact of actually being …

Related Blogs


(https://www.edureka.co/blog/interview-questions/hadoop-interview-questions-hdfs-2/)

Top Hadoop Interview Questions To Prepare In 2018 – HDFS (https://www.edureka.co/blog/interview-questions/hadoop-interview-questions-hdfs-2/)


(https://www.edureka.co/blog/introduction-to-apache-hadoop-hdfs/)

Introduction to Apache MapReduce and HDFS (https://www.edureka.co/blog/introduction-to-apache-hadoop-hdfs/)


(https://www.edureka.co/blog/10-reasons-why-big-data-analytics-is-the-best-career-move)

10 Reasons Why Big Data Analytics is the Best Career… (https://www.edureka.co/blog/10-reasons-why-big-data-analytics-is-the-best-career-move)


(https://www.edureka.co/blog/5-reasons-to-learn-hadoop)

5 Reasons to Learn Hadoop (https://www.edureka.co/blog/5-reasons-to-learn-hadoop)

Careers (https://www.edureka.co/careers)

Privacy policy (https://www.edureka.co/privacy-policy)

Sitemap (https://www.edureka.co/sitemap)

edureka! (https://www.edureka.co)