HDFS-1312: Re-balance disks within a Datanode

Anu Engineer, aengineer@hortonworks.com

This is an old JIRA, which intends to fix the issue that datanodes do not fill up disks evenly. A lot of community discussion has happened on this JIRA which captures pain points and possible solutions. However things have changed a bit from the time this JIRA was filed. Some JIRAs have partially addressed some of the issues and development of some new features makes this more desirable. All of that information along with a proposed solution is discussed in this proposal.

This is the original problem description from the HDFS-1312.

> Datanodes fill their storage directories unevenly, leading to situations where certain disks are full while others are significantly less used. Users at many different sites have experienced this issue, and HDFS administrators are taking steps like:
>
> - Manually re-balancing blocks in storage directories
> - Decommissioning nodes & later re-adding them

A bunch of JIRAs have partially addressed this problem over time.

- HDFS-1120 Make DataNode's block-to-device placement policy pluggable
- HDFS-1804 Add a new block-volume device choosing policy that looks at free space

Both of these JIRAs help with distributing data more evenly on the disks during writes. However the solutions offered by HDFS-1804 does not seem to have completely addressed[1] the pain points of many customers. A rather long thread of discussion from February, 2014 is available at this Linkedin thread.

1 Need for re-balancing

The need for re-balancing arises mainly due to :

- Disk replacement : This is perhaps the most important reason for data re-balancing. Development of this feature, HDFS-1362 Volume Management for DataNode makes having the ability to re-balance disks much more attractive.
- RoundRobin disk scheduling with random I/O : Random writes and deletes can lead to non-uniform distribution of data between disks, Also our block scheduling policies does not consider disk sizes.

[1] It would be interesting to get information on disk imbalances from a real cluster running free-space based policy. The tool that is proposed in this JIRA would help us get that data easily from clusters running that policy. Anecdotal evidence seems to suggest that either HDFS-1804 is not very widely deployed yet or there are pain-points that users have even with HDFS-1804.

2 Current Issues

It seems that current issues that our users run into are :

- No tools for disk balancing - We have no tools in hadoop eco-system which addresses this issue. Running balancer does not address this.[2] However users look towards balancer to solve this issue since we don't have tools that actually solves this issue.
- Lack of real world data or adoption of HDFS-1804 - HDFS-1804 introduces a totally new storage policy(free-space) into the mix. Lack of real world data is a big hurdle for most administrators and is preventing adoption of HDFS-1804.

  1. We need measurements from clusters where free-space policy is deployed vis-à-vis round-robin. This JIRA will help with that by providing a tool to measure data distribution across volumes of a node.
  2. We need to see real operational clusters that successfully migrated from round-robin policy to free-space based policy. The impact of this migration needs to be documented.
  3. We need measurements on performance, data distribution and maintenance overheads of free-space vs.round-robin policy. Things like, does all writes degenerate into a single disk write if a disk is replaced, what is the performance impact etc.

  Lack of this data is halting wider adoption of free-space disk based policy. Generally nobody wants to touch a working cluster - deploying free-space based policy is a deeply invasive change to all datanodes.
- Newer disk sizes - Newer disk sizes are emerging which causes severe imbalance in data stored per volume. This is true as users are replacing disks in JBODs. This leads to JBODs or volumes with different capacities.
- Need for expanding storage without adding more compute power - Even though it is technically feasible to upgrade few disks to expand storage, hadoop related actions are expensive from an administrator's point of view.
- Lack of ability to monitor this situation - There is no simple command/tool to check which all machines in the cluster have this problem. Balancer addresses the issue of spreading data across nodes, but spreading data across volumes or even getting a list of datanodes with data imbalance between volumes is hard.
- Fear of datanode management work flow - In one of the threads, a user recommended action was to set the *dfs.datanode.failed.volumes.tolerated* to

[2] In the HDFS-1312, we have a longer discussion about this, one of the remarks was that customers might be using clusters which are nearly full, or under provisioned cluster. But that is the reality of situation and many customers run into this scenario where they are not able to just decommission a node and re-add them - due to smaller clusters and the effort required for that. There was another suggestion that NOT mapping MR jobs to its own local disk path might be the cause of this.

a high value and wait until you lose that many volumes, so that datanode management workflow can be avoided.[3] The rationale for such a recommendation might be that a node with 12 x 3 TB disks, one disk failure means 9% of the node is affected.

- Current Apache solution is far from Ideal - *3.12. On an individual data node, how do you balance the blocks on the disk?*
  Hadoop currently does not have a method by which to do this automatically. To do this manually:

  1. Shutdown the DataNode involved
  2. Use the UNIX mv command to move the individual block replica and meta pairs from one directory to another on the selected host. On releases which have HDFS-6482 (Apache Hadoop 2.6.0+) you also need to ensure the subdir-named directory structure remains exactly the same when moving the blocks across the disks. For example, if the block replica and its meta pair were under `/data/1/dfs/dn/current/BP-1788246909-172.23.1.202-1412278461680/current/finalized/subdir0/subdir1/`, and you wanted to move it to `/data/5/` disk, then it MUST be moved into the same subdirectory structure underneath that, i.e. `/data/5/dfs/dn/current/BP-1788246909-172.23.1.202-1412278461680/current/finalized/subdir0/subdir1/`. If this is not maintained, the DN will no longer be able to locate the replicas after the move.
  3. Restart the DataNode.

- Multiple Github Solutions - There are a bunch of unmaintained github based solutions; many of them are posted as links on HDFS-1312, and a comment saying that data formats have changed so many of those tools are obsolete now. It would be nice to have an officially maintained tool from hadoop.

3 Solution

From the users point of view, spreading data across machines and on disks is something that should be done by HDFS automatically. Balancer and HDFS-1804 addresses this to an extend. However users still need tools to able to measure the data asymmetry on nodes and if asymmetry is found (due to whatever reason) an easy way to address that problem. This solution proposes that we will build a diskBalancer tool that is similar to balancer, but will focus on distributing data within a node.

Diskbalancer would have the following abilities:

- Data spread report : It would define a way to measure which machines in the cluster suffer from the data distribution asymmetry. In other words, users would be able to get a list of machines sorted by data asymmetry so that they know which machines suffer from this problem and if they want to act on it.

[3] This document does not recommend that course of action, this means you have to suffer decreased storage capacity

- Ability to balance data between volumes while datanodes are alive: Users would have the ability to move data from volume to another, or better still would have the ability to ask the machine be balanced.With HDFS-2832 we also introduced different storageTypes and different storage polices, diskBalancer would take care of spreading data without violating the constraints of HDFS-2832 while data nodes are operational.

4 Disk Balancer

Disk balancer would support 2 major functions, reporting and balancing.

4.1 Data Spread Report

In order to be able to give a report if data is spread evenly we should be able to define a metric so that we can measure how data is spread. In order to do that we define a HDFS **Volume data density** metric. This metric allows us to compare how well the data is spread across different volumes of a given node. We also define a **Node data density** metric which allows us to compare between nodes, so that we can locate nodes that require attention.

4.1.1 Volume Data Density or Intra-Node Data Density

This metric computes how much data exists on a node and what should be the ideal storage on each volume. This is computed by total data stored on that node divided by the total disk capacity on that node for each storageType.[4] This gives us the ideal storage percentage for each device. To illustrate this, let us say we have a machine with four volumes - Disk1, Disk2, Disk3, Disk4. The following table captures the metrics and how it is computed.

Table 1: Disk capacity and usage on a machine

	Disk1	Disk2	Disk3	Disk4
capacity	200 GB	300 GB	350 GB	500 GB
dfsUsed	100 GB	76 GB	300 GB	475 GB
dfsUsedRatio	0.5	0.25	0.85	0.95
volumeDataDensity	0.20	0.45	-0.15	-0.24

In this example,

$$totalCapacity = 200 + 300 + 350 + 500 = 1350GB$$

and

$$totalUsed = 100 + 76 + 300 + 475 = 951GB$$

therefore the ideal storage on each volume/disk should be

$$idealStorage = totalUsed \div totalCapacity = 951 \div 1350 = 0.70$$

[4]HDFS-2832, Introduced storageTypes like RAM_DISK, SSD, DISK and ARCHIVE. diskBalancer will skip RAM_DISKs since it is transient in nature

or 70% of capacity of each disk. volume Data Density is difference from idealStorage and current dfsUsedRatio , in other words volume data density for disk 1 is

$$volumeDataDensity = idealStorage - dfsUsedRatio = 0.70 - 0.50 = 0.20$$

A positive value for volumeDataDensity indicates that disk is under utilized and a negative value indicates that disk is over-utilized in relation to the current idealStorage target.

4.1.2 Node Data Density or Inter-Node Data Density

Once we have volumeDataDensity defined, we are now in a position to compare which all nodes in the data center would need our attention. This is done by computing nodes with maximum skew from the idealStorage values, for that we simply add up all absolute values of volumeDataDensity.

$$nodeDataDensity = \sum_{d_i \in volumeDataDensity(I)} |d_i|$$

nodeDataDensity allows us to compare nodes that need our attention in a given cluster. Lower nodeDataDensity values indicate better spread and higher values indicate more skewed data distribution.

4.1.3 Reports

Once we have volumeDataDensity and nodeDataDensity, diskBalancer can now answer questions like *what are the top 20 nodes in my cluster that has skewed data distribution ?* or given a node we can get the volumeDataDensity.

./diskBalancer -top would print out something like the top 20 nodes in that given cluster which has most data density skew, or a command like **./diskbalancer -node *datanodeID*** would print out the volumes and volumeDataDensity of each volumes.

It is feasible to compute these matrices in diskBalancer, but it is also trivial to expose these in dataNode itself. The advantage of exposing these as DataNodeMetrics would be that they are universally available, which would allow tools like Apache Ambari, Cloudera Manager or plain old python or bash scripts to leverage these metrics.

4.2 Disk Balancing

Once we know that certain node needs balancing we compute/read the current volumeDataDensity. With this information we can easily decide which volumes are over-provisioned vs. which are under-provisioned. In order to move data from one volume to another the dataNode would add an protoc based RPC similar to one used by balancer. [5].

Once we have that RPC moving a block from one volume to another is straight forward. Here is a pseudo-code for the move. Please note that all moves are between same disk types, that is sourceVolume and thisVolume is always assumed to same StorageType.

- For each volume in a StorageType {SSD, DISK, ARCHIVE }

[5] DataTransferProtocol.replaceBlock

- if volume.dfsUsed <idealStorage for that StorageType
  - copy a finalized block from sourceVolume to thisVolume
  - verify checksums match between the original and new block
- Update volumeMap (in-Memory DataNode metadata)
- delete older copy of the block

This would allow users to run diskBalancer just like running balancer. With the diskBalancer users can replace disks[6] without having to worry about decommissioning a node, since diskBalancer would move data from one volume to another while nodes are alive.

5 FAQ

- Volume data changes due to write and deletes: We are moving block by block, and we could recompute the volumeDataDensity after a move. Hence datanodes write/delete should not be a problem.[7]
- Windows OS delete semantics : In the unix world, "rm" means remove from the namespace and remove the file when the last file handle is closed. However in windows world, if a block is open for reading then the remove operation will fail. There are couple of ways to deal with this.
  1. Retry the delete operation
  2. Do not support windows for time being
- Datanode failures : What happens in case of datanode failure after a copy is done but update or delete is not performed. We had a JIRA to handle duplicate blocks in DataNodes HDFS-1940, which was never fixed. We will test for this, and make sure that this does not cause an operational issue for datanodes. HDFS lazy persist feature also deals with this issue currently.
- Bandwidth Control and Excluding volumes : yes, we will support bandwidth control as well as excluding storageType or volumePaths from diskBalancing
- Make this part of balancer : Eventually yes, but for ease of testing we would like to get this out as independent tool that gets collapsed into balancer later.
- Impact on other HDFS features : Does this impact Encryption or snapshots. AFAIK, this should not impact those, but we will test this with all those features turned on.
- Will this disrupt RollingUpgrade or SafeMode : Again it should not. But as a policy - just like balancer, diskBalancer will also quit operation if the cluster is in safe mode or doing rollingUpgrade.

[6] by leveraging HDFS-1362
[7] Specifics of DataNode space calculation is in this JIRA - HDFS-3570