**HDFS Questions & Answers**

**1)How does Hadoop Namenode failover process works?**   
**Answer:** In a typical High Availability cluster, two separate machines are configured as NameNodes. At any point in time, exactly one of the NameNodes is in an Active state, and the other is in a Standby state. The Active NameNode is responsible for all client operations in the cluster, while the Standby is simply acting as a slave, maintaining enough state to provide a fast failover if necessary.  
In order for the Standby node to keep its state synchronized with the Active node, both nodes communicate with a group of separate daemons called “JournalNodes” (JNs). When any namespace modification is performed by the Active node, it durably logs a record of the modification to a majority of these JNs. The Standby node is capable of reading the edits from the JNs, and is constantly watching them for changes to the edit log. As the Standby Node sees the edits, it applies them to its own namespace. In the event of a failover, the Standby will ensure that it has read all of the edits from the JounalNodes before promoting itself to the Active state. This ensures that the namespace state is fully synchronized before a failover occurs.  
In order to provide a fast failover, it is also necessary that the Standby node have up-to-date information regarding the location of blocks in the cluster. In order to achieve this, the DataNodes are configured with the location of both NameNodes, and send block location information and heartbeats to both.  
It is vital for the correct operation of an HA cluster that only one of the NameNodes be Active at a time. Otherwise, the namespace state would quickly diverge between the two, risking data loss or other incorrect results. In order to ensure this property and prevent the so-called “split-brain scenario,” the JournalNodes will only ever allow a single NameNode to be a writer at a time. During a failover, the NameNode which is to become active will simply take over the role of writing to the JournalNodes, which will effectively prevent the other NameNode from continuing in the Active state, allowing the new Active to safely proceed with failover.  
  
**2) How can we initiate a manual failover when automatic failover is configured?**   
**Answer:** Even if automatic failover is configured, you may initiate a manual failover using the same hdfs haadmin command. It will perform a coordinated failover.   
  
**3) When not use Hadoop?**   
**Answer:** 1)Real Time Analytics: If you want to do some Real Time Analytics, where you are expecting result quickly, Hadoop should not be used directly. It is because Hadoop works on batch processing, hence response time is high.  
2. Not a Replacement for Existing Infrastructure: Hadoop is not a replacement for your existing data processing infrastructure. However, you can use Hadoop along with it.  
3. Multiple Smaller Datasets:Hadoop framework is not recommended for small-structured datasets as you have other tools available in market which can do this work quite easily and at a fast pace than Hadoop like MS Excel, RDBMS etc. For a small data analytics, Hadoop can be costlier than other tools.  
4. Novice Hadoopers:Unless you have a better understanding of the Hadoop framework, it’s not suggested to use Hadoop for production. Hadoop is a technology which should come with a disclaimer: “Handle with care”. You should know it before you use it or else you will end up like the kid below.  
5. Security is the primary Concern:Many enterprises especially within highly regulated industries dealing with sensitive data aren’t able to move as quickly as they would like towards implementing Big Data projects and Hadoop.

**4) When To Use Hadoop?**   
**Answer:** 1. Data Size and Data Diversity:When you are dealing with huge volumes of data coming from various sources and in a variety of formats then you can say that you are dealing with Big Data. In this case, Hadoop is the right technology for you.  
2. Future Planning: It is all about getting ready for challenges you may face in future. If you anticipate Hadoop as a future need then you should plan accordingly. To implement Hadoop on you data you should first understand the level of complexity of data and the rate with which it is going to grow. So, you need a cluster planning. It may begin with building a small or medium cluster in your industry as per data (in GBs or few TBs ) available at present and scale up your cluster in future depending on the growth of your data.  
3. Multiple Frameworks for Big Data: There are various tools for various purposes. Hadoop can be integrated with multiple analytic tools to get the best out of it, like Mahout for Machine-Learning, R and Python for Analytics and visualization, Python, Spark for real time processing, MongoDB and Hbase for Nosql database, Pentaho for BI etc.  
4. Lifetime Data Availability: When you want your data to be live and running forever, it can be achieved using Hadoop’s scalability. There is no limit to the size of cluster that you can have. You can increase the size anytime as per your need by adding datanodes to it with The bottom line is use the right technology as per your need.  
  
**5) When you run start-dfs.sh or stop-dfs.sh, you get the following warning message:WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable.How to fix this warning message?**  
**Answer:** The reason you saw that warning is the native Hadoop library $HADOOP\_HOME/lib/native/libhadoop.so.1.0.0 was actually compiled on 32 bit.Anyway, it's just a warning, and won't impact Hadoop's functionalities. Here is the way if you do want to eliminate this warning, download the source code of Hadoop and recompile libhadoop.so.1.0.0 on 64bit system, then replace the 32bit one.   
  
**6) What platforms and Java versions does Hadoop run on?**   
**Answer:** Java 1.6.x or higher, preferably, Linux and Windows are the supported operating systems, but BSD, Mac OS/X, and OpenSolaris are known to work. (Windows requires the installation of Cygwin).   
**7) As we talk about Hadoop is Highly scalable how well does it Scale?**   
**Answer:** Hadoop has been demonstrated on clusters of up to 4000 nodes. Sort performance on 900 nodes is good (sorting 9TB of data on 900 nodes takes around 1.8 hours) and improving using these non-default configuration values:  
dfs.block.size = 134217728  
dfs.namenode.handler.count = 40  
mapred.reduce.parallel.copies = 20  
mapred.child.java.opts = -Xmx512m  
fs.inmemory.size.mb = 200  
io.sort.factor = 100  
io.sort.mb = 200  
io.file.buffer.size = 131072  
  
Sort performances on 1400 nodes and 2000 nodes are pretty good too - sorting 14TB of data on a 1400-node cluster takes 2.2 hours; sorting 20TB on a 2000-node cluster takes 2.5 hours. The updates to the above configuration being:  
mapred.job.tracker.handler.count = 60  
mapred.reduce.parallel.copies = 50  
tasktracker.http.threads = 50  
mapred.child.java.opts = -Xmx1024m  
  
**8) What kind of hardware scales best for Hadoop?**   
**Answer:** The short answer is dual processor/dual core machines with 4-8GB of RAM using ECC memory, depending upon workflow needs. Machines should be moderately high-end commodity machines to be most cost-effective and typically cost 1/2 - 2/3 the cost of normal production application servers but are not desktop-class machines.   
  
**9) Among the software questions for setting up and running Hadoop, there a few other questions that relate to hardware scaling:   
  
i)What are the optimum machine configurations for running a hadoop cluster?  
ii) Should I use a smaller number of high end/performance machines or are a larger number of "commodity" machines?  
iii)How does the Hadoop/Parallel Distributed Processing community define "commodity"?**  
**Answer:** In answer to i and ii above, we can group the possible hardware options in to 3 rough categories:  
A)Database class machine with many (>10) fast SAS drives and >10GB memory, dual or quad x quad core cpu's. With an approximate cost of $20K.  
B)Generic production machine with 2 x 250GB SATA drives, 4-12GB RAM, dual x dual core CPU's (=Dell 1950). Cost is about $2-5K.  
C) POS beige box machine with 2 x SATA drives of variable size, 4 GB RAM, single dual core CPU. Cost is about $1K.  
  
For a $50K budget, most users would take 25xB over 50xC due to simpler and smaller admin issues even though cost/performance would be nominally about the same. Most users would avoid 2x(A) like the plague.  
  
For the discussion to iii, "commodity" hardware is best defined as consisting of standardized, easily available components which can be purchased from multiple distributors/retailers. Given this definition there are still ranges of quality that can be purchased for your cluster. As mentioned above, users generally avoid the low-end, cheap solutions. The primary motivating force to avoid low-end solutions is "real" cost; cheap parts mean greater number of failures requiring more maintanance/cost. Many users spend $2K-$5K per machine.  
  
More specifics:  
Multi-core boxes tend to give more computation per dollar, per watt and per unit of operational maintenance. But the highest clockrate processors tend to not be cost-effective, as do the very largest drives. So moderately high-end commodity hardware is the most cost-effective for Hadoop today.  
  
Some users use cast-off machines that were not reliable enough for other applications. These machines originally cost about 2/3 what normal production boxes cost and achieve almost exactly 1/2 as much. Production boxes are typically dual CPU's with dual cores.  
  
RAM:  
Many users find that most hadoop applications are very small in memory consumption. Users tend to have 4-8 GB machines with 2GB probably being too little. Hadoop benefits greatly from ECC memory, which is not low-end, however using ECC memory is RECOMMENDED  
  
**10) I have a new node I want to add to a running Hadoop cluster; how do I start services on just one node?**  
**Answer:** This also applies to the case where a machine has crashed and rebooted, etc, and you need to get it to rejoin the cluster. You do not need to shutdown and/or restart the entire cluster in this case.  
First, add the new node's DNS name to the conf/slaves file on the master node.  
Then log in to the new slave node and execute:  
  
$ cd path/to/hadoop  
$ bin/hadoop-daemon.sh start datanode  
$ bin/hadoop-daemon.sh start tasktracker  
  
If you are using the dfs.include/mapred.include functionality, you will need to additionally add the node to the dfs.include/mapred.include file, then issue hadoop dfsadmin -refreshNodes and hadoop mradmin -refreshNodes so that the NameNode and JobTracker know of the additional node that has been added.

**11) Is there an easy way to see the status and health of a cluster?**   
**Answer:** You can also see some basic HDFS cluster health data by running:   
$ bin/hadoop dfsadmin -report   
  
**12) How much network bandwidth might I need between racks in a medium size (40-80 node) Hadoop cluster?**   
**Answer:** The true answer depends on the types of jobs you're running. As a back of the envelope calculation one might figure something like this:   
60 nodes total on 2 racks = 30 nodes per rack Each node might process about 100MB/sec of data In the case of a sort job where the intermediate data is the same size as the input data, that means each node needs to shuffle 100MB/sec of data In aggregate, each rack is then producing about 3GB/sec of data However, given even reducer spread across the racks, each rack will need to send 1.5GB/sec to reducers running on the other rack. Since the connection is full duplex, that means you need 1.5GB/sec of bisection bandwidth for this theoretical job. So that's 12Gbps.   
However, the above calculations are probably somewhat of an upper bound. A large number of jobs have significant data reduction during the map phase, either by some kind of filtering/selection going on in the Mapper itself, or by good usage of Combiners. Additionally, intermediate data compression can cut the intermediate data transfer by a significant factor. Lastly, although your disks can probably provide 100MB sustained throughput, it's rare to see a MR job which can sustain disk speed IO through the entire pipeline. So, I'd say my estimate is at least a factor of 2 too high.   
So, the simple answer is that 4-6Gbps is most likely just fine for most practical jobs. If you want to be extra safe, many inexpensive switches can operate in a "stacked" configuration where the bandwidth between them is essentially backplane speed. That should scale you to 96 nodes with plenty of headroom. Many inexpensive gigabit switches also have one or two 10GigE ports which can be used effectively to connect to each other or to a 10GE core.   
  
**13) I am seeing connection refused in the logs. How do I troubleshoot this?**   
**Answer:** You get a ConnectionRefused Exception when there is a machine at the address specified, but there is no program listening on the specific TCP port the client is using -and there is no firewall in the way silently dropping TCP connection requests.   
Unless there is a configuration error at either end, a common cause for this is the Hadoop service isn't running.   
This stack trace is very common when the cluster is being shut down -because at that point Hadoop services are being torn down across the cluster, which is visible to those services and applications which haven't been shut down themselves. Seeing this error message during cluster shutdown is not anything to worry about.   
If the application or cluster is not working, and this message appears in the log, then it is more serious.   
Check the hostname the client using is correct. If it's in a Hadoop configuration option: examine it carefully, try doing an ping by hand.   
Check the IP address the client is trying to talk to for the hostname is correct.   
Make sure the destination address in the exception isn't 0.0.0.0 -this means that you haven't actually configured the client with the real address for that service, and instead it is picking up the server-side property telling it to listen on every port for connections.   
If the error message says the remote service is on "127.0.0.1" or "localhost" that means the configuration file is telling the client that the service is on the local server. If your client is trying to talk to a remote system, then your configuration is broken.   
Check that there isn't an entry for your hostname mapped to 127.0.0.1 or 127.0.1.1 in /etc/hosts (Ubuntu is notorious for this).   
Check the port the client is trying to talk to using matches that the server is offering a service on. The netstat command is useful there.   
On the server, try a telnet localhost (port) to see if the port is open there.   
On the client, try a telnet (server) (port) to see if the port is accessible remotely.   
Try connecting to the server/port from a different machine, to see if it just the single client misbehaving.   
If your client and the server are in different subdomains, it may be that the configuration of the service is only publishing the basic hostname, rather than the Fully Qualified Domain Name. The client in the different subdomain can be unintentionally attempt to bind to a host in the local subdomain —and failing.   
If you are using a Hadoop-based product from a third party, -please use the support channels provided by the vendor.   
  
**14)Does Hadoop require SSH?**   
**Answer:** Hadoop provided scripts (e.g., start-mapred.sh and start-dfs.sh) use ssh in order to start and stop the various daemons and some other utilities. The Hadoop framework in itself does not require ssh. Daemons (e.g. TaskTracker and DataNode) can also be started manually on each node without the script's help.  
  
**15) What does NFS: Cannot create lock on (some dir) mean?**   
**Answer:** This actually is not a problem with Hadoop, but represents a problem with the setup of the environment it is operating.   
Usually, this error means that the NFS server to which the process is writing does not support file system locks. NFS prior to v4 requires a locking service daemon to run (typically rpc.lockd) in order to provide this functionality. NFSv4 has file system locks built into the protocol.   
In some (rarer) instances, it might represent a problem with certain Linux kernels that did not implement the flock() system call properly.   
It is highly recommended that the only NFS connection in a Hadoop setup be the place where the NameNode writes a secondary or tertiary copy of the fsimage and edits log. All other users of NFS are not recommended for optimal performance.

**16) If I add new DataNodes to the cluster will HDFS move the blocks to the newly added nodes in order to balance disk space utilization between the nodes?**   
**Answer:** No, HDFS will not move blocks to new nodes automatically. However, newly created files will likely have their blocks placed on the new nodes.   
There are several ways to rebalance the cluster manually.   
Select a subset of files that take up a good percentage of your disk space; copy them to new locations in HDFS; remove the old copies of the files; rename the new copies to their original names.   
A simpler way, with no interruption of service, is to turn up the replication of files, wait for transfers to stabilize, and then turn the replication back down.  
Yet another way to re-balance blocks is to turn off the data-node, which is full, wait until its blocks are replicated, and then bring it back again. The over-replicated blocks will be randomly removed from different nodes, so you really get them rebalanced not just removed from the current node.  
  
Finally, you can use the bin/start-balancer.sh command to run a balancing process to move blocks around the cluster automatically.   
  
**17) What is the purpose of the secondary name-node?**   
**Answer:** The term "secondary name-node" is somewhat misleading. It is not a name-node in the sense that data-nodes cannot connect to the secondary name-node, and in no event it can replace the primary name-node in case of its failure.   
The only purpose of the secondary name-node is to perform periodic checkpoints. The secondary name-node periodically downloads current name-node image and edits log files, joins them into new image and uploads the new image back to the (primary and the only) name-node.   
So if the name-node fails and you can restart it on the same physical node then there is no need to shutdown data-nodes, just the name-node need to be restarted. If you cannot use the old node anymore you will need to copy the latest image somewhere else. The latest image can be found either on the node that used to be the primary before failure if available; or on the secondary name-node. The latter will be the latest checkpoint without subsequent edits logs, that is the most recent name space modifications may be missing there. You will also need to restart the whole cluster in this case.   
  
**18) Does the name-node stay in safe mode till all under-replicated files are fully replicated?**   
**Answer:** No. During safe mode replication of blocks is prohibited. The name-node awaits when all or majority of data-nodes report their blocks.   
Depending on how safe mode parameters are configured the name-node will stay in safe mode until a specific percentage of blocks of the system is minimally replicated dfs.replication.min. If the safe mode threshold dfs.safemode.threshold.pct is set to 1 then all blocks of all files should be minimally replicated.  
Minimal replication does not mean full replication. Some replicas may be missing and in order to replicate them the name-node needs to leave safe mode.   
  
**19) How do I set up a hadoop node to use multiple volumes?**   
**Answer:** Data-nodes can store blocks in multiple directories typically allocated on different local disk drives. In order to setup multiple directories one needs to specify a comma separated list of pathnames as a value of the configuration parameter dfs.datanode.data.dir. Data-nodes will attempt to place equal amount of data in each of the directories.   
The name-node also supports multiple directories, which in the case store the name space image and the edits log. The directories are specified via the dfs.namenode.name.dir configuration parameter. The name-node directories are used for the name space data replication so that the image and the log could be restored from the remaining volumes if one of them fails.   
  
**20) What happens if one Hadoop client renames a file or a directory containing this file while another client is still writing into it?**   
**Answer:** Starting with release hadoop-0.15, a file will appear in the name space as soon as it is created. If a writer is writing to a file and another client renames either the file itself or any of its path components, then the original writer will get an IOException either when it finishes writing to the current block or when it closes the file.   
  
**21)I want to make a large cluster smaller by taking out a bunch of nodes simultaneously. How can this be done?**   
**Answer:** On a large cluster removing one or two data-nodes will not lead to any data loss, because name-node will replicate their blocks as long as it will detect that the nodes are dead. With a large number of nodes getting removed or dying the probability of losing data is higher.   
Hadoop offers the decommission feature to retire a set of existing data-nodes. The nodes to be retired should be included into the exclude file, and the exclude file name should be specified as a configuration parameter dfs.hosts.exclude. This file should have been specified during namenode startup. It could be a zero length file. You must use the full hostname, ip or ip:port format in this file. (Note that some users have trouble using the host name. If your namenode shows some nodes in "Live" and "Dead" but not decommission, try using the full ip:port.) Then the shell command   
bin/hadoop dfsadmin -refreshNodes   
should be called, which forces the name-node to re-read the exclude file and start the decommission process.   
Decommission is not instant since it requires replication of potentially a large number of blocks and we do not want the cluster to be overwhelmed with just this one job. The decommission progress can be monitored on the name-node Web UI. Until all blocks are replicated the node will be in "Decommission In Progress" state. When decommission is done the state will change to "Decommissioned". The nodes can be removed whenever decommission is finished.   
The decommission process can be terminated at any time by editing the configuration or the exclude files and repeating the -refreshNodes command.   
  
**22) Does Wildcard characters work correctly in FsShell?**  
**Answer:** When you issue a command in FsShell, you may want to apply that command to more than one file. FsShell provides a wildcard character to help you do so. The \* (asterisk) character can be used to take the place of any set of characters. For example, if you would like to list all the files in your account which begin with the letter x, you could use the ls command with the \* wildcard:   
bin/hadoop dfs -ls x\*   
Sometimes, the native OS wildcard support causes unexpected results. To avoid this problem, Enclose the expression in Single or Double quotes and it should work correctly.   
bin/hadoop dfs -ls 'in\*'   
  
**23) Can I have multiple files in HDFS use different block sizes?**   
**Answer:** Yes. HDFS provides api to specify block size when you create a file.  
See FileSystem.create(Path, overwrite, bufferSize, replication, blockSize, progress)  
  
**24) Does HDFS make block boundaries between records?**   
**Answer:** No, HDFS does not provide record-oriented API and therefore is not aware of records and boundaries between them.   
  
**25) What happens when two clients try to write into the same HDFS file?**   
**Answer:** HDFS supports exclusive writes only.  
When the first client contacts the name-node to open the file for writing, the name-node grants a lease to the client to create this file. When the second client tries to open the same file for writing, the name-node will see that the lease for the file is already granted to another client, and will reject the open request for the second client.

**26) How to limit Data node's disk usage?**   
**Answer:** Use dfs.datanode.du.reserved configuration value in $HADOOP\_HOME/conf/hdfs-site.xml for limiting disk usage.  
value = 182400  
  
**27) On an individual data node, how do you balance the blocks on the disk?**   
**Answer:** 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.   
  
**28) What does "file could only be replicated to 0 nodes, instead of 1" mean?**   
**Answer:** The NameNode does not have any available DataNodes. This can be caused by a wide variety of reasons. Check the DataNode logs, the NameNode logs, network connectivity.   
  
**29) If the NameNode loses its only copy of the fsimage file, can the file system be recovered from the DataNodes?**   
**Answer:** No. This is why it is very important to configure dfs.namenode.name.dir to write to two filesystems on different physical hosts, use the SecondaryNameNode, etc.   
  
**30) I got a warning on the NameNode web UI "WARNING : There are about 32 missing blocks. Please check the log or run fsck." What does it mean?**   
**Answer:** This means that 32 blocks in your HDFS installation don’t have a single replica on any of the live DataNodes.   
Block replica files can be found on a DataNode in storage directories specified by configuration parameter dfs.datanode.data.dir. If the parameter is not set in the DataNode’s hdfs-site.xml, then the default location /tmp will be used. This default is intended to be used only for testing. In a production system this is an easy way to lose actual data, as local OS may enforce recycling policies on /tmp. Thus the parameter must be overridden.  
If dfs.datanode.data.dir correctly specifies storage directories on all DataNodes, then you might have a real data loss, which can be a result of faulty hardware or software bugs. If the file(s) containing missing blocks represent transient data or can be recovered from an external source, then the easiest way is to remove (and potentially restore) them. Run fsck in order to determine which files have missing blocks. If you would like (highly appreciated) to further investigate the cause of data loss, then you can dig into NameNode and DataNode logs. From the logs one can track the entire life cycle of a particular block and its replicas.

**31) If a block size of 64MB is used and a file is written that uses less than 64MB, will 64MB of disk space be consumed?**   
**Answer:** Short answer: No.  
Longer answer: Since HFDS does not do raw disk block storage, there are two block sizes in use when writing a file in HDFS: the HDFS blocks size and the underlying file system's block size. HDFS will create files up to the size of the HDFS block size as well as a meta file that contains CRC32 checksums for that block. The underlying file system store that file as increments of its block size on the actual raw disk, just as it would any other file.  
  
**32) What does the message "Operation category READ/WRITE is not supported in state standby" mean?**   
**Answer:** In an HA-enabled cluster, DFS clients cannot know in advance which namenode is active at a given time. So when a client contacts a namenode and it happens to be the standby, the READ or WRITE operation will be refused and this message is logged. The client will then automatically contact the other namenode and try the operation again. As long as there is one active and one standby namenode in the cluster, this message can be safely ignored.   
  
**33)On what concept the Hadoop framework works?**   
**Answer:** Hadoop Framework works on the following two core components-   
1)HDFS – Hadoop Distributed File System is the java based file system for scalable and reliable storage of large datasets. Data in HDFS is stored in the form of blocks and it operates on the Master Slave Architecture.   
2)Hadoop MapReduce-This is a java based programming paradigm of Hadoop framework that provides scalability across various Hadoop clusters. MapReduce distributes the workload into various tasks that can run in parallel. Hadoop jobs perform 2 separate tasks- job. The map job breaks down the data sets into key-value pairs or tuples. The reduce job then takes the output of the map job and combines the data tuples to into smaller set of tuples. The reduce job is always performed after the map job is executed.   
  
**34)What is Hadoop streaming?**   
**Answer:** Hadoop distribution has a generic application programming interface for writing Map and Reduce jobs in any desired programming language like Python, Perl, Ruby, etc. This is referred to as Hadoop Streaming. Users can create and run jobs with any kind of shell scripts or executable as the Mapper or Reducers.   
  
**35)Explain about the process of inter cluster data copying.?**   
**Answer:** HDFS provides a distributed data copying facility through the DistCP from source to destination. If this data copying is within the hadoop cluster then it is referred to as inter cluster data copying. DistCP requires both source and destination to have a compatible or same version of hadoop.   
  
**36)Differentiate between Structured and Unstructured data?**   
**Answer:** Data which can be stored in traditional database systems in the form of rows and columns, for example the online purchase transactions can be referred to as Structured Data. Data which can be stored only partially in traditional database systems, for example, data in XML records can be referred to as semi structured data. Unorganized and raw data that cannot be categorized as semi structured or structured data is referred to as unstructured data. Facebook updates, Tweets on Twitter, Reviews, web logs, etc. are all examples of unstructured data.   
  
**37)Explain the difference between NameNode, Backup Node and Checkpoint NameNode?**  
**Answer:** NameNode: NameNode is at the heart of the HDFS file system which manages the metadata i.e. the data of the files is not stored on the NameNode but rather it has the directory tree of all the files present in the HDFS file system on a hadoop cluster. NameNode uses two files for the namespace-   
fsimage file- It keeps track of the latest checkpoint of the namespace.   
edits file-It is a log of changes that have been made to the namespace since checkpoint.   
Checkpoint Node:   
Checkpoint Node keeps track of the latest checkpoint in a directory that has same structure as that of NameNode’s directory. Checkpoint node creates checkpoints for the namespace at regular intervals by downloading the edits and fsimage file from the NameNode and merging it locally. The new image is then again updated back to the active NameNode.   
BackupNode:   
Backup Node also provides check pointing functionality like that of the checkpoint node but it also maintains its up-to-date in-memory copy of the file system namespace that is in sync with the active NameNode.   
  
**38)How can you overwrite the replication factors in HDFS?**   
**Answer:** The replication factor in HDFS can be modified or overwritten in 2 ways-   
1)Using the Hadoop FS Shell, replication factor can be changed per file basis using the below command-   
$hadoop fs –setrep –w 2 /my/test\_file (test\_file is the filename whose replication factor will be set to 2)   
2)Using the Hadoop FS Shell, replication factor of all files under a given directory can be modified using the below command-   
3)$hadoop fs –setrep –w 5 /my/test\_dir (test\_dir is the name of the directory and all the files in this directory will have a replication factor set to 5) 

**39)Explain what happens if during the PUT operation, HDFS block is assigned a replication factor 1 instead of the default value 3?**   
**Answer:** Replication factor is a property of HDFS that can be set accordingly for the entire cluster to adjust the number of times the blocks are to be replicated to ensure high data availability. For every block that is stored in HDFS, the cluster will have n-1 duplicated blocks. So, if the replication factor during the PUT operation is set to 1 instead of the default value 3, then it will have a single copy of data. Under these circumstances when the replication factor is set to 1 ,if the DataNode crashes under any circumstances, then only single copy of the data would be lost.   
  
**40)What is the process to change the files at arbitrary locations in HDFS?**   
Answer)HDFS does not support modifications at arbitrary offsets in the file or multiple writers but files are written by a single writer in append only format i.e. writes to a file in HDFS are always made at the end of the file.

**41)Explain about the indexing process in HDFS?**  
**Answer:** Indexing process in HDFS depends on the block size. HDFS stores the last part of the data that further points to the address where the next part of data chunk is stored.   
  
**42)What is a rack awareness and on what basis is data stored in a rack?**  
**Answer:** All the data nodes put together form a storage area i.e. the physical location of the data nodes is referred to as Rack in HDFS. The rack information i.e. the rack id of each data node is acquired by the NameNode. The process of selecting closer data nodes depending on the rack information is known as Rack Awareness.   
The contents present in the file are divided into data block as soon as the client is ready to load the file into the hadoop cluster. After consulting with the NameNode, client allocates 3 data nodes for each data block. For each data block, there exists 2 copies in one rack and the third copy is present in another rack. This is generally referred to as the Replica Placement Policy.   
  
**43)What happens to a NameNode that has no data?**  
  
Answer)There does not exist any NameNode without data. If it is a NameNode then it should have some sort of data in it.   
  
**44) What happens when a user submits a Hadoop job when the NameNode is down- does the job get in to hold or does it fail.**  
**Answer:** The Hadoop job fails when the NameNode is down.   
  
**45) What happens when a user submits a Hadoop job when the Job Tracker is down- does the job get in to hold or does it fail.**   
**Answer:** The Hadoop job fails when the Job Tracker is down.   
  
**46) Whenever a client submits a hadoop job, who receives it?**  
**Answer:** NameNode receives the Hadoop job which then looks for the data requested by the client and provides the block information. JobTracker takes care of resource allocation of the hadoop job to ensure timely completion.   
  
**47) What do you understand by edge nodes in Hadoop?**   
**Answer:** Edges nodes are the interface between hadoop cluster and the external network. Edge nodes are used for running cluster adminstration tools and client applications.Edge nodes are also referred to as gateway nodes.   
  
**48)What are real-time industry applications of Hadoop?**  
**Answer:** Hadoop, well known as Apache Hadoop, is an open-source software platform for scalable and distributed computing of large volumes of data. It provides rapid, high performance and cost-effective analysis of structured and unstructured data generated on digital platforms and within the enterprise. It is used in almost all departments and sectors today.Some of the instances where Hadoop is used:  
Managing traffic on streets.  
Streaming processing.  
Content Management and Archiving Emails.  
Processing Rat Brain Neuronal Signals using a Hadoop Computing Cluster.  
Fraud detection and Prevention.  
Advertisements Targeting Platforms are using Hadoop to capture and analyze click stream, transaction, video and social media data.  
Managing content, posts, images and videos on social media platforms.  
Analyzing customer data in real-time for improving business performance.  
Public sector fields such as intelligence, defense, cyber security and scientific research.  
Financial agencies are using Big Data Hadoop to reduce risk, analyze fraud patterns, identify rogue traders, more precisely target their marketing campaigns based on customer segmentation, and improve customer satisfaction.  
Getting access to unstructured data like output from medical devices, doctor’s notes, lab results, imaging reports, medical correspondence, clinical data, and financial data.  
  
  
**49)What all modes Hadoop can be run in?**  
**Answer:** Hadoop can run in three modes:  
Standalone Mode: Default mode of Hadoop, it uses local file stystem for input and output operations. This mode is mainly used for debugging purpose, and it does not support the use of HDFS. Further, in this mode, there is no custom configuration required for mapred-site.xml, core-site.xml, hdfs-site.xml files. Much faster when compared to other modes.  
Pseudo-Distributed Mode (Single Node Cluster): In this case, you need configuration for all the three files mentioned above. In this case, all daemons are running on one node and thus, both Master and Slave node are the same.  
Fully Distributed Mode (Multiple Cluster Node): This is the production phase of Hadoop (what Hadoop is known for) where data is used and distributed across several nodes on a Hadoop cluster. Separate nodes are allotted as Master and Slave.  
  
**50)Explain the major difference between HDFS block and InputSplit?**  
**Answer:** In simple terms, block is the physical representation of data while split is the logical representation of data present in the block. Split acts as an intermediary between block and mapper.  
Suppose we have two blocks:  
Block 1: ii bbhhaavveesshhll  
Block 2: Ii inntteerrvviieewwll  
Now, considering the map, it will read first block from ii till ll, but does not know how to process the second block at the same time. Here comes Split into play, which will form a logical group of Block1 and Block 2 as a single block.  
It then forms key-value pair using inputformat and records reader and sends map for further processing With inputsplit, if you have limited resources, you can increase the split size to limit the number of maps. For instance, if there are 10 blocks of 640MB (64MB each) and there are limited resources, you can assign ‘split size’ as 128MB. This will form a logical group of 128MB, with only 5 maps executing at a time.  
However, if the ‘split size’ property is set to false, whole file will form one inputsplit and is processed by single map, consuming more time when the file is bigger.   
  
**51)What are the most common Input Formats in Hadoop?**  
**Answer:** There are three most common input formats in Hadoop:  
Text Input Format: Default input format in Hadoop.  
Key Value Input Format: used for plain text files where the files are broken into lines  
Sequence File Input Format: used for reading files in sequence  
  
**52)What is Speculative Execution in Hadoop?**  
**Answer:** One limitation of Hadoop is that by distributing the tasks on several nodes, there are chances that few slow nodes limit the rest of the program. Tehre are various reasons for the tasks to be slow, which are sometimes not easy to detect. Instead of identifying and fixing the slow-running tasks, Hadoop tries to detect when the task runs slower than expected and then launches other equivalent task as backup. This backup mechanism in Hadoop is Speculative Execution.  
It creates a duplicate task on another disk. The same input can be processed multiple times in parallel. When most tasks in a job comes to completion, the speculative execution mechanism schedules duplicate copies of remaining tasks (which are slower) across the nodes that are free currently. When these tasks finish, it is intimated to the JobTracker. If other copies are executing speculatively, Hadoop notifies the TaskTrackers to quit those tasks and reject their output.  
  
**53)What is Fault Tolerance?**   
**Answer:** Suppose you have a file stored in a system, and due to some technical problem that file gets destroyed. Then there is no chance of getting the data back present in that file. To avoid such situations, Hadoop has introduced the feature of fault tolerance in HDFS. In Hadoop, when we store a file, it automatically gets replicated at two other locations also. So even if one or two of the systems collapse, the file is still available on the third system.   
  
**54)What is a heartbeat in HDFS?**  
**Answer:** A heartbeat is a signal indicating that it is alive. A datanode sends heartbeat to Namenode and task tracker will send its heart beat to job tracker. If the Namenode or job tracker does not receive heart beat then they will decide that there is some problem in datanode or task tracker is unable to perform the assigned task.   
  
**55)How to keep HDFS cluster balanced?**   
**Answer:** When copying data into HDFS, it’s important to consider cluster balance. HDFS works best when the file blocks are evenly spread across the cluster, so you want to ensure that distcp doesn’t disrupt this. For example, if you specified -m 1, a single map would do the copy, which — apart from being slow and not using the cluster resources efficiently — would mean that the first replica of each block would reside on the node running the map (until the disk filled up). The second and third replicas would be spread across the cluster, but this one node would be unbalanced. By having more maps than nodes in the cluster, this problem is avoided. For this reason, it’s best to start by running distcp with the default of 20 maps per node.+   
However, it’s not always possible to prevent a cluster from becoming unbalanced. Perhaps you want to limit the number of maps so that some of the nodes can be used by other jobs. In this case, you can use the balancer tool (see Balancer) to subsequently even out the block distribution across the cluster.   
  
**56)How to deal with small files in Hadoop?**  
**Answer:** Hadoop Archives (HAR) offers an effective way to deal with the small files problem.  
Hadoop Archives or HAR is an archiving facility that packs files in to HDFS blocks efficiently and hence HAR can be used to tackle the small files problem in Hadoop. HAR is created from a collection of files and the archiving tool (a simple command) will run a MapReduce job to process the input files in parallel and create an archive file.  
  
HAR command   
hadoop archive -archiveName myhar.har /input/location /output/location   
Once a .har file is created, you can do a listing on the .har file and you will see it is made up of index files and part files. Part files are nothing but the original files concatenated together in to a big file. Index files are look up files which is used to look up the individual small files inside the big part files.   
hadoop fs -ls /output/location/myhar.har   
/output/location/myhar.har/\_index  
/output/location/myhar.har/\_masterindex  
/output/location/myhar.har/part-0  
  
**57) How to copy file from HDFS to the local file system . There is no physical location of a file under the file , not even directory ?**  
**Answer:** bin/hadoop fs -get /hdfs/source/path /localfs/destination/path  
bin/hadoop fs -copyToLocal /hdfs/source/path /localfs/destination/path  
Point your web browser to HDFS WEBUI(namenode\_machine:50070), browse to the file you intend to copy, scroll down the page and click on download the file.  
  
**58) What's the difference between “hadoop fs” shell commands and “hdfs dfs” shell commands? Are they supposed to be equal? but, why the "hadoop fs" commands show the hdfs files while the "hdfs dfs" commands show the local files?**   
**Answer:** Following are the three commands which appears same but have minute differences   
hadoop fs {args}  
hadoop dfs {args}  
hdfs dfs {args}  
hadoop fs {args}  
FS relates to a generic file system which can point to any file systems like local, HDFS etc. So this can be used when you are dealing with different file systems such as Local FS, HFTP FS, S3 FS, and others  
  
hadoop dfs {args}  
dfs is very specific to HDFS. would work for operation relates to HDFS. This has been deprecated and we should use hdfs dfs instead.  
  
hdfs dfs {args}  
same as 2nd i.e would work for all the operations related to HDFS and is the recommended command instead of hadoop dfs  
  
below is the list categorized as HDFS commands.  
  
\*\*#hdfs commands\*\*  
namenode|secondarynamenode|datanode|dfs|dfsadmin|fsck|balancer|fetchdt|oiv|dfsgroups  
So even if you use Hadoop dfs , it will look locate hdfs and delegate that command to hdfs dfs  
  
  
**59)How to check HDFS Directory size?**   
**Answer:** Prior to 0.20.203, and officially deprecated in 2.6.0:   
  
hadoop fs -dus [directory]  
Since 0.20.203 (dead link) 1.0.4 and still compatible through 2.6.0:  
  
hdfs dfs -du [-s] [-h] URI [URI …]  
You can also run hadoop fs -help for more info and specifics.  
  
**60)Why is there no 'hadoop fs -head' shell command?A fast method for inspecting files on HDFS is to use tail:   
  
~$ hadoop fs -tail /path/to/file   
This displays the last kilobyte of data in the file, which is extremely helpful. However, the opposite command head does not appear to be part of the shell command collections. I find this very surprising.  
  
My hypothesis is that since HDFS is built for very fast streaming reads on very large files, there is some access-oriented issue that affects head.**  
**Answer:** I would say it's more to do with efficiency - a head can easily be replicated by piping the output of a hadoop fs -cat through the linux head command.   
  
hadoop fs -cat /path/to/file | head   
This is efficient as head will close out the underlying stream after the desired number of lines have been output   
  
Using tail in this manner would be considerably less efficient - as you'd have to stream over the entire file (all HDFS blocks) to find the final x number of lines.   
hadoop fs -cat /path/to/file | tail   
The hadoop fs -tail command as you note works on the last kilobyte - hadoop can efficiently find the last block and skip to the position of the final kilobyte, then stream the output. Piping via tail can't easily do this

**61) Difference between hadoop fs -put and hadoop fs -copyFromLocal?**  
**Answer:** copyFromLocal is similar to put command, except that the source is restricted to a local file reference.  
So, basically you can do with put, all that you do with copyFromLocal, but not vice-versa.   
Similarly,   
copyToLocal is similar to get command, except that the destination is restricted to a local file reference.   
Hence, you can use get instead of copyToLocal, but not the other way round.  
  
  
**62)Is their any HDFS free space available command? Is there a hdfs command to see available free space in hdfs. We can see that through browser at master:hdfsport in browser , but for some reason I can't access this and I need some command. I can see my disk usage through command ./bin/hadoop fs -du -h but cannot see free space available.**   
**Answer:** hdfs dfsadmin -report   
With older versions of Hadoop, try this:   
hadoop dfsadmin -report   
  
**63)The default data block size of HDFS/hadoop is 64MB. The block size in disk is generally 4KB. What does 64MB block size mean? ->Does it mean that the smallest unit of read from disk is 64MB?   
If yes, what is the advantage of doing that?-> easy for continuous access of large file in HDFS?   
Can we do the same by using the original 4KB block size in disk**   
**Answer:** What does 64MB block size mean?   
The block size is the smallest unit of data that a file system can store. If you store a file that's 1k or 60Mb, it'll take up one block. Once you cross the 64Mb boundry, you need a second block.   
  
If yes, what is the advantage of doing that?   
HDFS is meant to handle large files. Lets say you have a 1000Mb file. With a 4k block size, you'd have to make 256,000 requests to get that file (1 request per block). In HDFS, those requests go across a network and come with a lot of overhead. Each request has to be processed by the Name Node to figure out where that block can be found. That's a lot of traffic! If you use 64Mb blocks, the number of requests goes down to 16, greatly reducing the cost of overhead and load on the Name Node.   
  
**64)How to specify username when putting files on HDFS from a remote machine? I have a Hadoop cluster setup and working under a common default username "user1". I want to put files into hadoop from a remote machine which is not part of the hadoop cluster. I configured hadoop files on the remote machine in a way that when   
  
hadoop dfs -put file1 ...   
is called from the remote machine, it puts the file1 on the Hadoop cluster.   
  
  
the only problem is that I am logged in as "user2" on the remote machine and that doesn't give me the result I expect. In fact, the above code can only be executed on the remote machine as:   
  
  
hadoop dfs -put file1 /user/user2/testFolder   
However, what I really want is to be able to store the file as:   
  
hadoop dfs -put file1 /user/user1/testFolder   
If I try to run the last code, hadoop throws error because of access permissions. Is there anyway that I can specify the username within hadoop dfs command?   
  
I am looking for something like:   
hadoop dfs -username user1 file1 /user/user1/testFolder**   
**Answer:** By default authentication and authorization is turned off in Hadoop.   
The user identity that Hadoop uses for permissions in HDFS is determined by running the whoami command on the client system. Similarly, the group names are derived from the output of running groups.  
So, you can create a new whoami command which returns the required username and put it in the PATH appropriately, so that the created whoami is found before the actual whoami which comes with Linux is found. Similarly, you can play with the groups command also.  
  
This is a hack and won't work once the authentication and authorization has been turned on.  
If you use the HADOOP\_USER\_NAME env variable you can tell HDFS which user name to operate with. Note that this only works if your cluster isn't using security features (e.g. Kerberos). For example:  
  
HADOOP\_USER\_NAME=hdfs hadoop dfs -put ..   
  
**65)Where HDFS stores files locally by default?**   
**Answer:** You need to look in your hdfs-default.xml configuration file for the dfs.data.dir setting. The default setting is: ${hadoop.tmp.dir}/dfs/data and note that the ${hadoop.tmp.dir} is actually in core-default.xml described here.   
The configuration options are described here. The description for this setting is:   
Determines where on the local filesystem an DFS data node should store its blocks. If this is a comma-delimited list of directories, then data will be stored in all named directories, typically on different devices. Directories that do not exist are ignored. 

**66)Hadoop has configuration parameter hadoop.tmp.dir which, as per documentation, is A base for other temporary directories. I presume, this path refers to local file system.   
I set this value to /mnt/hadoop-tmp/hadoop-${user.name}. After formatting the namenode and starting all services, I see exactly same path created on HDFS.   
Does this mean, hadoop.tmp.dir refers to temporary location on HDFS?**   
**Answer:** It's confusing, but hadoop.tmp.dir is used as the base for temporary directories locally, and also in HDFS. The document isn't great, but mapred.system.dir is set by default to "${hadoop.tmp.dir}/mapred/system", and this defines the Path on the HDFS where the Map/Reduce framework stores system files.  
If you want these to not be tied together, you can edit your mapred-site.xml such that the definition of mapred.system.dir is something that's not tied to ${hadoop.tmp.dir}  
  
There're three HDFS properties which contain hadoop.tmp.dir in their values   
dfs.name.dir: directory where namenode stores its metadata, with default value ${hadoop.tmp.dir}/dfs/name.   
dfs.data.dir: directory where HDFS data blocks are stored, with default value ${hadoop.tmp.dir}/dfs/data.   
fs.checkpoint.dir: directory where secondary namenode store its checkpoints, default value is ${hadoop.tmp.dir}/dfs/namesecondary.   
This is why you saw the /mnt/hadoop-tmp/hadoop-${user.name} in your HDFS after formatting namenode.   
  
**67)Is it possible to append to HDFS file from multiple clients in parallel? Basically whole question is in the title. I'm wondering if it's possible to append to file located on HDFS from multiple computers simultaneously? Something like storing stream of events constantly produced by multiple processes. Order is not important.   
I recall hearing on one of the Google tech presentations that GFS supports such append functionality but trying some limited testing with HDFS (either with regular file append() or with SequenceFile) doesn't seems to work.**   
**Answer:** I don't think that this is possible with HDFS. Even though you don't care about the order of the records, you do care about the order of the bytes in the file. You don't want writer A to write a partial record that then gets corrupted by writer B. This is a hard problem for HDFS to solve on its own, so it doesn't.  
Create a file per writer. Pass all the files to any MapReduce worker that needs to read this data. This is much simpler and fits the design of HDFS and Hadoop. If non-MapReduce code needs to read this data as one stream then either stream each file sequentially or write a very quick MapReduce job to consolidate the files.

**68)I have 1000+ files available in HDFS with a naming convention of 1\_fileName.txt to N\_fileName.txt. Size of each file is 1024 MB. I need to merge these files in to one (HDFS)with keeping the order of the file. Say 5\_FileName.txt should append only after 4\_fileName.txt   
What is the best and fastest way to perform this operation.Is there any method to perform this merging without copying the actual data between data nodes? For e-g: Get the block locations of this files and create a new entry (FileName) in the Namenode with these block locations**   
**Answer:** There is no efficient way of doing this, you'll need to move all the data to one node, then back to HDFS.   
A command line scriptlet to do this could be as follows:   
hadoop fs -text \*\_fileName.txt | hadoop fs -put - targetFilename.txt   
This will cat all files that match the glob to standard output, then you'll pipe that stream to the put command and output the stream to an HDFS file named targetFilename.txt   
  
The only problem you have is the filename structure you have gone for - if you have fixed width, zeropadded the number part it would be easier, but in it's current state you'll get an unexpected lexigraphic order (1, 10, 100, 1000, 11, 110, etc) rather than numeric order (1,2,3,4, etc). You could work around this by amending the scriptlet to:   
hadoop fs -text [0-9]\_fileName.txt [0-9][0-9]\_fileName.txt \   
[0-9][0-9[0-9]\_fileName.txt | hadoop fs -put - targetFilename.txt   
  
**69)How to list all files in a directory and its subdirectories in hadoop hdfs?I have a folder in hdfs which has two subfolders each one has about 30 subfolders which,finally,each one contains xml files. I want to list all xml files giving only the main folder's path. Locally I can do this with apache commons-io's FileUtils.listFiles(). I have tried this   
FileStatus[] status = fs.listStatus( new Path( args[ 0 ] ) );   
but it only lists the two first subfolders and it doesn't go further. Is there any way to do this in hadoop?**   
**Answer:** You'll need to use the FileSystem object and perform some logic on the resultant FileStatus objects to manually recurse into the subdirectories.   
You can also apply a PathFilter to only return the xml files using the listStatus(Path, PathFilter) method   
The hadoop FsShell class has examples of this for the hadoop fs -lsr command, which is a recursive ls -   
If you are using hadoop 2.\* API there are more elegant solutions:   
Configuration conf = getConf();   
Job job = Job.getInstance(conf);   
FileSystem fs = FileSystem.get(conf);   
  
//the second boolean parameter here sets the recursion to true   
RemoteIterator (LocatedFileStatus) fileStatusListIterator = fs.listFiles(   
new Path("path/to/lib"), true);   
while(fileStatusListIterator.hasNext()){   
LocatedFileStatus fileStatus = fileStatusListIterator.next();   
//do stuff with the file like …   
job.addFileToClassPath(fileStatus.getPath());   
}   
  
**70)Is there a simple command for hadoop that can change the name of a file (in the HDFS) from its old name to a new name?**   
**Answer:** Use the following : hadoop fs -mv oldname newname   
  
**71)Is there a hdfs command to list files in HDFS directory as per timestamp, ascending or descending? By default, hdfs dfs -ls command gives unsorted list of files.   
When I searched for answers what I got was a workaround i.e. hdfs dfs -ls /tmp | sort -k6,7. But is there any better way, inbuilt in hdfs dfs commandline?**   
**Answer:** No, there is no other option to sort the files based on datetime.   
If you are using hadoop version less than 2.7, you will have to use sort -k6,7 as you are doing:   
  
hdfs dfs -ls /tmp | sort -k6,7   
And for hadoop 2.7.x ls command , there are following options available :   
  
Usage: hadoop fs -ls [-d] [-h] [-R] [-t] [-S] [-r] [-u] [args]   
  
Options:   
-d: Directories are listed as plain files.   
-h: Format file sizes in a human-readable fashion (eg 64.0m instead of 67108864).   
-R: Recursively list subdirectories encountered.   
-t: Sort output by modification time (most recent first).   
-S: Sort output by file size.   
-r: Reverse the sort order.   
-u: Use access time rather than modification time for display and sorting.   
So you can easily sort the files:   
  
hdfs dfs -ls -t -R (-r) /tmp 

**72)How to unzip .gz files in a new directory in hadoop?I have a bunch of .gz files in a folder in hdfs. I want to unzip all of these .gz files to a new folder in hdfs. How should i do this?**   
**Answer:** Using Linux command line   
Following command worked.   
  
hadoop fs -cat /tmp/Links.txt.gz | gzip -d | hadoop fs -put - /tmp/unzipped/Links.txt   
My gzipped file is Links.txt.gz   
The output gets stored in /tmp/unzipped/Links.txt   
  
**73)I'm using hdfs -put to load a large 20GB file into hdfs. Currently the process runs @ 4mins. I'm trying to improve the write time of loading data into hdfs. I tried utilizing different block sizes to improve write speed but got the below results:   
  
512M blocksize = 4mins;   
256M blocksize = 4mins;   
128M blocksize = 4mins;   
64M blocksize = 4mins;   
Does anyone know what the bottleneck could be and other options we could explore to improve performance of the -put cmd?**   
**Answer:** 20GB / 4minute comes out to about 85MB/sec. That's pretty reasonable throughput to expect from a single drive with all the overhead of HDFS protocol and network. I'm betting that is your bottleneck. Without changing your ingest process, you're not going to be able to make this magically faster.   
The core problem is that 20GB is a decent amount of data and that data getting pushed into HDFS as a single stream. You are limited by disk I/O which is pretty lame given you have a large number of disks in a Hadoop cluster.. You've got a while to go to saturate a 10GigE network (and probably a 1GigE, too).   
Changing block size shouldn't change this behavior, as you saw. It's still the same amount of data off disk into HDFS.   
I suggest you split the file up into 1GB files and spread them over multiple disks, then push them up with -put in parallel. You might want even want to consider splitting these files over multiple nodes if network becomes a bottleneck. Can you change the way you receive your data to make this faster? Obvious splitting the file and moving it around will take time, too.