name value description hadoop.hdfs.configuration.version 1 version of this configuration file dfs.namenode.rpc-address RPC address that handles all clients requests. In the case of HA/Federation where multiple namenodes exist, the name service id is added to the name e.g. dfs.namenode.rpc-address.ns1 dfs.namenode.rpc-address.EXAMPLENAMESERVICE The value of this property will take the form of nn-host1:rpc-port. The NameNode's default RPC port is 9820. dfs.namenode.rpc-bind-host The actual address the RPC server will bind to. If this optional address is set, it overrides only the hostname portion of dfs.namenode.rpc-address. It can also be specified per name node or name service for HA/Federation. This is useful for making the name node listen on all interfaces by setting it to 0.0.0.0. dfs.namenode.servicerpc-address RPC address for HDFS Services communication. BackupNode, Datanodes and all other services should be connecting to this address if it is configured. In the case of HA/Federation where multiple namenodes exist, the name service id is added to the name e.g. dfs.namenode.servicerpc-address.ns1 dfs.namenode.rpc-address.EXAMPLENAMESERVICE The value of this property will take the form of nn-host1:rpc-port. If the value of this property is unset the value of dfs.namenode.rpc-address will be used as the default. dfs.namenode.servicerpc-bind-host The actual address the service RPC server will bind to. If this optional address is set, it overrides only the hostname portion of dfs.namenode.servicerpc-address. It can also be specified per name node or name service for HA/Federation. This is useful for making the name node listen on all interfaces by setting it to 0.0.0.0. dfs.namenode.lifeline.rpc-address NameNode RPC lifeline address. This is an optional separate RPC address that can be used to isolate health checks and liveness to protect against resource exhaustion in the main RPC handler pool. In the case of HA/Federation where multiple NameNodes exist, the name service ID is added to the name e.g. dfs.namenode.lifeline.rpc-address.ns1. The value of this property will take the form of nn-host1:rpc-port. If this property is not defined, then the NameNode will not start a lifeline RPC server. By default, the property is not defined. dfs.namenode.lifeline.rpc-bind-host The actual address the lifeline RPC server will bind to. If this optional address is set, it overrides only the hostname portion of dfs.namenode.lifeline.rpc-address. It can also be specified per name node or name service for HA/Federation. This is useful for making the name node listen on all interfaces by setting it to 0.0.0.0. dfs.namenode.secondary.http-address 0.0.0.0:9868 The secondary namenode http server address and port. dfs.namenode.secondary.https-address 0.0.0.0:9869 The secondary namenode HTTPS server address and port. dfs.datanode.address 0.0.0.0:9866 The datanode server address and port for data transfer. dfs.datanode.http.address 0.0.0.0:9864 The datanode http server address and port. dfs.datanode.ipc.address 0.0.0.0:9867 The datanode ipc server address and port. dfs.datanode.handler.count 10 The number of server threads for the datanode. dfs.namenode.http-address 0.0.0.0:9870 The address and the base port where the dfs namenode web ui will listen on. dfs.namenode.http-bind-host The actual address the HTTP server will bind to. If this optional address is set, it overrides only the hostname portion of dfs.namenode.http-address. It can also be specified per name node or name service for HA/Federation. This is useful for making the name node HTTP server listen on all interfaces by setting it to 0.0.0.0. dfs.namenode.heartbeat.recheck-interval 300000 This time decides the interval to check for expired datanodes. With this value and dfs.heartbeat.interval, the interval of deciding the datanode is stale or not is also calculated. The unit of this configuration is millisecond. dfs.http.policy HTTP_ONLY Decide if HTTPS(SSL) is supported on HDFS This configures the HTTP endpoint for HDFS daemons: The following values are supported: - HTTP_ONLY : Service is provided only on http - HTTPS_ONLY : Service is provided only on https - HTTP_AND_HTTPS : Service is provided both on http and https dfs.client.https.need-auth false Whether SSL client certificate authentication is required dfs.client.cached.conn.retry 3 The number of times the HDFS client will pull a socket from the cache. Once this number is exceeded, the client will try to create a new socket. dfs.https.server.keystore.resource ssl-server.xml Resource file from which ssl server keystore information will be extracted dfs.client.https.keystore.resource ssl-client.xml Resource file from which ssl client keystore information will be extracted dfs.datanode.https.address 0.0.0.0:9865 The datanode secure http server address and port. dfs.namenode.https-address 0.0.0.0:9871 The namenode secure http server address and port. dfs.namenode.https-bind-host The actual address the HTTPS server will bind to. If this optional address is set, it overrides only the hostname portion of dfs.namenode.https-address. It can also be specified per name node or name service for HA/Federation. This is useful for making the name node HTTPS server listen on all interfaces by setting it to 0.0.0.0. dfs.datanode.dns.interface default The name of the Network Interface from which a data node should report its IP address. e.g. eth2. This setting may be required for some multi-homed nodes where the DataNodes are assigned multiple hostnames and it is desirable for the DataNodes to

use a non-default hostname. Prefer using hadoop.security.dns.interface over dfs.datanode.dns.interface. dfs.datanode.dns.nameserver default The host name or IP address of the name server (DNS) which a DataNode should use to determine its own host name. Prefer using hadoop.security.dns.nameserver over dfs.datanode.dns.nameserver. dfs.namenode.backup.address 0.0.0.0:50100 The backup node server address and port. If the port is 0 then the server will start on a free port. dfs.namenode.backup.http-address 0.0.0.0:50105 The backup node http server address and port. If the port is 0 then the server will start on a free port. dfs.namenode.redundancy.considerLoad true Decide if chooseTarget considers the target's load or not dfs.namenode.redundancy.considerLoad.factor 2.0 The factor by which a node's load can exceed the average before being rejected for writes, only if considerLoad is true. dfs.default.chunk.view.size 32768 The number of bytes to view for a file on the browser. dfs.datanode.du.reserved 0 Reserved space in bytes per volume. Always leave this much space free for non dfs use. Specific storage type based reservation is also supported. The property can be followed with corresponding storage types ([ssd]/[disk]/[archive]/[ram_disk]) for cluster with heterogeneous storage. For example, reserved space for RAM_DISK storage can be configured using property 'dfs.datanode.du.reserved.ram_disk'. If specific storage type reservation is not configured then dfs.datanode.du.reserved will be used. dfs.namenode.name.dir file://${hadoop.tmp.dir}/dfs/name Determines where on the local filesystem the DFS name node should store the name table(fsimage). If this is a comma-delimited list of directories then the name table is replicated in all of the directories, for redundancy. dfs.namenode.name.dir.restore false Set to true to enable NameNode to attempt recovering a previously failed dfs.namenode.name.dir. When enabled, a recovery of any failed directory is attempted during checkpoint. dfs.namenode.fs-limits.max-component-length 255 Defines the maximum number of bytes in UTF-8 encoding in each component of a path. A value of 0 will disable the check. dfs.namenode.fs-limits.max-directory-items 1048576 Defines the maximum number of items that a directory may contain. Cannot set the property to a value less than 1 or more than 6400000. dfs.namenode.fs-limits.min-block-size 1048576 Minimum block size in bytes, enforced by the Namenode at create time. This prevents the accidental creation of files with tiny block sizes (and thus many blocks), which can degrade performance. dfs.namenode.fs-limits.max-blocks-per-file 10000 Maximum number of blocks per file, enforced by the Namenode on write. This prevents the creation of extremely large files which can degrade performance. dfs.namenode.edits.dir ${dfs.namenode.name.dir} Determines where on the local filesystem the DFS name node should store the transaction (edits) file. If this is a comma-delimited list of directories then the transaction file is replicated in all of the directories, for redundancy. Default value is same as dfs.namenode.name.dir dfs.namenode.edits.dir.required This should be a subset of dfs.namenode.edits.dir, to ensure that the transaction (edits) file in these places is always up-to-date. dfs.namenode.shared.edits.dir A directory on shared storage between the multiple namenodes in an HA cluster. This directory will be written by the active and read by the standby in order to keep the namespaces synchronized. This directory does not need to be listed in dfs.namenode.edits.dir above. It should be left empty in a non-HA cluster. dfs.namenode.edits.journal-plugin.qjournal org.apache.hadoop.hdfs.qjournal.client.QuorumJournalManager dfs.permissions.enabled true If "true", enable permission checking in HDFS. If "false", permission checking is turned off, but all other behavior is unchanged. Switching from one parameter value to the other does not change the mode, owner or group of files or directories. dfs.permissions.superusergroup supergroup The name of the group of super-users. The value should be a single group name. dfs.cluster.administrators ACL for the admins, this configuration is used to control who can access the default servlets in the namenode, etc. The value should be a comma separated list of users and groups. The user list comes first and is separated by a space followed by the group list, e.g. "user1,user2 group1,group2". Both users and groups are optional, so "user1", " group1", "", "user1 group1", "user1,user2 group1,group2" are all valid (note the leading space in " group1"). '*' grants access to all users and groups, e.g. '*', '* ' and ' *' are all valid. dfs.namenode.acls.enabled false Set to true to enable support for HDFS ACLs (Access Control Lists). By default, ACLs are disabled. When ACLs are disabled, the NameNode rejects all RPCs related to setting or getting ACLs. dfs.namenode.posix.acl.inheritance.enabled true Set to true to enable POSIX style ACL inheritance. When it is enabled and the create request comes from a compatible client, the NameNode will apply default ACLs from the parent directory to the create mode and ignore the client umask. If no default ACL found, it will apply the client umask. dfs.namenode.lazypersist.file.scrub.interval.sec 300 The NameNode periodically scans the namespace for LazyPersist files with missing blocks and unlinks them from the namespace. This configuration key controls the interval between successive scans. Set it to a negative value to disable this

behavior. dfs.block.access.token.enable false If "true", access tokens are used as capabilities for accessing datanodes. If "false", no access tokens are checked on accessing datanodes.
dfs.block.access.key.update.interval 600 Interval in minutes at which namenode updates its access keys.
dfs.block.access.token.lifetime 600 The lifetime of access tokens in minutes.
dfs.block.access.token.protobuf.enable false If "true", block tokens are written using Protocol Buffers. If "false", block tokens are written using Legacy format. dfs.datanode.data.dir file://${hadoop.tmp.dir}/dfs/data 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. The directories should be tagged with corresponding storage types ([SSD]/[DISK]/[ARCHIVE]/[RAM_DISK]) for HDFS storage policies. The default storage type will be DISK if the directory does not have a storage type tagged explicitly. Directories that do not exist will be created if local filesystem permission allows. dfs.datanode.data.dir.perm 700 Permissions for the directories on on the local filesystem where the DFS data node store its blocks. The permissions can either be octal or symbolic. dfs.replication 3 Default block replication. The actual number of replications can be specified when the file is created. The default is used if replication is not specified in create time. dfs.replication.max 512 Maximal block replication. dfs.namenode.replication.min 1 Minimal block replication.
dfs.namenode.maintenance.replication.min 1 Minimal live block replication in existence of maintenance mode. dfs.namenode.safemode.replication.min a separate minimum replication factor for calculating safe block count. This is an expert level setting. Setting this lower than the dfs.namenode.replication.min is not recommend and/or dangerous for production setups. When it's not set it takes value from dfs.namenode.replication.min dfs.blocksize 134217728 The default block size for new files, in bytes. You can use the following suffix (case insensitive): k(kilo), m(mega), g(giga), t(tera), p(peta), e(exa) to specify the size (such as 128k, 512m, 1g, etc.), Or provide complete size in bytes (such as 134217728 for 128 MB).
dfs.client.block.write.retries 3 The number of retries for writing blocks to the data nodes, before we signal failure to the application. dfs.client.block.write.replace-datanode-on-failure.enable true If there is a datanode/network failure in the write pipeline, DFSClient will try to remove the failed datanode from the pipeline and then continue writing with the remaining datanodes. As a result, the number of datanodes in the pipeline is decreased. The feature is to add new datanodes to the pipeline. This is a site-wide property to enable/disable the feature. When the cluster size is extremely small, e.g. 3 nodes or less, cluster administrators may want to set the policy to NEVER in the default configuration file or disable this feature. Otherwise, users may experience an unusually high rate of pipeline failures since it is impossible to find new datanodes for replacement. See also dfs.client.block.write.replace-datanode-on-failure.policy
dfs.client.block.write.replace-datanode-on-failure.policy DEFAULT This property is used only if the value of dfs.client.block.write.replace-datanode-on-failure.enable is true. ALWAYS: always add a new datanode when an existing datanode is removed. NEVER: never add a new datanode. DEFAULT: Let r be the replication number. Let n be the number of existing datanodes. Add a new datanode only if r is greater than or equal to 3 and either (1) floor(r/2) is greater than or equal to n; or (2) r is greater than n and the block is hflushed/appended. dfs.client.block.write.replace-datanode-on-failure.best-effort false This property is used only if the value of dfs.client.block.write.replace-datanode-on-failure.enable is true. Best effort means that the client will try to replace a failed datanode in write pipeline (provided that the policy is satisfied), however, it continues the write operation in case that the datanode replacement also fails. Suppose the datanode replacement fails. false: An exception should be thrown so that the write will fail. true : The write should be resumed with the remaining datandoes. Note that setting this property to true allows writing to a pipeline with a smaller number of datanodes. As a result, it increases the probability of data loss.
dfs.client.block.write.replace-datanode-on-failure.min-replication 0 The minimum number of replications that are needed to not to fail the write pipeline if new datanodes can not be found to replace failed datanodes (could be due to network failure) in the write pipeline. If the number of the remaining datanodes in the write pipeline is greater than or equal to this property value, continue writing to the remaining nodes. Otherwise throw exception. If this is set to 0, an exception will be thrown, when a replacement can not be found. See also dfs.client.block.write.replace-datanode-on-failure.policy dfs.blockreport.intervalMsec 21600000 Determines block reporting interval in milliseconds. dfs.blockreport.initialDelay 0s Delay for first block report in seconds. Support multiple time unit suffix(case insensitive), as described in dfs.heartbeat.interval.
dfs.blockreport.split.threshold 1000000 If the number of blocks on the DataNode is below this threshold then it will send block reports for all Storage Directories in a single message. If the number of blocks

exceeds this threshold then the DataNode will send block reports for each Storage Directory in separate messages. Set to zero to always split. dfs.namenode.max.full.block.report.leases 6 The maximum number of leases for full block reports that the NameNode will issue at any given time. This prevents the NameNode from being flooded with full block reports that use up all the RPC handler threads. This number should never be more than the number of RPC handler threads or less than 1.
dfs.namenode.full.block.report.lease.length.ms 300000 The number of milliseconds that the NameNode will wait before invalidating a full block report lease. This prevents a crashed DataNode from permanently using up a full block report lease. dfs.datanode.directoryscan.interval 21600s Interval in seconds for Datanode to scan data directories and reconcile the difference between blocks in memory and on the disk. Support multiple time unit suffix(case insensitive), as described in dfs.heartbeat.interval.
dfs.datanode.directoryscan.threads 1 How many threads should the threadpool used to compile reports for volumes in parallel have. dfs.datanode.directoryscan.throttle.limit.ms.per.sec 1000 The report compilation threads are limited to only running for a given number of milliseconds per second, as configured by the property. The limit is taken per thread, not in aggregate, e.g. setting a limit of 100ms for 4 compiler threads will result in each thread being limited to 100ms, not 25ms. Note that the throttle does not interrupt the report compiler threads, so the actual running time of the threads per second will typically be somewhat higher than the throttle limit, usually by no more than 20%. Setting this limit to 1000 disables compiler thread throttling. Only values between 1 and 1000 are valid. Setting an invalid value will result in the throttle being disabled and an error message being logged. 1000 is the default setting. dfs.heartbeat.interval 3s Determines datanode heartbeat interval in seconds. Can use the following suffix (case insensitive): ms(millis), s(sec), m(min), h(hour), d(day) to specify the time (such as 2s, 2m, 1h, etc.). Or provide complete number in seconds (such as 30 for 30 seconds). dfs.datanode.lifeline.interval.seconds Sets the interval in seconds between sending DataNode Lifeline Protocol messages from the DataNode to the NameNode. The value must be greater than the value of dfs.heartbeat.interval. If this property is not defined, then the default behavior is to calculate the interval as 3x the value of dfs.heartbeat.interval. Note that normal heartbeat processing may cause the DataNode to postpone sending lifeline messages if they are not required. Under normal operations with speedy heartbeat processing, it is possible that no lifeline messages will need to be sent at all. This property has no effect if dfs.namenode.lifeline.rpc-address is not defined.
dfs.namenode.handler.count 10 The number of Namenode RPC server threads that listen to requests from clients. If dfs.namenode.servicerpc-address is not configured then Namenode RPC server threads listen to requests from all nodes. dfs.namenode.service.handler.count 10 The number of Namenode RPC server threads that listen to requests from DataNodes and from all other non-client nodes. dfs.namenode.service.handler.count will be valid only if dfs.namenode.servicerpc-address is configured.
dfs.namenode.lifeline.handler.ratio 0.10 A ratio applied to the value of dfs.namenode.handler.count, which then provides the number of RPC server threads the NameNode runs for handling the lifeline RPC server. For example, if dfs.namenode.handler.count is 100, and dfs.namenode.lifeline.handler.factor is 0.10, then the NameNode starts 100 * 0.10 = 10 threads for handling the lifeline RPC server. It is common to tune the value of dfs.namenode.handler.count as a function of the number of DataNodes in a cluster. Using this property allows for the lifeline RPC server handler threads to be tuned automatically without needing to touch a separate property. Lifeline message processing is lightweight, so it is expected to require many fewer threads than the main NameNode RPC server. This property is not used if dfs.namenode.lifeline.handler.count is defined, which sets an absolute thread count. This property has no effect if dfs.namenode.lifeline.rpc-address is not defined. dfs.namenode.lifeline.handler.count Sets an absolute number of RPC server threads the NameNode runs for handling the DataNode Lifeline Protocol and HA health check requests from ZKFC. If this property is defined, then it overrides the behavior of dfs.namenode.lifeline.handler.ratio. By default, it is not defined. This property has no effect if dfs.namenode.lifeline.rpc-address is not defined. dfs.namenode.safemode.threshold-pct 0.999f Specifies the percentage of blocks that should satisfy the minimal replication requirement defined by dfs.namenode.replication.min. Values less than or equal to 0 mean not to wait for any particular percentage of blocks before exiting safemode. Values greater than 1 will make safe mode permanent.
dfs.namenode.safemode.min.datanodes 0 Specifies the number of datanodes that must be considered alive before the name node exits safemode. Values less than or equal to 0 mean not to take the number of live datanodes into account when deciding whether to remain in safe mode during startup. Values greater than the number of datanodes in the cluster will make safe mode permanent. dfs.namenode.safemode.extension

30000 Determines extension of safe mode in milliseconds after the threshold level is reached. Support multiple time unit suffix (case insensitive), as described in dfs.heartbeat.interval. dfs.namenode.resource.check.interval 5000 The interval in milliseconds at which the NameNode resource checker runs. The checker calculates the number of the NameNode storage volumes whose available spaces are more than dfs.namenode.resource.du.reserved, and enters safemode if the number becomes lower than the minimum value specified by dfs.namenode.resource.checked.volumes.minimum. dfs.namenode.resource.du.reserved 104857600 The amount of space to reserve/require for a NameNode storage directory in bytes. The default is 100MB. dfs.namenode.resource.checked.volumes A list of local directories for the NameNode resource checker to check in addition to the local edits directories. dfs.namenode.resource.checked.volumes.minimum 1 The minimum number of redundant NameNode storage volumes required. dfs.datanode.balance.bandwidthPerSec 10m Specifies the maximum amount of bandwidth that each datanode can utilize for the balancing purpose in term of the number of bytes per second. You can use the following suffix (case insensitive): k(kilo), m(mega), g(giga), t(tera), p(peta), e(exa)to specify the size (such as 128k, 512m, 1g, etc.). Or provide complete size in bytes (such as 134217728 for 128 MB). dfs.hosts Names a file that contains a list of hosts that are permitted to connect to the namenode. The full pathname of the file must be specified. If the value is empty, all hosts are permitted. dfs.hosts.exclude Names a file that contains a list of hosts that are not permitted to connect to the namenode. The full pathname of the file must be specified. If the value is empty, no hosts are excluded. dfs.namenode.max.objects 0 The maximum number of files, directories and blocks dfs supports. A value of zero indicates no limit to the number of objects that dfs supports. dfs.namenode.datanode.registration.ip-hostname-check true If true (the default), then the namenode requires that a connecting datanode's address must be resolved to a hostname. If necessary, a reverse DNS lookup is performed. All attempts to register a datanode from an unresolvable address are rejected. It is recommended that this setting be left on to prevent accidental registration of datanodes listed by hostname in the excludes file during a DNS outage. Only set this to false in environments where there is no infrastructure to support reverse DNS lookup. dfs.namenode.decommission.interval 30s Namenode periodicity in seconds to check if decommission or maintenance is complete. Support multiple time unit suffix(case insensitive), as described in dfs.heartbeat.interval. dfs.namenode.decommission.blocks.per.interval 500000 The approximate number of blocks to process per decommission or maintenance interval, as defined in dfs.namenode.decommission.interval. dfs.namenode.decommission.max.concurrent.tracked.nodes 100 The maximum number of decommission-in-progress or entering-maintenance datanodes nodes that will be tracked at one time by the namenode. Tracking these datanode consumes additional NN memory proportional to the number of blocks on the datnode. Having a conservative limit reduces the potential impact of decommissioning or maintenance of a large number of nodes at once. A value of 0 means no limit will be enforced. dfs.namenode.redundancy.interval.seconds 3s The periodicity in seconds with which the namenode computes low redundancy work for datanodes. Support multiple time unit suffix(case insensitive), as described in dfs.heartbeat.interval. dfs.namenode.accesstime.precision 3600000 The access time for HDFS file is precise upto this value. The default value is 1 hour. Setting a value of 0 disables access times for HDFS. dfs.datanode.plugins Comma-separated list of datanode plug-ins to be activated. dfs.namenode.plugins Comma-separated list of namenode plug-ins to be activated. dfs.namenode.block-placement-policy.default.prefer-local-node true Controls how the default block placement policy places the first replica of a block. When true, it will prefer the node where the client is running. When false, it will prefer a node in the same rack as the client. Setting to false avoids situations where entire copies of large files end up on a single node, thus creating hotspots. dfs.stream-buffer-size 4096 The size of buffer to stream files. The size of this buffer should probably be a multiple of hardware page size (4096 on Intel x86), and it determines how much data is buffered during read and write operations. dfs.bytes-per-checksum 512 The number of bytes per checksum. Must not be larger than dfs.stream-buffer-size dfs.client-write-packet-size 65536 Packet size for clients to write dfs.client.write.exclude.nodes.cache.expiry.interval.millis 600000 The maximum period to keep a DN in the excluded nodes list at a client. After this period, in milliseconds, the previously excluded node(s) will be removed automatically from the cache and will be considered good for block allocations again. Useful to lower or raise in situations where you keep a file open for very long periods (such as a Write-Ahead-Log (WAL) file) to make the writer tolerant to cluster maintenance restarts. Defaults to 10 minutes. dfs.namenode.checkpoint.dir file://${hadoop.tmp.dir}/dfs/namesecondary Determines where on the local filesystem the DFS secondary name node should store the temporary images

to merge. If this is a comma-delimited list of directories then the image is replicated in all of the directories for redundancy. dfs.namenode.checkpoint.edits.dir ${dfs.namenode.checkpoint.dir} Determines where on the local filesystem the DFS secondary name node should store the temporary edits to merge. If this is a comma-delimited list of directories then the edits is replicated in all of the directories for redundancy. Default value is same as dfs.namenode.checkpoint.dir dfs.namenode.checkpoint.period 3600s The number of seconds between two periodic checkpoints. Support multiple time unit suffix(case insensitive), as described in dfs.heartbeat.interval. dfs.namenode.checkpoint.txns 1000000 The Secondary NameNode or CheckpointNode will create a checkpoint of the namespace every 'dfs.namenode.checkpoint.txns' transactions, regardless of whether 'dfs.namenode.checkpoint.period' has expired. dfs.namenode.checkpoint.check.period 60s The SecondaryNameNode and CheckpointNode will poll the NameNode every 'dfs.namenode.checkpoint.check.period' seconds to query the number of uncheckpointed transactions. Support multiple time unit suffix(case insensitive), as described in dfs.heartbeat.interval. dfs.namenode.checkpoint.max-retries 3 The SecondaryNameNode retries failed checkpointing. If the failure occurs while loading fsimage or replaying edits, the number of retries is limited by this variable. dfs.namenode.checkpoint.check.quiet-multiplier 1.5 Used to calculate the amount of time between retries when in the 'quiet' period for creating checkpoints (active namenode already has an up-to-date image from another checkpointer), so we wait a multiplier of the dfs.namenode.checkpoint.check.period before retrying the checkpoint because another node likely is already managing the checkpoints, allowing us to save bandwidth to transfer checkpoints that don't need to be used. dfs.namenode.num.checkpoints.retained 2 The number of image checkpoint files (fsimage_*) that will be retained by the NameNode and Secondary NameNode in their storage directories. All edit logs (stored on edits_* files) necessary to recover an up-to-date namespace from the oldest retained checkpoint will also be retained. dfs.namenode.num.extra.edits.retained 1000000 The number of extra transactions which should be retained beyond what is minimally necessary for a NN restart. It does not translate directly to file's age, or the number of files kept, but to the number of transactions (here "edits" means transactions). One edit file may contain several transactions (edits). During checkpoint, NameNode will identify the total number of edits to retain as extra by checking the latest checkpoint transaction value, subtracted by the value of this property. Then, it scans edits files to identify the older ones that don't include the computed range of retained transactions that are to be kept around, and purges them subsequently. The retainment can be useful for audit purposes or for an HA setup where a remote Standby Node may have been offline for some time and need to have a longer backlog of retained edits in order to start again. Typically each edit is on the order of a few hundred bytes, so the default of 1 million edits should be on the order of hundreds of MBs or low GBs. NOTE: Fewer extra edits may be retained than value specified for this setting if doing so would mean that more segments would be retained than the number configured by dfs.namenode.max.extra.edits.segments.retained. dfs.namenode.max.extra.edits.segments.retained 10000 The maximum number of extra edit log segments which should be retained beyond what is minimally necessary for a NN restart. When used in conjunction with dfs.namenode.num.extra.edits.retained, this configuration property serves to cap the number of extra edits files to a reasonable value. dfs.namenode.delegation.key.update-interval 86400000 The update interval for master key for delegation tokens in the namenode in milliseconds. dfs.namenode.delegation.token.max-lifetime 604800000 The maximum lifetime in milliseconds for which a delegation token is valid. dfs.namenode.delegation.token.renew-interval 86400000 The renewal interval for delegation token in milliseconds. dfs.datanode.failed.volumes.tolerated 0 The number of volumes that are allowed to fail before a datanode stops offering service. By default any volume failure will cause a datanode to shutdown. dfs.image.compress false Should the dfs image be compressed? dfs.image.compression.codec org.apache.hadoop.io.compress.DefaultCodec If the dfs image is compressed, how should they be compressed? This has to be a codec defined in io.compression.codecs. dfs.image.transfer.timeout 60000 Socket timeout for image transfer in milliseconds. This timeout and the related dfs.image.transfer.bandwidthPerSec parameter should be configured such that normal image transfer can complete successfully. This timeout prevents client hangs when the sender fails during image transfer. This is socket timeout during image transfer. dfs.image.transfer.bandwidthPerSec 0 Maximum bandwidth used for regular image transfers (instead of bootstrapping the standby namenode), in bytes per second. This can help keep normal namenode operations responsive during checkpointing. The maximum bandwidth and timeout in dfs.image.transfer.timeout should be set such that normal image transfers can complete

successfully. A default value of 0 indicates that throttling is disabled. The maximum bandwidth used for bootstrapping standby namenode is configured with dfs.image.transfer-bootstrap-standby.bandwidthPerSec. dfs.image.transfer-bootstrap-standby.bandwidthPerSec 0 Maximum bandwidth used for transferring image to bootstrap standby namenode, in bytes per second. A default value of 0 indicates that throttling is disabled. This default value should be used in most cases, to ensure timely HA operations. The maximum bandwidth used for regular image transfers is configured with dfs.image.transfer.bandwidthPerSec. dfs.image.transfer.chunksize 65536 Chunksize in bytes to upload the checkpoint. Chunked streaming is used to avoid internal buffering of contents of image file of huge size. dfs.edit.log.transfer.timeout 30000 Socket timeout for edit log transfer in milliseconds. This timeout should be configured such that normal edit log transfer for journal node syncing can complete successfully. dfs.edit.log.transfer.bandwidthPerSec 0 Maximum bandwidth used for transferring edit log to between journal nodes for syncing, in bytes per second. A default value of 0 indicates that throttling is disabled. dfs.namenode.support.allow.format true Does HDFS namenode allow itself to be formatted? You may consider setting this to false for any production cluster, to avoid any possibility of formatting a running DFS. dfs.datanode.max.transfer.threads 4096 Specifies the maximum number of threads to use for transferring data in and out of the DN. dfs.datanode.scan.period.hours 504 If this is positive, the DataNode will not scan any individual block more than once in the specified scan period. If this is negative, the block scanner is disabled. If this is set to zero, then the default value of 504 hours or 3 weeks is used. Prior versions of HDFS incorrectly documented that setting this key to zero will disable the block scanner. dfs.block.scanner.volume.bytes.per.second 1048576 If this is 0, the DataNode's block scanner will be disabled. If this is positive, this is the number of bytes per second that the DataNode's block scanner will try to scan from each volume. dfs.datanode.readahead.bytes 4194304 While reading block files, if the Hadoop native libraries are available, the datanode can use the posix_fadvise system call to explicitly page data into the operating system buffer cache ahead of the current reader's position. This can improve performance especially when disks are highly contended. This configuration specifies the number of bytes ahead of the current read position which the datanode will attempt to read ahead. This feature may be disabled by configuring this property to 0. If the native libraries are not available, this configuration has no effect. dfs.datanode.drop.cache.behind.reads false In some workloads, the data read from HDFS is known to be significantly large enough that it is unlikely to be useful to cache it in the operating system buffer cache. In this case, the DataNode may be configured to automatically purge all data from the buffer cache after it is delivered to the client. This behavior is automatically disabled for workloads which read only short sections of a block (e.g HBase random-IO workloads). This may improve performance for some workloads by freeing buffer cache space usage for more cacheable data. If the Hadoop native libraries are not available, this configuration has no effect. dfs.datanode.drop.cache.behind.writes false In some workloads, the data written to HDFS is known to be significantly large enough that it is unlikely to be useful to cache it in the operating system buffer cache. In this case, the DataNode may be configured to automatically purge all data from the buffer cache after it is written to disk. This may improve performance for some workloads by freeing buffer cache space usage for more cacheable data. If the Hadoop native libraries are not available, this configuration has no effect. dfs.datanode.sync.behind.writes false If this configuration is enabled, the datanode will instruct the operating system to enqueue all written data to the disk immediately after it is written. This differs from the usual OS policy which may wait for up to 30 seconds before triggering writeback. This may improve performance for some workloads by smoothing the IO profile for data written to disk. If the Hadoop native libraries are not available, this configuration has no effect. dfs.client.failover.max.attempts 15 Expert only. The number of client failover attempts that should be made before the failover is considered failed. dfs.client.failover.sleep.base.millis 500 Expert only. The time to wait, in milliseconds, between failover attempts increases exponentially as a function of the number of attempts made so far, with a random factor of +/- 50%. This option specifies the base value used in the failover calculation. The first failover will retry immediately. The 2nd failover attempt will delay at least dfs.client.failover.sleep.base.millis milliseconds. And so on. dfs.client.failover.sleep.max.millis 15000 Expert only. The time to wait, in milliseconds, between failover attempts increases exponentially as a function of the number of attempts made so far, with a random factor of +/- 50%. This option specifies the maximum value to wait between failovers. Specifically, the time between two failover attempts will not exceed +/- 50% of dfs.client.failover.sleep.max.millis milliseconds. dfs.client.failover.connection.retries 0 Expert only. Indicates the number of retries a failover IPC client will make to establish a server connection. dfs.client.failover.connection.retries.on.timeouts 0 Expert only. The

number of retry attempts a failover IPC client will make on socket timeout when establishing a server connection. dfs.client.datanode-restart.timeout 30s Expert only. The time to wait, in seconds, from reception of an datanode shutdown notification for quick restart, until declaring the datanode dead and invoking the normal recovery mechanisms. The notification is sent by a datanode when it is being shutdown using the shutdownDatanode admin command with the upgrade option. Support multiple time unit suffix(case insensitive), as described in dfs.heartbeat.interval. dfs.nameservices Comma-separated list of nameservices. dfs.nameservice.id The ID of this nameservice. If the nameservice ID is not configured or more than one nameservice is configured for dfs.nameservices it is determined automatically by matching the local node's address with the configured address. dfs.internal.nameservices Comma-separated list of nameservices that belong to this cluster. Datanode will report to all the nameservices in this list. By default this is set to the value of dfs.nameservices. dfs.ha.namenodes.EXAMPLENAMESERVICE The prefix for a given nameservice, contains a comma-separated list of namenodes for a given nameservice (eg EXAMPLENAMESERVICE). Unique identifiers for each NameNode in the nameservice, delimited by commas. This will be used by DataNodes to determine all the NameNodes in the cluster. For example, if you used “mycluster” as the nameservice ID previously, and you wanted to use “nn1” and “nn2” as the individual IDs of the NameNodes, you would configure a property dfs.ha.namenodes.mycluster, and its value "nn1,nn2". dfs.ha.namenode.id The ID of this namenode. If the namenode ID is not configured it is determined automatically by matching the local node's address with the configured address. dfs.ha.log-roll.period 120s How often, in seconds, the StandbyNode should ask the active to roll edit logs. Since the StandbyNode only reads from finalized log segments, the StandbyNode will only be as up-to-date as how often the logs are rolled. Note that failover triggers a log roll so the StandbyNode will be up to date before it becomes active. Support multiple time unit suffix(case insensitive), as described in dfs.heartbeat.interval. dfs.ha.tail-edits.period 60s How often, in seconds, the StandbyNode should check for new finalized log segments in the shared edits log. Support multiple time unit suffix(case insensitive), as described in dfs.heartbeat.interval. dfs.ha.tail-edits.namenode-retries 3 Number of retries to use when contacting the namenode when tailing the log. dfs.ha.tail-edits.rolledits.timeout 60 The timeout in seconds of calling rollEdits RPC on Active NN. dfs.ha.automatic-failover.enabled false Whether automatic failover is enabled. See the HDFS High Availability documentation for details on automatic HA configuration. dfs.client.use.datanode.hostname false Whether clients should use datanode hostnames when connecting to datanodes. dfs.datanode.use.datanode.hostname false Whether datanodes should use datanode hostnames when connecting to other datanodes for data transfer. dfs.client.local.interfaces A comma separated list of network interface names to use for data transfer between the client and datanodes. When creating a connection to read from or write to a datanode, the client chooses one of the specified interfaces at random and binds its socket to the IP of that interface. Individual names may be specified as either an interface name (eg "eth0"), a subinterface name (eg "eth0:0"), or an IP address (which may be specified using CIDR notation to match a range of IPs). dfs.datanode.shared.file.descriptor.paths /dev/shm,/tmp A comma-separated list of paths to use when creating file descriptors that will be shared between the DataNode and the DFSClient. Typically we use /dev/shm, so that the file descriptors will not be written to disk. Systems that don't have /dev/shm will fall back to /tmp by default.
dfs.short.circuit.shared.memory.watcher.interrupt.check.ms 60000 The length of time in milliseconds that the short-circuit shared memory watcher will go between checking for java interruptions sent from other threads. This is provided mainly for unit tests. dfs.namenode.kerberos.principal The NameNode service principal. This is typically set to nn/_HOST@REALM.TLD. Each NameNode will substitute _HOST with its own fully qualified hostname at startup. The _HOST placeholder allows using the same configuration setting on both NameNodes in an HA setup. dfs.namenode.keytab.file The keytab file used by each NameNode daemon to login as its service principal. The principal name is configured with dfs.namenode.kerberos.principal. dfs.datanode.kerberos.principal The DataNode service principal. This is typically set to dn/_HOST@REALM.TLD. Each DataNode will substitute _HOST with its own fully qualified hostname at startup. The _HOST placeholder allows using the same configuration setting on all DataNodes. dfs.datanode.keytab.file The keytab file used by each DataNode daemon to login as its service principal. The principal name is configured with dfs.datanode.kerberos.principal.
dfs.journalnode.kerberos.principal The JournalNode service principal. This is typically set to jn/_HOST@REALM.TLD. Each JournalNode will substitute _HOST with its own fully qualified hostname at startup. The _HOST placeholder allows using the same configuration setting on all JournalNodes.

dfs.journalnode.keytab.file The keytab file used by each JournalNode daemon to login as its service principal. The principal name is configured with dfs.journalnode.kerberos.principal. dfs.namenode.kerberos.internal.spnego.principal ${dfs.web.authentication.kerberos.principal} The server principal used by the NameNode for web UI SPNEGO authentication when Kerberos security is enabled. This is typically set to HTTP/_HOST@REALM.TLD The SPNEGO server principal begins with the prefix HTTP/ by convention. If the value is '*', the web server will attempt to login with every principal specified in the keytab file dfs.web.authentication.kerberos.keytab. dfs.journalnode.kerberos.internal.spnego.principal The server principal used by the JournalNode HTTP Server for SPNEGO authentication when Kerberos security is enabled. This is typically set to HTTP/_HOST@REALM.TLD. The SPNEGO server principal begins with the prefix HTTP/ by convention. If the value is '*', the web server will attempt to login with every principal specified in the keytab file dfs.web.authentication.kerberos.keytab. For most deployments this can be set to ${dfs.web.authentication.kerberos.principal} i.e use the value of dfs.web.authentication.kerberos.principal. dfs.secondary.namenode.kerberos.internal.spnego.principal ${dfs.web.authentication.kerberos.principal} The server principal used by the Secondary NameNode for web UI SPNEGO authentication when Kerberos security is enabled. Like all other Secondary NameNode settings, it is ignored in an HA setup. If the value is '*', the web server will attempt to login with every principal specified in the keytab file dfs.web.authentication.kerberos.keytab. dfs.web.authentication.kerberos.principal The server principal used by the NameNode for WebHDFS SPNEGO authentication. Required when WebHDFS and security are enabled. In most secure clusters this setting is also used to specify the values for dfs.namenode.kerberos.internal.spnego.principal and dfs.journalnode.kerberos.internal.spnego.principal. dfs.web.authentication.kerberos.keytab The keytab file for the principal corresponding to dfs.web.authentication.kerberos.principal. dfs.namenode.kerberos.principal.pattern * A client-side RegEx that can be configured to control allowed realms to authenticate with (useful in cross-realm env.) dfs.namenode.avoid.read.stale.datanode false Indicate whether or not to avoid reading from "stale" datanodes whose heartbeat messages have not been received by the namenode for more than a specified time interval. Stale datanodes will be moved to the end of the node list returned for reading. See dfs.namenode.avoid.write.stale.datanode for a similar setting for writes. dfs.namenode.avoid.write.stale.datanode false Indicate whether or not to avoid writing to "stale" datanodes whose heartbeat messages have not been received by the namenode for more than a specified time interval. Writes will avoid using stale datanodes unless more than a configured ratio (dfs.namenode.write.stale.datanode.ratio) of datanodes are marked as stale. See dfs.namenode.avoid.read.stale.datanode for a similar setting for reads. dfs.namenode.stale.datanode.interval 30000 Default time interval in milliseconds for marking a datanode as "stale", i.e., if the namenode has not received heartbeat msg from a datanode for more than this time interval, the datanode will be marked and treated as "stale" by default. The stale interval cannot be too small since otherwise this may cause too frequent change of stale states. We thus set a minimum stale interval value (the default value is 3 times of heartbeat interval) and guarantee that the stale interval cannot be less than the minimum value. A stale data node is avoided during lease/block recovery. It can be conditionally avoided for reads (see dfs.namenode.avoid.read.stale.datanode) and for writes (see dfs.namenode.avoid.write.stale.datanode). dfs.namenode.write.stale.datanode.ratio 0.5f When the ratio of number stale datanodes to total datanodes marked is greater than this ratio, stop avoiding writing to stale nodes so as to prevent causing hotspots. dfs.namenode.invalidate.work.pct.per.iteration 0.32f *Note*: Advanced property. Change with caution. This determines the percentage amount of block invalidations (deletes) to do over a single DN heartbeat deletion command. The final deletion count is determined by applying this percentage to the number of live nodes in the system. The resultant number is the number of blocks from the deletion list chosen for proper invalidation over a single heartbeat of a single DN. Value should be a positive, non-zero percentage in float notation (X.Yf), with 1.0f meaning 100%. dfs.namenode.replication.work.multiplier.per.iteration 2 *Note*: Advanced property. Change with caution. This determines the total amount of block transfers to begin in parallel at a DN, for replication, when such a command list is being sent over a DN heartbeat by the NN. The actual number is obtained by multiplying this multiplier with the total number of live nodes in the cluster. The result number is the number of blocks to begin transfers immediately for, per DN heartbeat. This number can be any positive, non-zero integer. nfs.server.port 2049 Specify the port number used by Hadoop NFS. nfs.mountd.port 4242 Specify the port number used by Hadoop mount daemon. nfs.dump.dir /tmp/.hdfs-nfs This directory is used to temporarily save out-of-order writes before writing to HDFS. For

each file, the out-of-order writes are dumped after they are accumulated to exceed certain threshold (e.g., 1MB) in memory. One needs to make sure the directory has enough space. nfs.rtmax 1048576 This is the maximum size in bytes of a READ request supported by the NFS gateway. If you change this, make sure you also update the nfs mount's rsize(add rsize= # of bytes to the mount directive). nfs.wtmax 1048576 This is the maximum size in bytes of a WRITE request supported by the NFS gateway. If you change this, make sure you also update the nfs mount's wsize(add wsize= # of bytes to the mount directive). nfs.keytab.file *Note*: Advanced property. Change with caution. This is the path to the keytab file for the hdfs-nfs gateway. This is required when the cluster is kerberized. nfs.kerberos.principal *Note*: Advanced property. Change with caution. This is the name of the kerberos principal. This is required when the cluster is kerberized.It must be of this format: nfs-gateway-user/nfs-gateway-host@kerberos-realm nfs.allow.insecure.ports true When set to false, client connections originating from unprivileged ports (those above 1023) will be rejected. This is to ensure that clients connecting to this NFS Gateway must have had root privilege on the machine where they're connecting from. hadoop.fuse.connection.timeout 300 The minimum number of seconds that we'll cache libhdfs connection objects in fuse_dfs. Lower values will result in lower memory consumption; higher values may speed up access by avoiding the overhead of creating new connection objects. hadoop.fuse.timer.period 5 The number of seconds between cache expiry checks in fuse_dfs. Lower values will result in fuse_dfs noticing changes to Kerberos ticket caches more quickly. dfs.namenode.metrics.logger.period.seconds 600 This setting controls how frequently the NameNode logs its metrics. The logging configuration must also define one or more appenders for NameNodeMetricsLog for the metrics to be logged. NameNode metrics logging is disabled if this value is set to zero or less than zero. dfs.datanode.metrics.logger.period.seconds 600 This setting controls how frequently the DataNode logs its metrics. The logging configuration must also define one or more appenders for DataNodeMetricsLog for the metrics to be logged. DataNode metrics logging is disabled if this value is set to zero or less than zero. dfs.metrics.percentiles.intervals Comma-delimited set of integers denoting the desired rollover intervals (in seconds) for percentile latency metrics on the Namenode and Datanode. By default, percentile latency metrics are disabled. dfs.datanode.peer.stats.enabled false A switch to turn on/off tracking DataNode peer statistics. dfs.datanode.outliers.report.interval 30m This setting controls how frequently DataNodes will report their peer latencies to the NameNode via heartbeats. This setting supports multiple time unit suffixes as described in dfs.heartbeat.interval. If no suffix is specified then milliseconds is assumed. It is ignored if dfs.datanode.peer.stats.enabled is false.
dfs.datanode.fileio.profiling.sampling.percentage 0 This setting controls the percentage of file I/O events which will be profiled for DataNode disk statistics. The default value of 0 disables disk statistics. Set to an integer value between 1 and 100 to enable disk statistics. hadoop.user.group.metrics.percentiles.intervals A comma-separated list of the granularity in seconds for the metrics which describe the 50/75/90/95/99th percentile latency for group resolution in milliseconds. By default, percentile latency metrics are disabled. dfs.encrypt.data.transfer false Whether or not actual block data that is read/written from/to HDFS should be encrypted on the wire. This only needs to be set on the NN and DNs, clients will deduce this automatically. It is possible to override this setting per connection by specifying custom logic via dfs.trustedchannel.resolver.class. dfs.encrypt.data.transfer.algorithm This value may be set to either "3des" or "rc4". If nothing is set, then the configured JCE default on the system is used (usually 3DES.) It is widely believed that 3DES is more cryptographically secure, but RC4 is substantially faster. Note that if AES is supported by both the client and server then this encryption algorithm will only be used to initially transfer keys for AES. (See dfs.encrypt.data.transfer.cipher.suites.) dfs.encrypt.data.transfer.cipher.suites This value may be either undefined or AES/CTR/NoPadding. If defined, then dfs.encrypt.data.transfer uses the specified cipher suite for data encryption. If not defined, then only the algorithm specified in dfs.encrypt.data.transfer.algorithm is used. By default, the property is not defined.
dfs.encrypt.data.transfer.cipher.key.bitlength 128 The key bitlength negotiated by dfsclient and datanode for encryption. This value may be set to either 128, 192 or 256. dfs.trustedchannel.resolver.class TrustedChannelResolver is used to determine whether a channel is trusted for plain data transfer. The TrustedChannelResolver is invoked on both client and server side. If the resolver indicates that the channel is trusted, then the data transfer will not be encrypted even if dfs.encrypt.data.transfer is set to true. The default implementation returns false indicating that the channel is not trusted. dfs.data.transfer.protection A comma-separated list of SASL protection values used for secured connections to the DataNode when reading or writing block data. Possible values are authentication, integrity and privacy. authentication means

authentication only and no integrity or privacy; integrity implies authentication and integrity are enabled; and privacy implies all of authentication, integrity and privacy are enabled. If dfs.encrypt.data.transfer is set to true, then it supersedes the setting for dfs.data.transfer.protection and enforces that all connections must use a specialized encrypted SASL handshake. This property is ignored for connections to a DataNode listening on a privileged port. In this case, it is assumed that the use of a privileged port establishes sufficient trust. dfs.data.transfer.saslproperties.resolver.class SaslPropertiesResolver used to resolve the QOP used for a connection to the DataNode when reading or writing block data. If not specified, the value of hadoop.security.saslproperties.resolver.class is used as the default value. dfs.journalnode.rpc-address 0.0.0.0:8485 The JournalNode RPC server address and port. dfs.journalnode.http-address 0.0.0.0:8480 The address and port the JournalNode HTTP server listens on. If the port is 0 then the server will start on a free port. dfs.journalnode.https-address 0.0.0.0:8481 The address and port the JournalNode HTTPS server listens on. If the port is 0 then the server will start on a free port. dfs.namenode.audit.loggers default List of classes implementing audit loggers that will receive audit events. These should be implementations of org.apache.hadoop.hdfs.server.namenode.AuditLogger. The special value "default" can be used to reference the default audit logger, which uses the configured log system. Installing custom audit loggers may affect the performance and stability of the NameNode. Refer to the custom logger's documentation for more details. dfs.datanode.available-space-volume-choosing-policy.balanced-space-threshold 10737418240 Only used when the dfs.datanode.fsdataset.volume.choosing.policy is set to org.apache.hadoop.hdfs.server.datanode.fsdataset.AvailableSpaceVolumeChoosingPolicy. This setting controls how much DN volumes are allowed to differ in terms of bytes of free disk space before they are considered imbalanced. If the free space of all the volumes are within this range of each other, the volumes will be considered balanced and block assignments will be done on a pure round robin basis. dfs.datanode.available-space-volume-choosing-policy.balanced-space-preference-fraction 0.75f Only used when the dfs.datanode.fsdataset.volume.choosing.policy is set to org.apache.hadoop.hdfs.server.datanode.fsdataset.AvailableSpaceVolumeChoosingPolicy. This setting controls what percentage of new block allocations will be sent to volumes with more available disk space than others. This setting should be in the range 0.0 - 1.0, though in practice 0.5 - 1.0, since there should be no reason to prefer that volumes with less available disk space receive more block allocations. dfs.namenode.edits.noeditlogchannelflush false Specifies whether to flush edit log file channel. When set, expensive FileChannel#force calls are skipped and synchronous disk writes are enabled instead by opening the edit log file with RandomAccessFile("rws") flags. This can significantly improve the performance of edit log writes on the Windows platform. Note that the behavior of the "rws" flags is platform and hardware specific and might not provide the same level of guarantees as FileChannel#force. For example, the write will skip the disk-cache on SAS and SCSI devices while it might not on SATA devices. This is an expert level setting, change with caution. dfs.client.cache.drop.behind.writes Just like dfs.datanode.drop.cache.behind.writes, this setting causes the page cache to be dropped behind HDFS writes, potentially freeing up more memory for other uses. Unlike dfs.datanode.drop.cache.behind.writes, this is a client-side setting rather than a setting for the entire datanode. If present, this setting will override the DataNode default. If the native libraries are not available to the DataNode, this configuration has no effect. dfs.client.cache.drop.behind.reads Just like dfs.datanode.drop.cache.behind.reads, this setting causes the page cache to be dropped behind HDFS reads, potentially freeing up more memory for other uses. Unlike dfs.datanode.drop.cache.behind.reads, this is a client-side setting rather than a setting for the entire datanode. If present, this setting will override the DataNode default. If the native libraries are not available to the DataNode, this configuration has no effect. dfs.client.cache.readahead When using remote reads, this setting causes the datanode to read ahead in the block file using posix_fadvise, potentially decreasing I/O wait times. Unlike dfs.datanode.readahead.bytes, this is a client-side setting rather than a setting for the entire datanode. If present, this setting will override the DataNode default. When using local reads, this setting determines how much readahead we do in BlockReaderLocal. If the native libraries are not available to the DataNode, this configuration has no effect. dfs.client.server-defaults.validity.period.ms 3600000 The amount of milliseconds after which cached server defaults are updated. By default this parameter is set to 1 hour. dfs.namenode.enable.retrycache true This enables the retry cache on the namenode. Namenode tracks for non-idempotent requests the corresponding response. If a client retries the request, the response from the retry cache is sent. Such operations are tagged with annotation @AtMostOnce in namenode protocols. It is recommended that this flag be set to true. Setting it to false, will result in clients getting failure responses to

retried request. This flag must be enabled in HA setup for transparent fail-overs. The entries in the cache have expiration time configurable using dfs.namenode.retrycache.expirytime.millis. dfs.namenode.retrycache.expirytime.millis 600000 The time for which retry cache entries are retained. dfs.namenode.retrycache.heap.percent 0.03f This parameter configures the heap size allocated for retry cache (excluding the response cached). This corresponds to approximately 4096 entries for every 64MB of namenode process java heap size. Assuming retry cache entry expiration time (configured using dfs.namenode.retrycache.expirytime.millis) of 10 minutes, this enables retry cache to support 7 operations per second sustained for 10 minutes. As the heap size is increased, the operation rate linearly increases. dfs.client.mmap.enabled true If this is set to false, the client won't attempt to perform memory-mapped reads. dfs.client.mmap.cache.size 256 When zero-copy reads are used, the DFSClient keeps a cache of recently used memory mapped regions. This parameter controls the maximum number of entries that we will keep in that cache. The larger this number is, the more file descriptors we will potentially use for memory-mapped files. mmaped files also use virtual address space. You may need to increase your ulimit virtual address space limits before increasing the client mmap cache size. Note that you can still do zero-copy reads when this size is set to 0. dfs.client.mmap.cache.timeout.ms 3600000 The minimum length of time that we will keep an mmap entry in the cache between uses. If an entry is in the cache longer than this, and nobody uses it, it will be removed by a background thread. dfs.client.mmap.retry.timeout.ms 300000 The minimum amount of time that we will wait before retrying a failed mmap operation. dfs.client.short.circuit.replica.stale.threshold.ms 1800000 The maximum amount of time that we will consider a short-circuit replica to be valid, if there is no communication from the DataNode. After this time has elapsed, we will re-fetch the short-circuit replica even if it is in the cache. dfs.namenode.path.based.cache.block.map.allocation.percent 0.25 The percentage of the Java heap which we will allocate to the cached blocks map. The cached blocks map is a hash map which uses chained hashing. Smaller maps may be accessed more slowly if the number of cached blocks is large; larger maps will consume more memory. dfs.datanode.max.locked.memory 0 The amount of memory in bytes to use for caching of block replicas in memory on the datanode. The datanode's maximum locked memory soft ulimit (RLIMIT_MEMLOCK) must be set to at least this value, else the datanode will abort on startup. By default, this parameter is set to 0, which disables in-memory caching. If the native libraries are not available to the DataNode, this configuration has no effect. dfs.namenode.list.cache.directives.num.responses 100 This value controls the number of cache directives that the NameNode will send over the wire in response to a listDirectives RPC. dfs.namenode.list.cache.pools.num.responses 100 This value controls the number of cache pools that the NameNode will send over the wire in response to a listPools RPC. dfs.namenode.path.based.cache.refresh.interval.ms 30000 The amount of milliseconds between subsequent path cache rescans. Path cache rescans are when we calculate which blocks should be cached, and on what datanodes. By default, this parameter is set to 30 seconds. dfs.namenode.path.based.cache.retry.interval.ms 30000 When the NameNode needs to uncache something that is cached, or cache something that is not cached, it must direct the DataNodes to do so by sending a DNA_CACHE or DNA_UNCACHE command in response to a DataNode heartbeat. This parameter controls how frequently the NameNode will resend these commands. dfs.datanode.fsdatasetcache.max.threads.per.volume 4 The maximum number of threads per volume to use for caching new data on the datanode. These threads consume both I/O and CPU. This can affect normal datanode operations. dfs.cachereport.intervalMsec 10000 Determines cache reporting interval in milliseconds. After this amount of time, the DataNode sends a full report of its cache state to the NameNode. The NameNode uses the cache report to update its map of cached blocks to DataNode locations. This configuration has no effect if in-memory caching has been disabled by setting dfs.datanode.max.locked.memory to 0 (which is the default). If the native libraries are not available to the DataNode, this configuration has no effect. dfs.namenode.edit.log.autoroll.multiplier.threshold 2.0 Determines when an active namenode will roll its own edit log. The actual threshold (in number of edits) is determined by multiplying this value by dfs.namenode.checkpoint.txns. This prevents extremely large edit files from accumulating on the active namenode, which can cause timeouts during namenode startup and pose an administrative hassle. This behavior is intended as a failsafe for when the standby or secondary namenode fail to roll the edit log by the normal checkpoint threshold. dfs.namenode.edit.log.autoroll.check.interval.ms 300000 How often an active namenode will check if it needs to roll its edit log, in milliseconds. dfs.webhdfs.user.provider.user.pattern ^[A-Za-z_][A-Za-z0-9._-]*[$]?$ Valid pattern for user and group names for webhdfs, it must be a valid java regex.

dfs.webhdfs.acl.provider.permission.pattern ^(default:)?(user|group|mask|other):[[A-Za-z_][A-Za-z0-9._-]]*:([rwx-]{3})?(,(default:)?(user|group|mask|other):[[A-Za-z_][A-Za-z0-9._-]]*:([rwx-]{3})?)*$ Valid pattern for user and group names in webhdfs acl operations, it must be a valid java regex.
dfs.webhdfs.socket.connect-timeout 60s Socket timeout for connecting to WebHDFS servers. This prevents a WebHDFS client from hanging if the server hostname is misconfigured, or the server does not response before the timeout expires. Value is followed by a unit specifier: ns, us, ms, s, m, h, d for nanoseconds, microseconds, milliseconds, seconds, minutes, hours, days respectively. Values should provide units, but milliseconds are assumed. dfs.webhdfs.socket.read-timeout 60s Socket timeout for reading data from WebHDFS servers. This prevents a WebHDFS client from hanging if the server stops sending data. Value is followed by a unit specifier: ns, us, ms, s, m, h, d for nanoseconds, microseconds, milliseconds, seconds, minutes, hours, days respectively. Values should provide units, but milliseconds are assumed.
dfs.client.context default The name of the DFSClient context that we should use. Clients that share a context share a socket cache and short-circuit cache, among other things. You should only change this if you don't want to share with another set of threads. dfs.client.read.shortcircuit false This configuration parameter turns on short-circuit local reads. dfs.client.socket.send.buffer.size 0 Socket send buffer size for a write pipeline in DFSClient side. This may affect TCP connection throughput. If it is set to zero or negative value, no buffer size will be set explicitly, thus enable tcp auto-tuning on some system. The default value is 0.
dfs.domain.socket.path Optional. This is a path to a UNIX domain socket that will be used for communication between the DataNode and local HDFS clients. If the string "_PORT" is present in this path, it will be replaced by the TCP port of the DataNode. dfs.client.read.shortcircuit.skip.checksum false If this configuration parameter is set, short-circuit local reads will skip checksums. This is normally not recommended, but it may be useful for special setups. You might consider using this if you are doing your own checksumming outside of HDFS. dfs.client.read.shortcircuit.streams.cache.size 256 The DFSClient maintains a cache of recently opened file descriptors. This parameter controls the maximum number of file descriptors in the cache. Setting this higher will use more file descriptors, but potentially provide better performance on workloads involving lots of seeks. dfs.client.read.shortcircuit.streams.cache.expiry.ms 300000 This controls the minimum amount of time file descriptors need to sit in the client cache context before they can be closed for being inactive for too long. dfs.datanode.shared.file.descriptor.paths /dev/shm,/tmp Comma separated paths to the directory on which shared memory segments are created. The client and the DataNode exchange information via this shared memory segment. It tries paths in order until creation of shared memory segment succeeds. dfs.namenode.audit.log.debug.cmdlist A comma separated list of NameNode commands that are written to the HDFS namenode audit log only if the audit log level is debug. dfs.client.use.legacy.blockreader.local false Legacy short-circuit reader implementation based on HDFS-2246 is used if this configuration parameter is true. This is for the platforms other than Linux where the new implementation based on HDFS-347 is not available. dfs.block.local-path-access.user Comma separated list of the users allowed to open block files on legacy short-circuit local read.
dfs.client.domain.socket.data.traffic false This control whether we will try to pass normal data traffic over UNIX domain socket rather than over TCP socket on node-local data transfer. This is currently experimental and turned off by default. dfs.namenode.reject-unresolved-dn-topology-mapping false If the value is set to true, then namenode will reject datanode registration if the topology mapping for a datanode is not resolved and NULL is returned (script defined by net.topology.script.file.name fails to execute). Otherwise, datanode will be registered and the default rack will be assigned as the topology path. Topology paths are important for data resiliency, since they define fault domains. Thus it may be unwanted behavior to allow datanode registration with the default rack if the resolving topology failed. dfs.namenode.xattrs.enabled true Whether support for extended attributes is enabled on the NameNode. dfs.namenode.fs-limits.max-xattrs-per-inode 32 Maximum number of extended attributes per inode. dfs.namenode.fs-limits.max-xattr-size 16384 The maximum combined size of the name and value of an extended attribute in bytes. It should be larger than 0, and less than or equal to maximum size hard limit which is 32768. dfs.client.slow.io.warning.threshold.ms 30000 The threshold in milliseconds at which we will log a slow io warning in a dfsclient. By default, this parameter is set to 30000 milliseconds (30 seconds). dfs.datanode.slow.io.warning.threshold.ms 300 The threshold in milliseconds at which we will log a slow io warning in a datanode. By default, this parameter is set to 300 milliseconds. dfs.namenode.lease-recheck-interval-ms 2000 During the release of lease a lock is hold that make any operations on the namenode stuck. In order to not block them during a too long duration we stop releasing lease after this max lock limit. dfs.namenode.max-lock-hold-to-release-lease-ms 25

During the release of lease a lock is hold that make any operations on the namenode stuck. In order to not block them during a too long duration we stop releasing lease after this max lock limit. dfs.namenode.write-lock-reporting-threshold-ms 5000 When a write lock is held on the namenode for a long time, this will be logged as the lock is released. This sets how long the lock must be held for logging to occur. dfs.namenode.read-lock-reporting-threshold-ms 5000 When a read lock is held on the namenode for a long time, this will be logged as the lock is released. This sets how long the lock must be held for logging to occur. dfs.namenode.lock.detailed-metrics.enabled false If true, the namenode will keep track of how long various operations hold the Namesystem lock for and emit this as metrics. These metrics have names of the form FSN(Read|Write)LockNanosOperationName, where OperationName denotes the name of the operation that initiated the lock hold (this will be OTHER for certain uncategorized operations) and they export the hold time values in nanoseconds. dfs.namenode.fslock.fair true If this is true, the FS Namesystem lock will be used in Fair mode, which will help to prevent writer threads from being starved, but can provide lower lock throughput. See java.util.concurrent.locks.ReentrantReadWriteLock for more information on fair/non-fair locks. dfs.namenode.startup.delay.block.deletion.sec 0 The delay in seconds at which we will pause the blocks deletion after Namenode startup. By default it's disabled. In the case a directory has large number of directories and files are deleted, suggested delay is one hour to give the administrator enough time to notice large number of pending deletion blocks and take corrective action.
dfs.datanode.block.id.layout.upgrade.threads 12 The number of threads to use when creating hard links from current to previous blocks during upgrade of a DataNode to block ID-based block layout (see HDFS-6482 for details on the layout). dfs.namenode.list.encryption.zones.num.responses 100 When listing encryption zones, the maximum number of zones that will be returned in a batch. Fetching the list incrementally in batches improves namenode performance. dfs.namenode.list.reencryption.status.num.responses 100 When listing re-encryption status, the maximum number of zones that will be returned in a batch. Fetching the list incrementally in batches improves namenode performance. dfs.namenode.list.openfiles.num.responses 1000 When listing open files, the maximum number of open files that will be returned in a single batch. Fetching the list incrementally in batches improves namenode performance.
dfs.namenode.edekcacheloader.interval.ms 1000 When KeyProvider is configured, the interval time of warming up edek cache on NN starts up / becomes active. All edeks will be loaded from KMS into provider cache. The edek cache loader will try to warm up the cache until succeed or NN leaves active state.
dfs.namenode.edekcacheloader.initial.delay.ms 3000 When KeyProvider is configured, the time delayed until the first attempt to warm up edek cache on NN start up / become active.
dfs.namenode.reencrypt.sleep.interval 1m Interval the re-encrypt EDEK thread sleeps in the main loop. The interval accepts units. If none given, millisecond is assumed. dfs.namenode.reencrypt.batch.size 1000 How many EDEKs should the re-encrypt thread process in one batch.
dfs.namenode.reencrypt.throttle.limit.handler.ratio 1.0 Throttling ratio for the re-encryption, indicating what fraction of time should the re-encrypt handler thread work under NN read lock. Larger than 1.0 values are interpreted as 1.0. Negative value or 0 are invalid values and will fail NN startup.
dfs.namenode.reencrypt.throttle.limit.updater.ratio 1.0 Throttling ratio for the re-encryption, indicating what fraction of time should the re-encrypt updater thread work under NN write lock. Larger than 1.0 values are interpreted as 1.0. Negative value or 0 are invalid values and will fail NN startup.
dfs.namenode.reencrypt.edek.threads 10 Maximum number of re-encrypt threads to contact the KMS and re-encrypt the edeks. dfs.namenode.inotify.max.events.per.rpc 1000 Maximum number of events that will be sent to an inotify client in a single RPC response. The default value attempts to amortize away the overhead for this RPC while avoiding huge memory requirements for the client and NameNode (1000 events should consume no more than 1 MB.) dfs.user.home.dir.prefix /user The directory to prepend to user name to get the user's home direcotry. dfs.datanode.cache.revocation.timeout.ms 900000 When the DFSClient reads from a block file which the DataNode is caching, the DFSClient can skip verifying checksums. The DataNode will keep the block file in cache until the client is done. If the client takes an unusually long time, though, the DataNode may need to evict the block file from the cache anyway. This value controls how long the DataNode will wait for the client to release a replica that it is reading without checksums.
dfs.datanode.cache.revocation.polling.ms 500 How often the DataNode should poll to see if the clients have stopped using a replica that the DataNode wants to uncache. dfs.storage.policy.enabled true Allow users to change the storage policy on files and directories. dfs.namenode.legacy-oiv-image.dir Determines where to save the namespace in the old fsimage format during checkpointing by standby NameNode or

SecondaryNameNode. Users can dump the contents of the old format fsimage by oiv_legacy command. If the value is not specified, old format fsimage will not be saved in checkpoint. dfs.namenode.top.enabled true Enable nntop: reporting top users on namenode dfs.namenode.top.window.num.buckets 10 Number of buckets in the rolling window implementation of nntop dfs.namenode.top.num.users 10 Number of top users returned by the top tool dfs.namenode.top.windows.minutes 1,5,25 comma separated list of nntop reporting periods in minutes dfs.webhdfs.ugi.expire.after.access 600000 How long in milliseconds after the last access the cached UGI will expire. With 0, never expire. dfs.namenode.blocks.per.postponedblocks.rescan 10000 Number of blocks to rescan for each iteration of postponedMisreplicatedBlocks. dfs.datanode.block-pinning.enabled false Whether pin blocks on favored DataNode. dfs.client.block.write.locateFollowingBlock.initial.delay.ms 400 The initial delay (unit is ms) for locateFollowingBlock, the delay time will increase exponentially(double) for each retry. dfs.ha.zkfc.nn.http.timeout.ms 20000 The HTTP connection and read timeout value (unit is ms) when DFS ZKFC tries to get local NN thread dump after local NN becomes SERVICE_NOT_RESPONDING or SERVICE_UNHEALTHY. If it is set to zero, DFS ZKFC won't get local NN thread dump. dfs.ha.tail-edits.in-progress false Whether enable standby namenode to tail in-progress edit logs. Clients might want to turn it on when they want Standby NN to have more up-to-date data. dfs.namenode.ec.system.default.policy RS-6-3-1024k The default erasure coding policy name will be used on the path if no policy name is passed. dfs.namenode.ec.policies.max.cellsize 4194304 The maximum cell size of erasure coding policy. Default is 4MB. dfs.datanode.ec.reconstruction.stripedread.timeout.millis 5000 Datanode striped read timeout in milliseconds. dfs.datanode.ec.reconstruction.stripedread.buffer.size 65536 Datanode striped read buffer size. dfs.datanode.ec.reconstruction.threads 8 Number of threads used by the Datanode for background reconstruction work. dfs.datanode.ec.reconstruction.xmits.weight 0.5 Datanode uses xmits weight to calculate the relative cost of EC recovery tasks comparing to replicated block recovery, of which xmits is always 1. Namenode then uses xmits reported from datanode to throttle recovery tasks for EC and replicated blocks. The xmits of an erasure coding recovery task is calculated as the maximum value between the number of read streams and the number of write streams. dfs.namenode.quota.init-threads 4 The number of concurrent threads to be used in quota initialization. The speed of quota initialization also affects the namenode fail-over latency. If the size of name space is big, try increasing this. dfs.datanode.transfer.socket.send.buffer.size 0 Socket send buffer size for DataXceiver (mirroring packets to downstream in pipeline). This may affect TCP connection throughput. If it is set to zero or negative value, no buffer size will be set explicitly, thus enable tcp auto-tuning on some system. The default value is 0. dfs.datanode.transfer.socket.recv.buffer.size 0 Socket receive buffer size for DataXceiver (receiving packets from client during block writing). This may affect TCP connection throughput. If it is set to zero or negative value, no buffer size will be set explicitly, thus enable tcp auto-tuning on some system. The default value is 0. dfs.namenode.upgrade.domain.factor ${dfs.replication} This is valid only when block placement policy is set to BlockPlacementPolicyWithUpgradeDomain. It defines the number of unique upgrade domains any block's replicas should have. When the number of replicas is less or equal to this value, the policy ensures each replica has an unique upgrade domain. When the number of replicas is greater than this value, the policy ensures the number of unique domains is at least this value. dfs.ha.zkfc.port 8019 RPC port for Zookeeper Failover Controller. dfs.datanode.bp-ready.timeout 20s The maximum wait time for datanode to be ready before failing the received request. Setting this to 0 fails requests right away if the datanode is not yet registered with the namenode. This wait time reduces initial request failures after datanode restart. Support multiple time unit suffix(case insensitive), as described in dfs.heartbeat.interval. dfs.datanode.cached-dfsused.check.interval.ms 600000 The interval check time of loading DU_CACHE_FILE in each volume. When the cluster doing the rolling upgrade operations, it will usually lead dfsUsed cache file of each volume expired and redo the du operations in datanode and that makes datanode start slowly. Adjust this property can make cache file be available for the time as you want. dfs.webhdfs.rest-csrf.enabled false If true, then enables WebHDFS protection against cross-site request forgery (CSRF). The WebHDFS client also uses this property to determine whether or not it needs to send the custom CSRF prevention header in its HTTP requests. dfs.webhdfs.rest-csrf.custom-header X-XSRF-HEADER The name of a custom header that HTTP requests must send when protection against cross-site request forgery (CSRF) is enabled for WebHDFS by setting dfs.webhdfs.rest-csrf.enabled to true. The WebHDFS client also uses this property to determine whether or not it needs to send the custom CSRF prevention header in its HTTP requests. dfs.webhdfs.rest-csrf.methods-to-ignore

GET,OPTIONS,HEAD,TRACE A comma-separated list of HTTP methods that do not require HTTP requests to include a custom header when protection against cross-site request forgery (CSRF) is enabled for WebHDFS by setting dfs.webhdfs.rest-csrf.enabled to true. The WebHDFS client also uses this property to determine whether or not it needs to send the custom CSRF prevention header in its HTTP requests. dfs.webhdfs.rest-csrf.browser-useragents-regex ^Mozilla.*,^Opera.* A comma-separated list of regular expressions used to match against an HTTP request's User-Agent header when protection against cross-site request forgery (CSRF) is enabled for WebHDFS by setting dfs.webhdfs.reset-csrf.enabled to true. If the incoming User-Agent matches any of these regular expressions, then the request is considered to be sent by a browser, and therefore CSRF prevention is enforced. If the request's User-Agent does not match any of these regular expressions, then the request is considered to be sent by something other than a browser, such as scripted automation. In this case, CSRF is not a potential attack vector, so the prevention is not enforced. This helps achieve backwards-compatibility with existing automation that has not been updated to send the CSRF prevention header. dfs.xframe.enabled true If true, then enables protection against clickjacking by returning X_FRAME_OPTIONS header value set to SAMEORIGIN. Clickjacking protection prevents an attacker from using transparent or opaque layers to trick a user into clicking on a button or link on another page. dfs.xframe.value SAMEORIGIN This configration value allows user to specify the value for the X-FRAME-OPTIONS. The possible values for this field are DENY, SAMEORIGIN and ALLOW-FROM. Any other value will throw an exception when namenode and datanodes are starting up.
dfs.balancer.keytab.enabled false Set to true to enable login using a keytab for Kerberized Hadoop.
dfs.balancer.address 0.0.0.0:0 The hostname used for a keytab based Kerberos login. Keytab based login can be enabled with dfs.balancer.keytab.enabled. dfs.balancer.keytab.file The keytab file used by the Balancer to login as its service principal. The principal name is configured with dfs.balancer.kerberos.principal. Keytab based login can be enabled with dfs.balancer.keytab.enabled. dfs.balancer.kerberos.principal The Balancer principal. This is typically set to balancer/_HOST@REALM.TLD. The Balancer will substitute _HOST with its own fully qualified hostname at startup. The _HOST placeholder allows using the same configuration setting on different servers. Keytab based login can be enabled with dfs.balancer.keytab.enabled. dfs.http.client.retry.policy.enabled false If "true", enable the retry policy of WebHDFS client. If "false", retry policy is turned off. Enabling the retry policy can be quite useful while using WebHDFS to copy large files between clusters that could timeout, or copy files between HA clusters that could failover during the copy. dfs.http.client.retry.policy.spec 10000,6,60000,10 Specify a policy of multiple linear random retry for WebHDFS client, e.g. given pairs of number of retries and sleep time (n0, t0), (n1, t1), ..., the first n0 retries sleep t0 milliseconds on average, the following n1 retries sleep t1 milliseconds on average, and so on. dfs.http.client.failover.max.attempts 15 Specify the max number of failover attempts for WebHDFS client in case of network exception. dfs.http.client.retry.max.attempts 10 Specify the max number of retry attempts for WebHDFS client, if the difference between retried attempts and failovered attempts is larger than the max number of retry attempts, there will be no more retries.
dfs.http.client.failover.sleep.base.millis 500 Specify the base amount of time in milliseconds upon which the exponentially increased sleep time between retries or failovers is calculated for WebHDFS client.
dfs.http.client.failover.sleep.max.millis 15000 Specify the upper bound of sleep time in milliseconds between retries or failovers for WebHDFS client. dfs.namenode.hosts.provider.classname org.apache.hadoop.hdfs.server.blockmanagement.HostFileManager The class that provides access for host files. org.apache.hadoop.hdfs.server.blockmanagement.HostFileManager is used by default which loads files specified by dfs.hosts and dfs.hosts.exclude. If org.apache.hadoop.hdfs.server.blockmanagement.CombinedHostFileManager is used, it will load the JSON file defined in dfs.hosts. To change class name, nn restart is required. "dfsadmin -refreshNodes" only refreshes the configuration files used by the class. datanode.https.port 50475 HTTPS port for DataNode.
dfs.balancer.dispatcherThreads 200 Size of the thread pool for the HDFS balancer block mover. dispatchExecutor dfs.balancer.movedWinWidth 5400000 Window of time in ms for the HDFS balancer tracking blocks and its locations. dfs.balancer.moverThreads 1000 Thread pool size for executing block moves. moverThreadAllocator dfs.balancer.max-size-to-move 10737418240 Maximum number of bytes that can be moved by the balancer in a single thread. dfs.balancer.getBlocks.min-block-size 10485760 Minimum block threshold size in bytes to ignore when fetching a source's block list. dfs.balancer.getBlocks.size 2147483648 Total size in bytes of Datanode blocks to get when fetching a source's block list.
dfs.balancer.block-move.timeout 0 Maximum amount of time in milliseconds for a block to move. If this is

set greater than 0, Balancer will stop waiting for a block move completion after this time. In typical clusters, a 3 to 5 minute timeout is reasonable. If timeout happens to a large proportion of block moves, this needs to be increased. It could also be that too much work is dispatched and many nodes are constantly exceeding the bandwidth limit as a result. In that case, other balancer parameters might need to be adjusted. It is disabled (0) by default. dfs.balancer.max-no-move-interval 60000 If this specified amount of time has elapsed and no block has been moved out of a source DataNode, on more effort will be made to move blocks out of this DataNode in the current Balancer iteration. dfs.block.invalidate.limit 1000 The maximum number of invalidate blocks sent by namenode to a datanode per heartbeat deletion command. This property works with "dfs.namenode.invalidate.work.pct.per.iteration" to throttle block deletions.
dfs.block.misreplication.processing.limit 10000 Maximum number of blocks to process for initializing replication queues. dfs.block.placement.ec.classname
org.apache.hadoop.hdfs.server.blockmanagement.BlockPlacementPolicyRackFaultTolerant Placement policy class for striped files. Defaults to BlockPlacementPolicyRackFaultTolerant.class
dfs.block.replicator.classname
org.apache.hadoop.hdfs.server.blockmanagement.BlockPlacementPolicyDefault Class representing block placement policy for non-striped files. There are four block placement policies currently being supported: BlockPlacementPolicyDefault, BlockPlacementPolicyWithNodeGroup, BlockPlacementPolicyRackFaultTolerant and BlockPlacementPolicyWithUpgradeDomain. BlockPlacementPolicyDefault chooses the desired number of targets for placing block replicas in a default way. BlockPlacementPolicyWithNodeGroup places block replicas on environment with node-group layer. BlockPlacementPolicyRackFaultTolerant places the replicas to more racks. BlockPlacementPolicyWithUpgradeDomain places block replicas that honors upgrade domain policy. The details of placing replicas are documented in the javadoc of the corresponding policy classes. The default policy is BlockPlacementPolicyDefault, and the corresponding class is org.apache.hadoop.hdfs.server.blockmanagement.BlockPlacementPolicyDefault.
dfs.blockreport.incremental.intervalMsec 0 If set to a positive integer, the value in ms to wait between sending incremental block reports from the Datanode to the Namenode. dfs.checksum.type CRC32C Checksum type dfs.client.block.write.locateFollowingBlock.retries 5 Number of retries to use when finding the next block during HDFS writes. dfs.client.failover.proxy.provider The prefix (plus a required nameservice ID) for the class name of the configured Failover proxy provider for the host. For more detailed information, please consult the "Configuration Details" section of the HDFS High Availability documentation. dfs.client.key.provider.cache.expiry 864000000 DFS client security key cache expiration in milliseconds. dfs.client.max.block.acquire.failures 3 Maximum failures allowed when trying to get block information from a specific datanode. dfs.client.read.prefetch.size The number of bytes for the DFSClient will fetch from the Namenode during a read operation. Defaults to 10 * ${dfs.blocksize}.
dfs.client.read.short.circuit.replica.stale.threshold.ms 1800000 Threshold in milliseconds for read entries during short-circuit local reads. dfs.client.read.shortcircuit.buffer.size 1048576 Buffer size in bytes for short-circuit local reads. dfs.client.read.striped.threadpool.size 18 The maximum number of threads used for parallel reading in striped layout. dfs.client.replica.accessor.builder.classes Comma-separated classes for building ReplicaAccessor. If the classes are specified, client will use external BlockReader that uses the ReplicaAccessor built by the builder. dfs.client.retry.interval-ms.get-last-block-length 4000 Retry interval in milliseconds to wait between retries in getting block lengths from the datanodes.
dfs.client.retry.max.attempts 10 Max retry attempts for DFSClient talking to namenodes.
dfs.client.retry.policy.enabled false If true, turns on DFSClient retry policy. dfs.client.retry.policy.spec 10000,6,60000,10 Set to pairs of timeouts and retries for DFSClient. dfs.client.retry.times.get-last-block-length 3 Number of retries for calls to fetchLocatedBlocksAndGetLastBlockLength().
dfs.client.retry.window.base 3000 Base time window in ms for DFSClient retries. For each retry attempt, this value is extended linearly (e.g. 3000 ms for first attempt and first retry, 6000 ms for second retry, 9000 ms for third retry, etc.). dfs.client.socket-timeout 60000 Default timeout value in milliseconds for all sockets.
dfs.client.socketcache.capacity 16 Socket cache capacity (in entries) for short-circuit reads.
dfs.client.socketcache.expiryMsec 3000 Socket cache expiration for short-circuit reads in msec.
dfs.client.test.drop.namenode.response.number 0 The number of Namenode responses dropped by DFSClient for each RPC call. Used for testing the NN retry cache. dfs.client.hedged.read.threadpool.size 0 Support 'hedged' reads in DFSClient. To enable this feature, set the parameter to a positive number. The

threadpool size is how many threads to dedicate to the running of these 'hedged', concurrent reads in your client. dfs.client.hedged.read.threshold.millis 500 Configure 'hedged' reads in DFSClient. This is the number of milliseconds to wait before starting up a 'hedged' read. dfs.client.write.byte-array-manager.count-limit 2048 The maximum number of arrays allowed for each array length. dfs.client.write.byte-array-manager.count-reset-time-period-ms 10000 The time period in milliseconds that the allocation count for each array length is reset to zero if there is no increment. dfs.client.write.byte-array-manager.count-threshold 128 The count threshold for each array length so that a manager is created only after the allocation count exceeds the threshold. In other words, the particular array length is not managed until the allocation count exceeds the threshold. dfs.client.write.byte-array-manager.enabled false If true, enables byte array manager used by DFSOutputStream. dfs.client.write.max-packets-in-flight 80 The maximum number of DFSPackets allowed in flight. dfs.content-summary.limit 5000 The maximum content summary counts allowed in one locking period. 0 or a negative number means no limit (i.e. no yielding). dfs.content-summary.sleep-microsec 500 The length of time in microseconds to put the thread to sleep, between reaquiring the locks in content summary computation. dfs.data.transfer.client.tcpnodelay true If true, set TCP_NODELAY to sockets for transferring data from DFS client. dfs.data.transfer.server.tcpnodelay true If true, set TCP_NODELAY to sockets for transferring data between Datanodes.
dfs.datanode.balance.max.concurrent.moves 50 Maximum number of threads for Datanode balancer pending moves. This value is reconfigurable via the "dfsadmin -reconfig" command. dfs.datanode.fsdataset.factory The class name for the underlying storage that stores replicas for a Datanode. Defaults to org.apache.hadoop.hdfs.server.datanode.fsdataset.impl.FsDatasetFactory.
dfs.datanode.fsdataset.volume.choosing.policy The class name of the policy for choosing volumes in the list of directories. Defaults to org.apache.hadoop.hdfs.server.datanode.fsdataset.RoundRobinVolumeChoosingPolicy. If you would like to take into account available disk space, set the value to "org.apache.hadoop.hdfs.server.datanode.fsdataset.AvailableSpaceVolumeChoosingPolicy".
dfs.datanode.hostname Optional. The hostname for the Datanode containing this configuration file. Will be different for each machine. Defaults to current hostname. dfs.datanode.lazywriter.interval.sec 60 Interval in seconds for Datanodes for lazy persist writes. dfs.datanode.network.counts.cache.max.size 2147483647 The maximum number of entries the datanode per-host network error count cache may contain.
dfs.datanode.oob.timeout-ms 1500,0,0,0 Timeout value when sending OOB response for each OOB type, which are OOB_RESTART, OOB_RESERVED1, OOB_RESERVED2, and OOB_RESERVED3, respectively. Currently, only OOB_RESTART is used. dfs.datanode.parallel.volumes.load.threads.num Maximum number of threads to use for upgrading data directories. The default value is the number of storage directories in the DataNode. dfs.datanode.ram.disk.replica.tracker Name of the class implementing the RamDiskReplicaTracker interface. Defaults to org.apache.hadoop.hdfs.server.datanode.fsdataset.impl.RamDiskReplicaLruTracker.
dfs.datanode.restart.replica.expiration 50 During shutdown for restart, the amount of time in seconds budgeted for datanode restart. dfs.datanode.socket.reuse.keepalive 4000 The window of time in ms before the DataXceiver closes a socket for a single request. If a second request occurs within that window, the socket can be reused. dfs.datanode.socket.write.timeout 480000 Timeout in ms for clients socket writes to DataNodes. dfs.datanode.sync.behind.writes.in.background false If set to true, then sync_file_range() system call will occur asynchronously. This property is only valid when the property dfs.datanode.sync.behind.writes is true. dfs.datanode.transferTo.allowed true If false, break block transfers on 32-bit machines greater than or equal to 2GB into smaller chunks. dfs.ha.fencing.methods A list of scripts or Java classes which will be used to fence the Active NameNode during a failover. See the HDFS High Availability documentation for details on automatic HA configuration. dfs.ha.standby.checkpoints true If true, a NameNode in Standby state periodically takes a checkpoint of the namespace, saves it to its local storage and then upload to the remote NameNode. dfs.ha.zkfc.port 8019 The port number that the zookeeper failover controller RPC server binds to. dfs.journalnode.edits.dir /tmp/hadoop/dfs/journalnode/ The directory where the journal edit files are stored. dfs.journalnode.enable.sync false If true, the journal nodes wil sync with each other. The journal nodes will periodically gossip with other journal nodes to compare edit log manifests and if they detect any missing log segment, they will download it from the other journal nodes.
dfs.journalnode.sync.interval 120000 Time interval, in milliseconds, between two Journal Node syncs. This configuration takes effect only if the journalnode sync is enabled by setting the configuration parameter

dfs.journalnode.enable.sync to true. dfs.journalnode.kerberos.internal.spnego.principal Kerberos SPNEGO principal name used by the journal node. dfs.journalnode.kerberos.principal Kerberos principal name for the journal node. dfs.journalnode.keytab.file Kerberos keytab file for the journal node. dfs.ls.limit 1000 Limit the number of files printed by ls. If less or equal to zero, at most DFS_LIST_LIMIT_DEFAULT (= 1000) will be printed. dfs.mover.movedWinWidth 5400000 The minimum time interval, in milliseconds, that a block can be moved to another location again. dfs.mover.moverThreads 1000 Configure the balancer's mover thread pool size. dfs.mover.retry.max.attempts 10 The maximum number of retries before the mover consider the move failed. dfs.mover.keytab.enabled false Set to true to enable login using a keytab for Kerberized Hadoop. dfs.mover.address 0.0.0.0:0 The hostname used for a keytab based Kerberos login. Keytab based login can be enabled with dfs.mover.keytab.enabled. dfs.mover.keytab.file The keytab file used by the Mover to login as its service principal. The principal name is configured with dfs.mover.kerberos.principal. Keytab based login can be enabled with dfs.mover.keytab.enabled. dfs.mover.kerberos.principal The Mover principal. This is typically set to mover/_HOST@REALM.TLD. The Mover will substitute _HOST with its own fully qualified hostname at startup. The _HOST placeholder allows using the same configuration setting on different servers. Keytab based login can be enabled with dfs.mover.keytab.enabled. dfs.mover.max-no-move-interval 60000 If this specified amount of time has elapsed and no block has been moved out of a source DataNode, on more effort will be made to move blocks out of this DataNode in the current Mover iteration. dfs.namenode.audit.log.async false If true, enables asynchronous audit log. dfs.namenode.audit.log.token.tracking.id false If true, adds a tracking ID for all audit log events. dfs.namenode.available-space-block-placement-policy.balanced-space-preference-fraction 0.6 Only used when the dfs.block.replicator.classname is set to org.apache.hadoop.hdfs.server.blockmanagement.AvailableSpaceBlockPlacementPolicy. Special value between 0 and 1, noninclusive. Increases chance of placing blocks on Datanodes with less disk space used. dfs.namenode.backup.dnrpc-address Service RPC address for the backup Namenode. dfs.namenode.delegation.token.always-use false For testing. Setting to true always allows the DT secret manager to be used, even if security is disabled. dfs.namenode.edits.asynclogging true If set to true, enables asynchronous edit logs in the Namenode. If set to false, the Namenode uses the traditional synchronous edit logs. dfs.namenode.edits.dir.minimum 1 dfs.namenode.edits.dir includes both required directories (specified by dfs.namenode.edits.dir.required) and optional directories. The number of usable optional directories must be greater than or equal to this property. If the number of usable optional directories falls below dfs.namenode.edits.dir.minimum, HDFS will issue an error. This property defaults to 1. dfs.namenode.edits.journal-plugin When FSEditLog is creating JournalManagers from dfs.namenode.edits.dir, and it encounters a URI with a schema different to "file" it loads the name of the implementing class from "dfs.namenode.edits.journal-plugin.[schema]". This class must implement JournalManager and have a constructor which takes (Configuration, URI). dfs.namenode.file.close.num-committed-allowed 0 Normally a file can only be closed with all its blocks are committed. When this value is set to a positive integer N, a file can be closed when N blocks are committed and the rest complete. dfs.namenode.inode.attributes.provider.class Name of class to use for delegating HDFS authorization. dfs.namenode.inode.attributes.provider.bypass.users A list of user principals (in secure cluster) or user names (in insecure cluster) for whom the external attributes provider will be bypassed for all operations. This means file attributes stored in HDFS instead of the external provider will be used for permission checking and be returned when requested. dfs.namenode.max-num-blocks-to-log 1000 Puts a limit on the number of blocks printed to the log by the Namenode after a block report. dfs.namenode.max.op.size 52428800 Maximum opcode size in bytes. dfs.namenode.missing.checkpoint.periods.before.shutdown 3 The number of checkpoint period windows (as defined by the property dfs.namenode.checkpoint.period) allowed by the Namenode to perform saving the namespace before shutdown. dfs.namenode.name.cache.threshold 10 Frequently accessed files that are accessed more times than this threshold are cached in the FSDirectory nameCache. dfs.namenode.replication.max-streams 2 Hard limit for the number of highest-priority replication streams. dfs.namenode.replication.max-streams-hard-limit 4 Hard limit for all replication streams. dfs.namenode.reconstruction.pending.timeout-sec 300 Timeout in seconds for block reconstruction. If this value is 0 or less, then it will default to 5 minutes. dfs.namenode.stale.datanode.minimum.interval 3 Minimum number of missed heartbeats intervals for a datanode to be marked stale by the Namenode. The actual interval is calculated as (dfs.namenode.stale.datanode.minimum.interval * dfs.heartbeat.interval) in seconds. If this value is greater than the property dfs.namenode.stale.datanode.interval, then the calculated

value above is used. dfs.namenode.storageinfo.defragment.timeout.ms 4 Timeout value in ms for the StorageInfo compaction run. dfs.namenode.storageinfo.defragment.interval.ms 600000 The thread for checking the StorageInfo for defragmentation will run periodically. The time between runs is determined by this property. dfs.namenode.storageinfo.defragment.ratio 0.75 The defragmentation threshold for the StorageInfo. dfs.namenode.snapshot.capture.openfiles false If true, snapshots taken will have an immutable shared copy of the open files that have valid leases. Even after the open files grow or shrink in size, snapshot will always have the previous point-in-time version of the open files, just like all other closed files. Default is false. Note: The file length captured for open files in snapshot is whats recorded in NameNode at the time of snapshot and it may be shorter than what the client has written till then. In order to capture the latest length, the client can call hflush/hsync with the flag SyncFlag.UPDATE_LENGTH on the open files handles. dfs.namenode.snapshot.skip.capture.accesstime-only-change false If accessTime of a file/directory changed but there is no other modification made to the file/directory, the changed accesstime will not be captured in next snapshot. However, if there is other modification made to the file/directory, the latest access time will be captured together with the modification in next snapshot. dfs.namenode.snapshotdiff.allow.snap-root-descendant true If enabled, snapshotDiff command can be run for any descendant directory under a snapshot root directory and the diff calculation will be scoped to the given descendant directory. Otherwise, snapshot diff command can only be run for a snapshot root directory. dfs.pipeline.ecn false If true, allows ECN (explicit congestion notification) from the Datanode. dfs.qjournal.accept-recovery.timeout.ms 120000 Quorum timeout in milliseconds during accept phase of recovery/synchronization for a specific segment. dfs.qjournal.finalize-segment.timeout.ms 120000 Quorum timeout in milliseconds during finalizing for a specific segment. dfs.qjournal.get-journal-state.timeout.ms 120000 Timeout in milliseconds when calling getJournalState(). JournalNodes. dfs.qjournal.new-epoch.timeout.ms 120000 Timeout in milliseconds when getting an epoch number for write access to JournalNodes. dfs.qjournal.prepare-recovery.timeout.ms 120000 Quorum timeout in milliseconds during preparation phase of recovery/synchronization for a specific segment. dfs.qjournal.queued-edits.limit.mb 10 Queue size in MB for quorum journal edits. dfs.qjournal.select-input-streams.timeout.ms 20000 Timeout in milliseconds for accepting streams from JournalManagers. dfs.qjournal.start-segment.timeout.ms 20000 Quorum timeout in milliseconds for starting a log segment. dfs.qjournal.write-txns.timeout.ms 20000 Write timeout in milliseconds when writing to a quorum of remote journals. dfs.quota.by.storage.type.enabled true If true, enables quotas based on storage type. dfs.secondary.namenode.kerberos.principal Kerberos principal name for the Secondary NameNode. dfs.secondary.namenode.keytab.file Kerberos keytab file for the Secondary NameNode. dfs.web.authentication.filter org.apache.hadoop.hdfs.web.AuthFilter Authentication filter class used for WebHDFS. dfs.web.authentication.simple.anonymous.allowed If true, allow anonymous user to access WebHDFS. Set to false to disable anonymous authentication. dfs.web.ugi dfs.web.ugi is deprecated. Use hadoop.http.staticuser.user instead. dfs.webhdfs.netty.high.watermark 65535 High watermark configuration to Netty for Datanode WebHdfs. dfs.webhdfs.netty.low.watermark 32768 Low watermark configuration to Netty for Datanode WebHdfs. dfs.webhdfs.oauth2.access.token.provider Access token provider class for WebHDFS using OAuth2. Defaults to org.apache.hadoop.hdfs.web.oauth2.ConfCredentialBasedAccessTokenProvider. dfs.webhdfs.oauth2.client.id Client id used to obtain access token with either credential or refresh token. dfs.webhdfs.oauth2.enabled false If true, enables OAuth2 in WebHDFS dfs.webhdfs.oauth2.refresh.url URL against which to post for obtaining bearer token with either credential or refresh token. ssl.server.keystore.keypassword Keystore key password for HTTPS SSL configuration ssl.server.keystore.location Keystore location for HTTPS SSL configuration ssl.server.keystore.password Keystore password for HTTPS SSL configuration ssl.server.truststore.location Truststore location for HTTPS SSL configuration ssl.server.truststore.password Truststore password for HTTPS SSL configuration dfs.disk.balancer.max.disk.throughputInMBperSec 10 Maximum disk bandwidth used by diskbalancer during read from a source disk. The unit is MB/sec. dfs.disk.balancer.block.tolerance.percent 10 When a disk balancer copy operation is proceeding, the datanode is still active. So it might not be possible to move the exactly specified amount of data. So tolerance allows us to define a percentage which defines a good enough move. dfs.disk.balancer.max.disk.errors 5 During a block move from a source to destination disk, we might encounter various errors. This defines how many errors we can tolerate before we declare a move between 2 disks (or a step) has failed. dfs.disk.balancer.enabled false This enables the diskbalancer feature on a cluster. By default, disk balancer is disabled. dfs.disk.balancer.plan.threshold.percent 10 The percentage

threshold value for volume Data Density in a plan. If the absolute value of volume Data Density which is out of threshold value in a node, it means that the volumes corresponding to the disks should do the balancing in the plan. The default value is 10. dfs.lock.suppress.warning.interval 10s Instrumentation reporting long critical sections will suppress consecutive warnings within this interval. httpfs.buffer.size 4096 The size buffer to be used when creating or opening httpfs filesystem IO stream. dfs.webhdfs.use.ipc.callq true Enables routing of webhdfs calls through rpc call queue dfs.datanode.disk.check.min.gap 15m The minimum gap between two successive checks of the same DataNode volume. This setting supports multiple time unit suffixes as described in dfs.heartbeat.interval. If no suffix is specified then milliseconds is assumed. dfs.datanode.disk.check.timeout 10m Maximum allowed time for a disk check to complete during DataNode startup. If the check does not complete within this time interval then the disk is declared as failed. This setting supports multiple time unit suffixes as described in dfs.heartbeat.interval. If no suffix is specified then milliseconds is assumed. dfs.use.dfs.network.topology true Enables DFSNetworkTopology to choose nodes for placing replicas. dfs.qjm.operations.timeout 60s Common key to set timeout for related operations in QuorumJournalManager. This setting supports multiple time unit suffixes as described in dfs.heartbeat.interval. If no suffix is specified then milliseconds is assumed. dfs.reformat.disabled false Disable reformat of NameNode. If it's value is set to "true" and metadata directories already exist then attempt to format NameNode will throw NameNodeFormatException.
dfs.federation.router.default.nameserviceId Nameservice identifier of the default subcluster to monitor.
dfs.federation.router.rpc.enable true If true, the RPC service to handle client requests in the router is enabled.
dfs.federation.router.rpc-address 0.0.0.0:8888 RPC address that handles all clients requests. The value of this property will take the form of router-host1:rpc-port. dfs.federation.router.rpc-bind-host The actual address the RPC server will bind to. If this optional address is set, it overrides only the hostname portion of dfs.federation.router.rpc-address. This is useful for making the name node listen on all interfaces by setting it to 0.0.0.0. dfs.federation.router.handler.count 10 The number of server threads for the router to handle RPC requests from clients. dfs.federation.router.handler.queue.size 100 The size of the queue for the number of handlers to handle RPC client requests. dfs.federation.router.reader.count 1 The number of readers for the router to handle RPC client requests. dfs.federation.router.reader.queue.size 100 The size of the queue for the number of readers for the router to handle RPC client requests. dfs.federation.router.connection.pool-size 1 Size of the pool of connections from the router to namenodes. dfs.federation.router.connection.clean.ms 10000 Time interval, in milliseconds, to check if the connection pool should remove unused connections. dfs.federation.router.connection.pool.clean.ms 60000 Time interval, in milliseconds, to check if the connection manager should remove unused connection pools. dfs.federation.router.metrics.enable true If the metrics in the router are enabled. dfs.federation.router.metrics.class org.apache.hadoop.hdfs.server.federation.metrics.FederationRPCPerformanceMonitor Class to monitor the RPC system in the router. It must implement the RouterRpcMonitor interface.
dfs.federation.router.admin.enable true If true, the RPC admin service to handle client requests in the router is enabled. dfs.federation.router.admin-address 0.0.0.0:8111 RPC address that handles the admin requests. The value of this property will take the form of router-host1:rpc-port. dfs.federation.router.admin-bind-host The actual address the RPC admin server will bind to. If this optional address is set, it overrides only the hostname portion of dfs.federation.router.admin-address. This is useful for making the name node listen on all interfaces by setting it to 0.0.0.0. dfs.federation.router.admin.handler.count 1 The number of server threads for the router to handle RPC requests from admin. dfs.federation.router.http-address 0.0.0.0:50071 HTTP address that handles the web requests to the Router. The value of this property will take the form of router-host1:http-port. dfs.federation.router.http-bind-host The actual address the HTTP server will bind to. If this optional address is set, it overrides only the hostname portion of dfs.federation.router.http-address. This is useful for making the name node listen on all interfaces by setting it to 0.0.0.0.
dfs.federation.router.https-address 0.0.0.0:50072 HTTPS address that handles the web requests to the Router. The value of this property will take the form of router-host1:https-port. dfs.federation.router.https-bind-host The actual address the HTTPS server will bind to. If this optional address is set, it overrides only the hostname portion of dfs.federation.router.https-address. This is useful for making the name node listen on all interfaces by setting it to 0.0.0.0. dfs.federation.router.http.enable true If the HTTP service to handle client requests in the router is enabled. dfs.federation.router.metrics.enable true If the metrics service in the router is enabled. dfs.federation.router.file.resolver.client.class org.apache.hadoop.hdfs.server.federation.MockResolver Class to resolve files to subclusters.

dfs.federation.router.namenode.resolver.client.class org.apache.hadoop.hdfs.server.federation.resolver.MembershipNamenodeResolver Class to resolve the namenode for a subcluster. dfs.federation.router.store.enable true If true, the Router connects to the State Store. dfs.federation.router.store.serializer org.apache.hadoop.hdfs.server.federation.store.driver.impl.StateStoreSerializerPBImpl Class to serialize State Store records. dfs.federation.router.store.driver.class org.apache.hadoop.hdfs.server.federation.store.driver.impl.StateStoreFileImpl Class to implement the State Store. By default it uses the local disk. dfs.federation.router.store.connection.test 60000 How often to check for the connection to the State Store in milliseconds. dfs.federation.router.cache.ttl 60000 How often to refresh the State Store caches in milliseconds. dfs.federation.router.store.membership.expiration 300000 Expiration time in milliseconds for a membership record. dfs.federation.router.heartbeat.enable true If true, the Router heartbeats into the State Store. dfs.federation.router.heartbeat.interval 5000 How often the Router should heartbeat into the State Store in milliseconds. dfs.federation.router.monitor.namenode The identifier of the namenodes to monitor and heartbeat. dfs.federation.router.monitor.localnamenode.enable true If true, the Router should monitor the namenode in the local machine. dfs.reformat.disabled false Disable reformat of NameNode. If it's value is set to "true" and metadata directories already exist then attempt to format NameNode will throw NameNodeFormatException.