University of South Australia

HDFS in detail

1

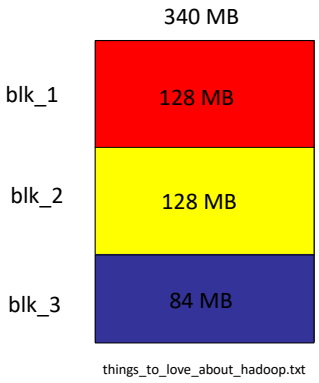
Block size

128 MB vs 64 MB?

What are the pros and cons of having small or large blocks?

Too large – processing time increases for each block

Too small – Too many data blocks, lots of metadata, seeking blocks takes time



University of South Australia

2

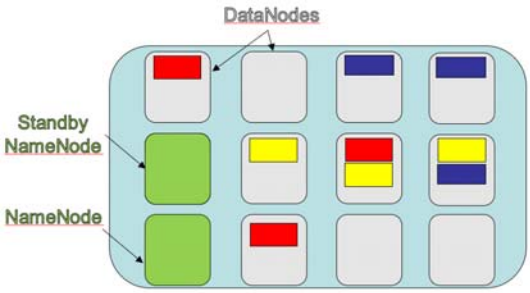
Replication factor

Hadoop's default replication factor is 3

What are the pros and cons of having small or large replication factors?

Larger– Safer, requires more storage, almost always unnecessary.

Smaller– Faster, requires less space, but dangerous

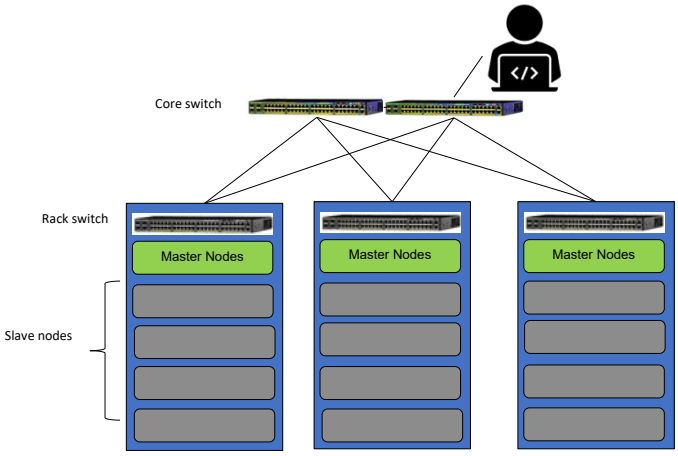


3

Data centre racks

A 'rack' in a data centre is a physical framework used to store a set of computing equipment.

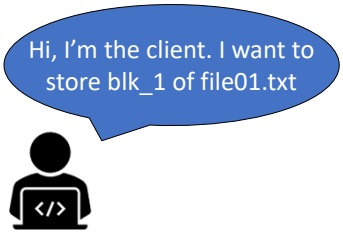
In our case the racks would consist of nodes in our cluster. That is, the machines used for our distributed storage and processing.



Yahoo's massive (!!) Hadoop cluster

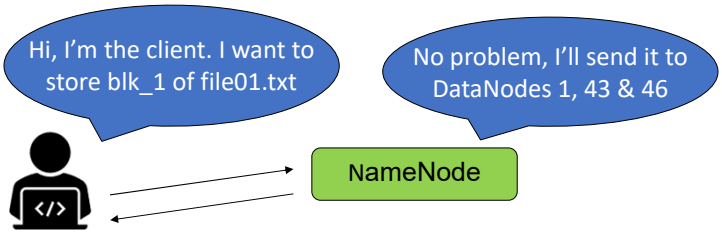
4

Rack awareness



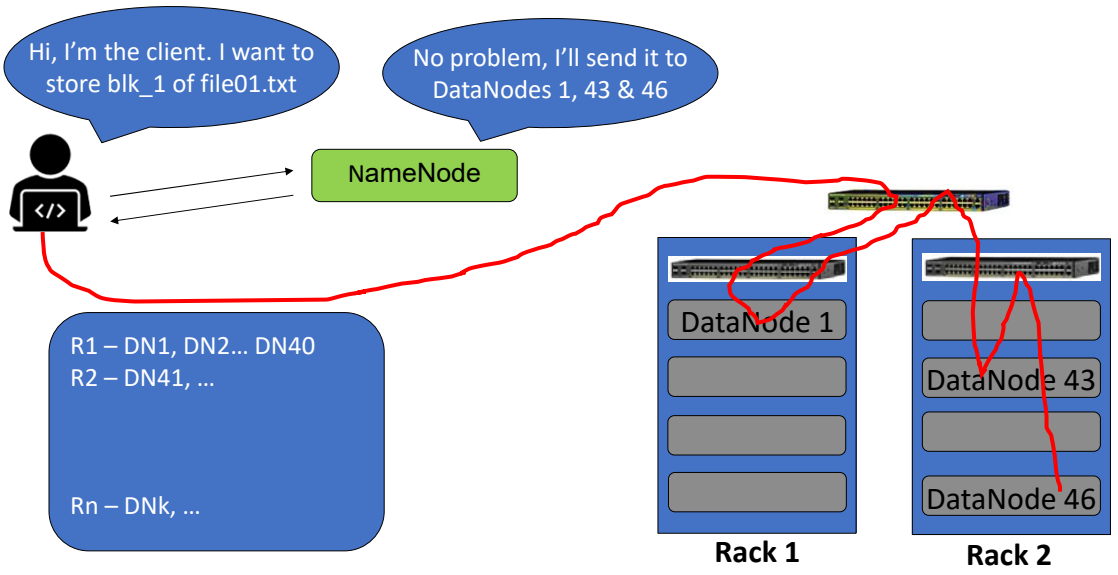
5

Rack awareness



6

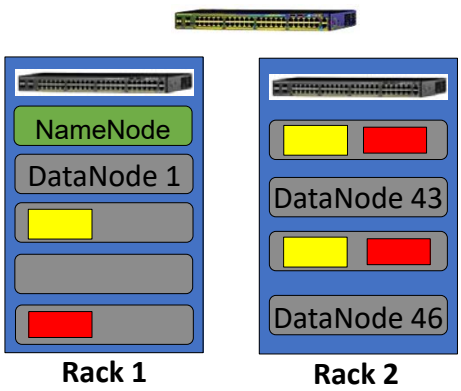
Rack awareness



7

Heartbeat

NameNode → DataNode



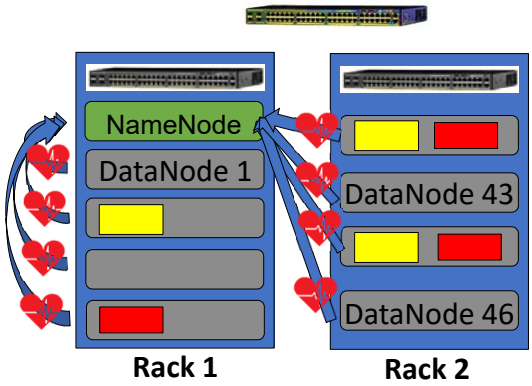
8

Heartbeat

NameNode ⟵ **DataNode**

By default a pulse is sent from the nameNode to the dataNode every three seconds

Once again… that value can be adjusted depending on context



9

How many parameters can we change?

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.replication.considerLoad	true	Decide if chooseTarget considers the target's load or not
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.
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 co 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 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 t
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
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 fil which can degrade performance.
dfs.namenode.fs-limits.max-blocks-per-file	1048576	Maximum number of blocks per file, enforced by the Namenode on write. This prevents the creation of extremely l
dfs.namenode.edits.dir	${dfs.namenode.name.dir}	Determines where on the local filesystem the DFS name node should store the transaction (edits) file. If this is a co transaction file is replicated in all of the directories, for redundancy. Default value is same as dfs.namenode.name.d
dfs.namenode.shared.edits.dir		A directory on shared storage between the multiple namenodes in an HA cluster. This directory will be written by th the namespaces synchronized. This directory does not need to be listed in dfs.namenode.edits.dir above. It should be
dfs.namenode.edits.journal-plugin.qjournal	org.apache.hadoop.hdfs.qjournal.client.QuorumJournalManager	
dfs.permissions.enabled	true	If "true", enable permission checking in HDFS. If "false", permission checking is turned off, but all other behavior is unchanged. Switching from one parameter value to the other does not change the mode, owner or group of files or directories.
dfs.permissions.superusergroup	supergroup	The name of the group of super-users.
dfs.namenode.acls.enabled	false	Set to true to enable support for HDFS ACLs (Access Control Lists). By default, ACLs are disabled. When ACLs are disabled, the NameNode rejects all RPCs related to setting or getting ACLs.
dfs.namenode.lazypersist.file.scrub.interval.sec	300	The NameNode periodically scans the namespace for LazyPersist files with missing blocks and unlinks them from the namespace. This configuration key controls the interval between successive scans. Set it to a negative value to disable this behavior.
dfs.block.access.token.enable	false	If "true", access tokens are used as capabilities for accessing datanodes. If "false", no access tokens are checked on accessing datanodes.
dfs.block.access.key.update.interval	600	Interval in minutes at which namenode updates its access keys.
dfs.block.access.token.lifetime	600	The lifetime of access tokens in minutes.
dfs.datanode.data.dir	file://${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. Directories that do not exist are ignored.
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.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

There's roughly 8 more tables of this size, full of values that we can change

10