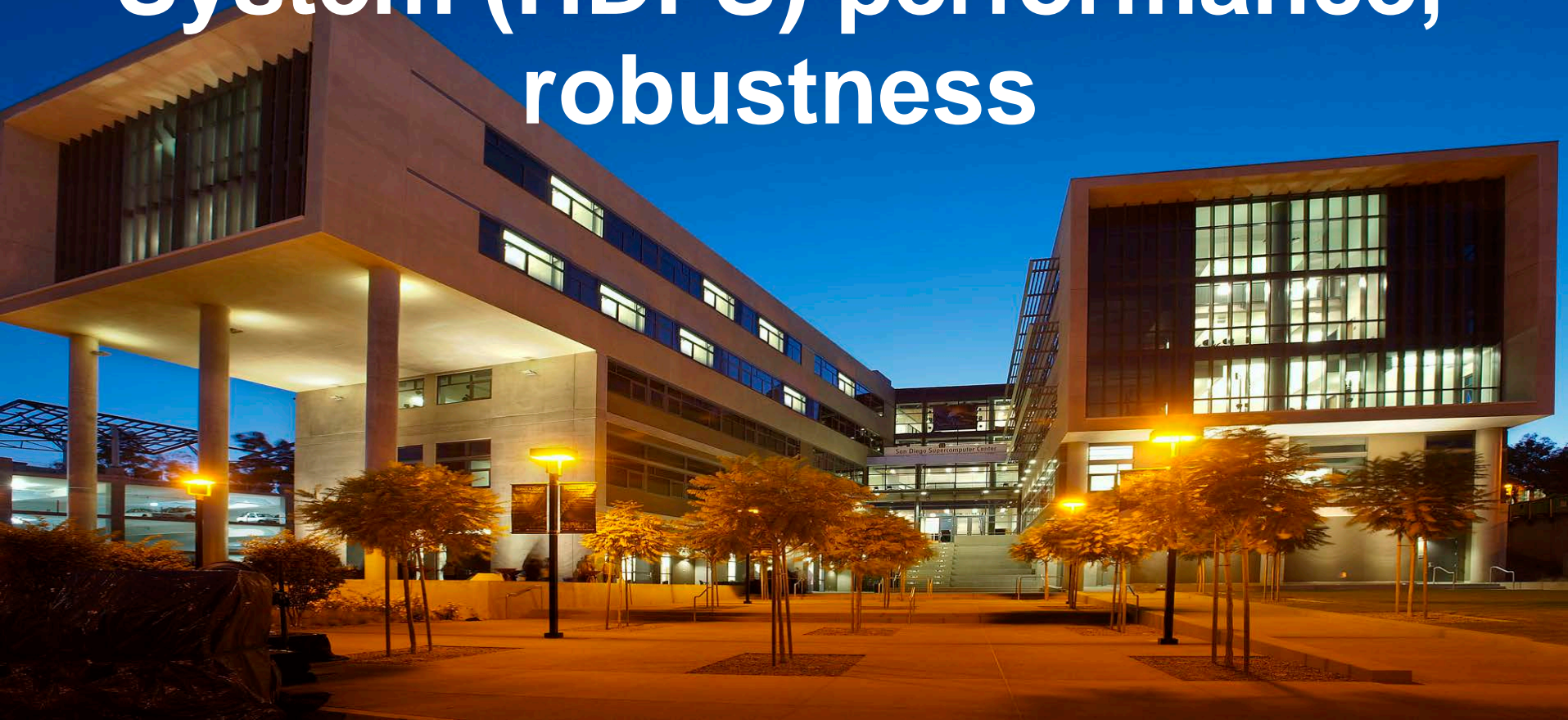


Hadoop Distributed File System (HDFS) performance, robustness

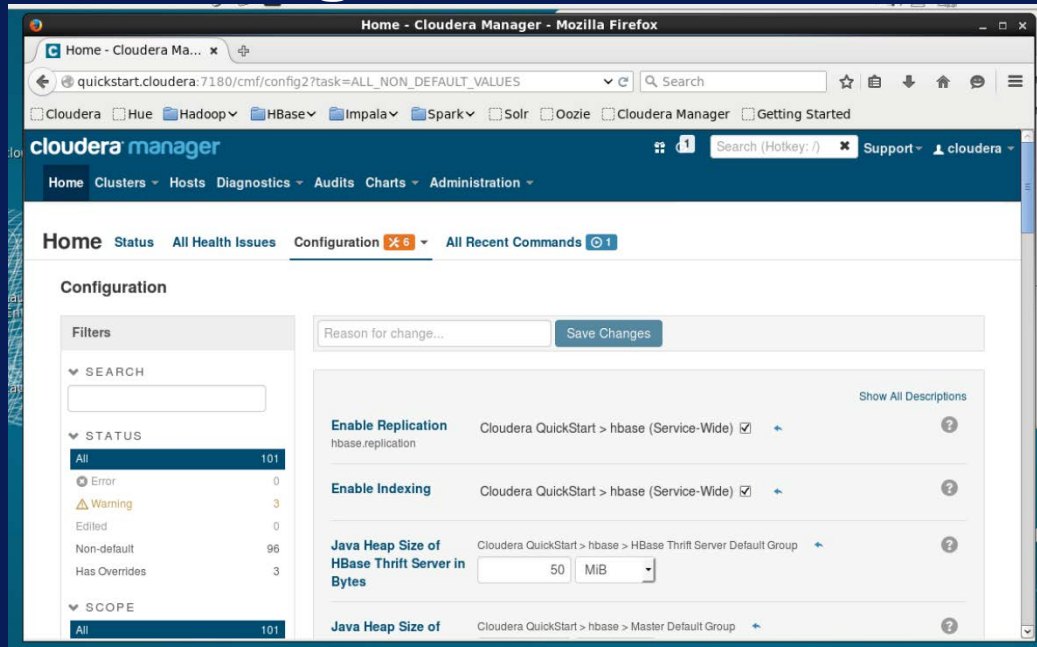


Overview of HDFS Tuning

- *Tuning parameters*
- *DFS Block Size*
- *NameNode, DataNode system/dfs parameters.*

HDFS Configuration file

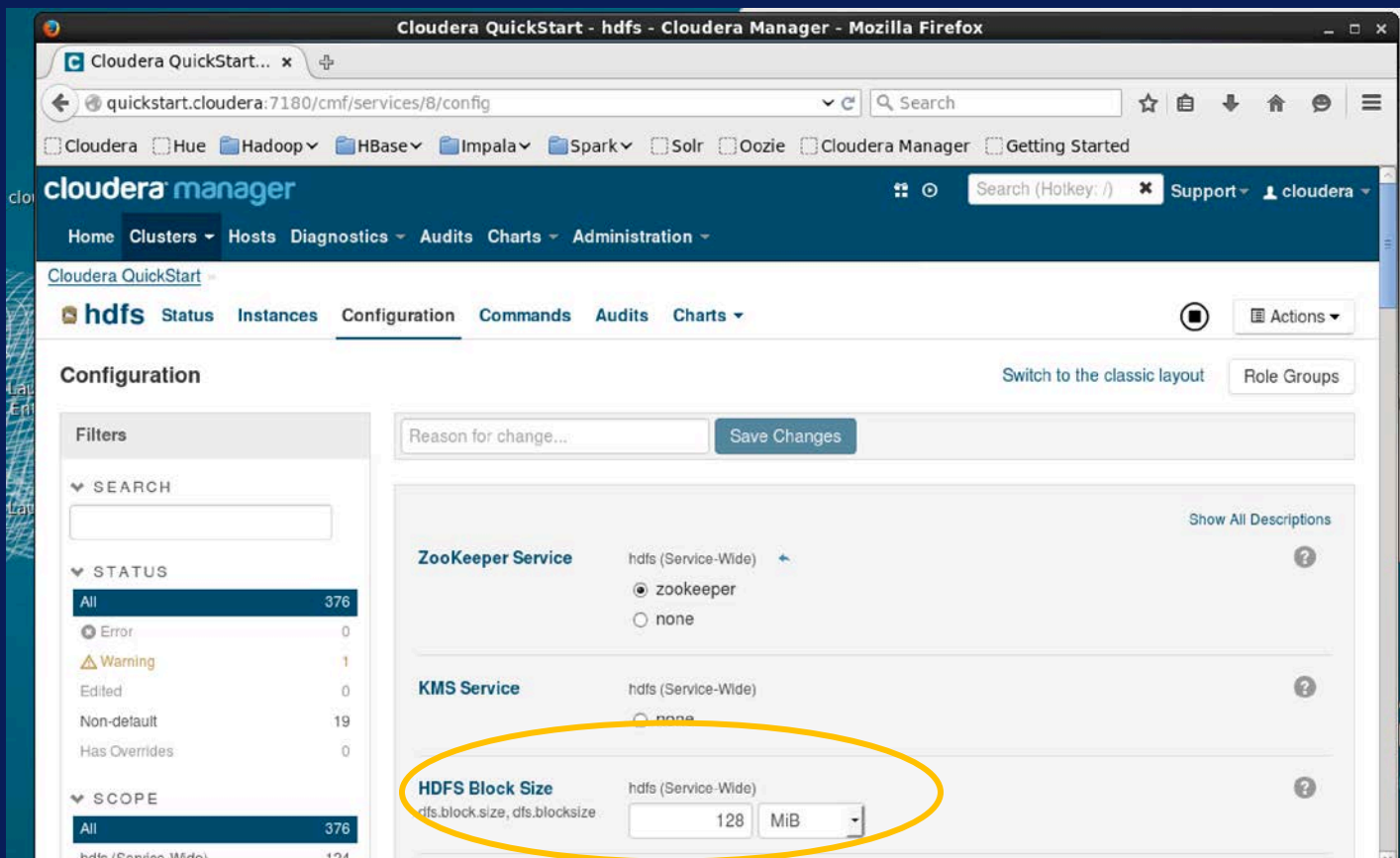
- *Parameters in hdfs-site.xml*
- Commercial vendors have GUI based management consoles to make changes.



HDFS Block Size

- Recall from previous video: *impacts NameNode memory, number of maps tasks, and hence performance.*
- *64MB is the default.* Can be changed based on workloads. Typically bumped up to 128MB.
- *dfs.blocksize, dfs.block.size*

HDFS Block Size



The screenshot shows the Cloudera Manager interface in a Mozilla Firefox browser. The address bar displays the URL `quickstart.cloudera:7180/cm/services/8/config`. The Cloudera Manager navigation bar includes links for Home, Clusters, Hosts, Diagnostics, Audits, Charts, and Administration. The main content area is titled "Configuration" and features a sidebar with filters for SEARCH, STATUS, and SCOPE. The central configuration panel lists services: ZooKeeper Service, KMS Service, and HDFS Block Size. The HDFS Block Size configuration is highlighted with a yellow oval, showing a value of 128 MiB. The configuration is set for the hdfs (Service-Wide) scope.

Cloudera QuickStart - hdfs - Cloudera Manager - Mozilla Firefox

Cloudera QuickStart... x

quickstart.cloudera:7180/cm/services/8/config

Cloudera Hue Hadoop HBase Impala Spark Solr Oozie Cloudera Manager Getting Started

cloudera manager

Home Clusters Hosts Diagnostics Audits Charts Administration

Cloudera QuickStart

hdfs Status Instances Configuration Commands Audits Charts

Configuration

Switch to the classic layout Role Groups

Filters

Reason for change... Save Changes

SEARCH

STATUS

SCOPE

ZooKeeper Service hdfs (Service-Wide) ?

zookeeper

none

KMS Service hdfs (Service-Wide) ?

none

HDFS Block Size hdfs (Service-Wide) ?

dfs.block.size, dfs.blocksize 128 MiB

HDFS Replication

- Default replication is 3.
- Parameter: *dfs.replication*
- Tradeoffs:
 - *Lower it to reduce replication cost*
 - *Less robust*
 - *Higher replication can make data local to more workers*
 - *Lower replication => more space*

HDFS Replication

The screenshot shows the Cloudera QuickStart configuration interface in a Mozilla Firefox browser. The address bar displays `quickstart.cloudera:7180/cmf/services/8/config`. The left sidebar contains a navigation menu with a 'SCOPE' section (All, hdfs (Service-Wide), Balancer, DataNode, Gateway, HttpFS, JournalNode, NFS Gateway, NameNode, SecondaryNameNode, Failover Controller) and a 'CATEGORY' section (All, Advanced, Checkpointing, Cloudera Navigator, High Availability, Logs, Main). The main content area lists several HDFS replication configuration parameters, each with a NameNode Default Group, a description, and a value input field. A yellow oval highlights the 'Replication Factor' setting, which is set to 1 for the 'hdfs (Service-Wide)' scope.

Configuration Parameter	NameNode Default Group	Value
Replication Work Multiplier Per Iteration <code>dfs.namenode.replication.work.multiplier.per.iteration</code>	NameNode Default Group	10
Maximum Number of Replication Threads on a DataNode <code>dfs.namenode.replication.max-streams</code>	NameNode Default Group	20
Hard Limit on the Number of Replication Threads on a Datanode <code>dfs.namenode.replication.max-streams-hard-limit</code>	NameNode Default Group	40
Replication Factor <code>dfs.replication</code>	hdfs (Service-Wide)	1
Minimal Block Replication <code>dfs.replication.min</code> , <code>dfs.namenode.replication.min</code>	hdfs (Service-Wide)	1

Lot of other parameters!

- *Various tunables for datanode, namenode.*

Examples:

- *dfs.datanode.handler.count* (10): Sets the number of server threads on each datanode.
- *dfs.namenode.fs-limits.max-blocks-per-file*: Maximum number of blocks per file.

Full list:

<http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/hdfs-default.xml>

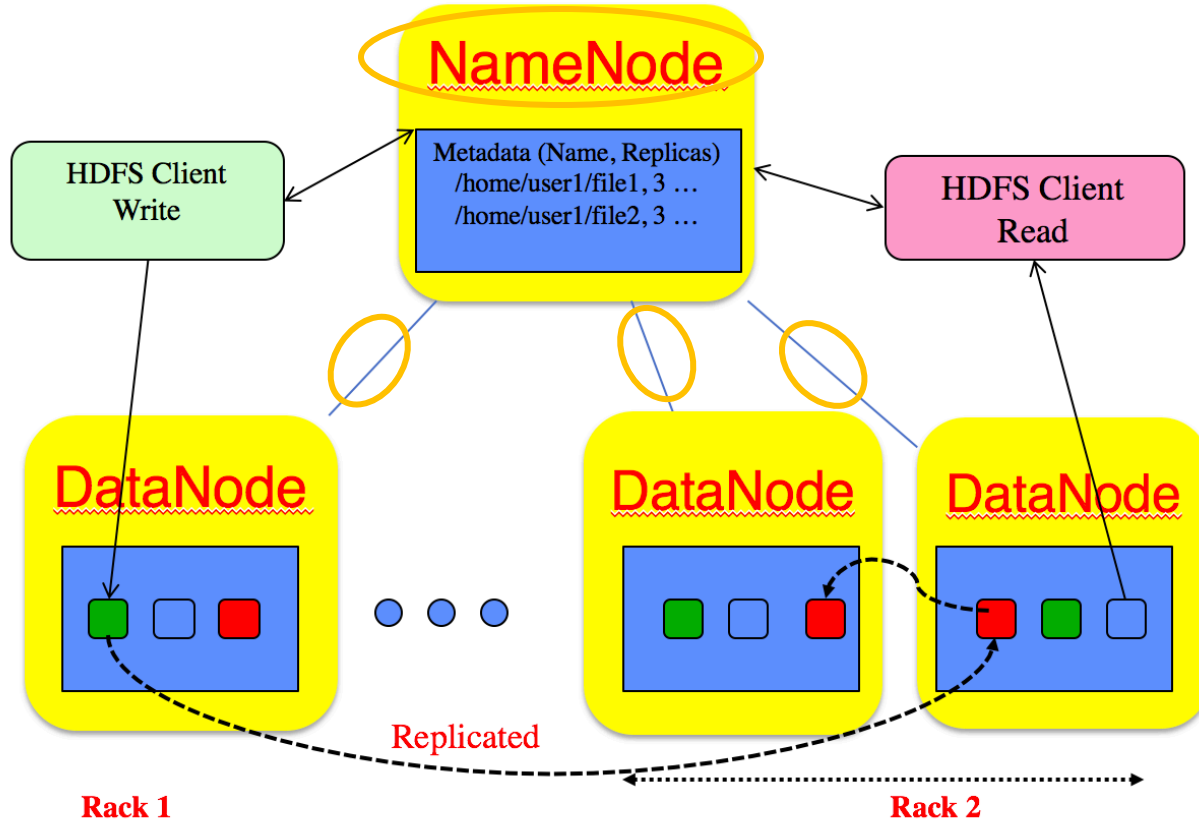
Performance & Robustness

- *How robustness is achieved*
- *Performance improvement and possible impact on robustness*

Common Failures

- ***DataNode Failures:*** Server can fail, disk can crash, data corruption.
- ***Network Failures***
- ***NameNode Failures:*** Disk failure, node failure

HDFS Robustness



- NameNode receives heartbeat and block reports from DataNodes*

Mitigation of common failures

- *Periodic heartbeat:* from DataNode to NameNode.
- DataNodes without recent heartbeat:
 - *Marked dead, no new IO sent*
 - *Blocks below replication factor re-replicated on other nodes.*

Mitigation of common failures

- *Checksum computed* on file creation
- *Checksums stored in HDFS namespace.*
- Used to check retrieved data, *re-read from alternate replica if need.*

Mitigation of common failures

- **Multiple copies of central meta data structures.**
- **Failover to standby
NameNode – manual by
default.**

Performance

- Recall: *changing blocksize and replication factor can improve performance*

Example: Distributed copy

- *Hadoop distcp allows parallel transfer of files.*
- This example is from one of SDSC's systems called Gordon.
- Copy 32 files and 512GB of data.
- Vary map and node counts, replication.

Distributed Copy, Replication=3

Nodes	#Map Tasks	Copy Rate (MB/s)
8	2	183.64
16	2	220.57
16	4	276.09
16	8	387.51
32	8	618.99
32	16	779.03

Distributed Copy, Replication=1

Nodes	#Map Tasks	Copy Rate (MB/s)
8	4	530.66
8	8	1018.03
16	8	793.17
16	16	1814.15
32	32	2995.93

Replication trade off w.r.t robustness.

- Reducing replication has a trade off w.r.t robustness:
 - Might lose a node or local disk during the run – *cannot recover if there is no replication.*
 - If there is data corruption of a block from one of the datanodes – *again cannot recover without replication.*
- Just one example. *Other parameter changes can have similar effects.* For example we saw with block size changes you can impact Namenode