The Elephant in the Room

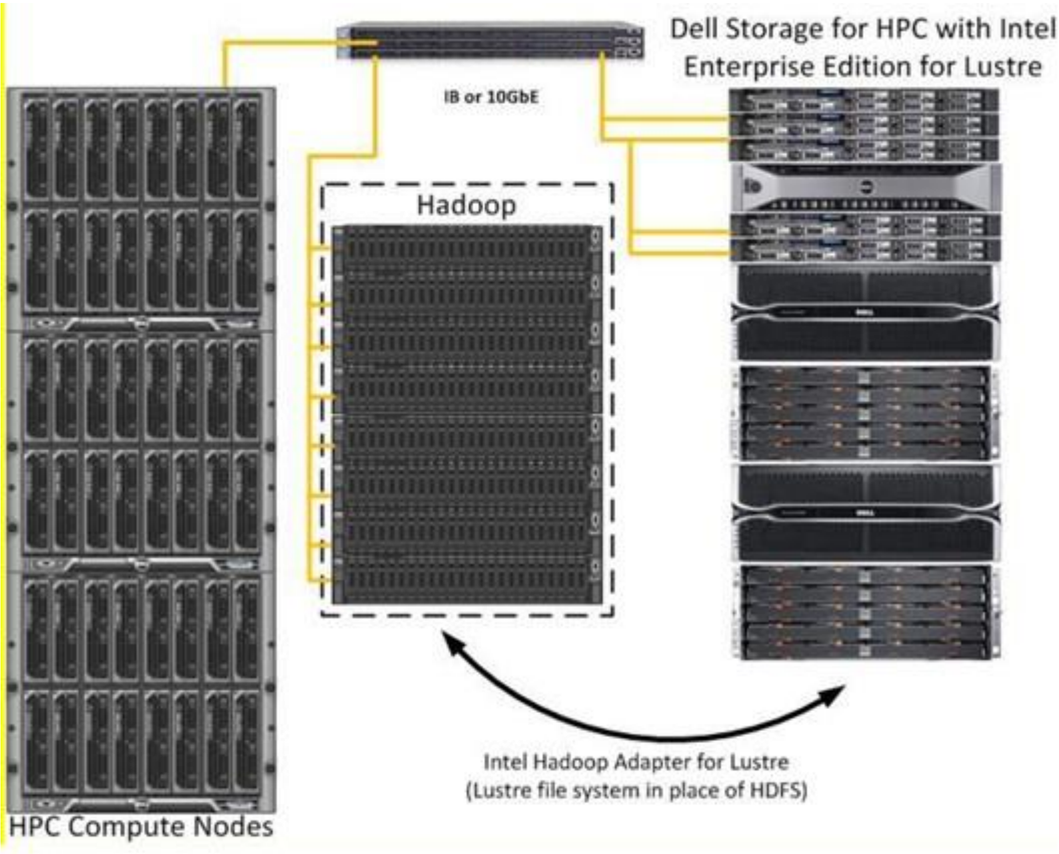
Quy Ta **23 Mar 2015**

This blog will explore a hybrid computing environment that takes Lustre®, a high performance parallel file system and integrates it with Hadoop®, a framework for processing and storing big data in a distributed environment. We will explore some reasons and benefits of such a hybrid approach and provide a foundation on how to easily and quickly implement the solution using Bright Cluster Manager® (BCM) to deploy and configure the hybrid cluster.

First, let's establish some definitions and technologies for our discussion. Hadoop is a software framework for distributed storage and processing of typically very large data sets on compute clusters. The Lustre file system is a parallel distributed file system that is often the choice for large scale computing clusters. In the context of this blog, we define a hybrid cluster as taking a traditional HPC cluster and integrating a Hadoop computing environment capable of processing MapReduce jobs using the Lustre File System. The hybrid solution that we will use as an example in this blog was jointly developed and consists of components from Dell, Intel, Cloudera and Bright Computing.

Why would you want to use the Lustre file system with Hadoop? Why not just use the native Hadoop file system, HDFS? Scientists and researchers have been looking for ways to use both Lustre and Hadoop from within a shared HPC infrastructure. This hybrid approach will allow them to use Lustre as both the file system for Hadoop analytics work as well as the file system for their general HPC workloads. They can also avoid standing up two different clusters (HPC and Hadoop), and the associated resources required, by allowing the re-purposed provisioning of the existing HPC cluster resources into a small to medium sized self-contained Hadoop cluster. This solution would typically target those HPC users that have a need to run periodic Hadoop specific jobs.

A key component to connecting the Hadoop and Lustre ecosystems is the Intel Hadoop Adapter for Lustre plug-in or Intel HAL for short. Intel HAL is bundled with the Intel Enterprise Edition for Lustre software. It allows the users to run MapReduce jobs directly on a Lustre file system. The immediate benefit is that Lustre is able to deliver faster, stable and easily managed storage for the MapReduce applications directly. A potential long term benefit using Lustre as the underlying Hadoop storage would be a higher raw capacity available when compared to HDFS due to the three time replication as well as the performance benefits of running Lustre on InfiniBand connectivity. The following architectural diagram will illustrate a typical topology for the hybrid solution.


Dell Storage for HPC with Intel Enterprise Edition for Lustre
IB or 10GbE
Hadoop
Intel Hadoop Adapter for Lustre
(Lustre file system in place of HDFS)
HPC Compute Nodes

The following will be a high level recount of how we easily implement the solution using the BCM tool to deploy and configure.

The first thing we did was to establish an optimized and fully functional Lustre environment. For this solution, we used the **Dell Storage for HPC with Intel Enterprise Edition (EE) for Lustre software** as the Lustre solution, **Cloudera CDH** as the Hadoop distribution and **Bright Cluster Manager (BCM)** as the imaging and cluster deployment tool.

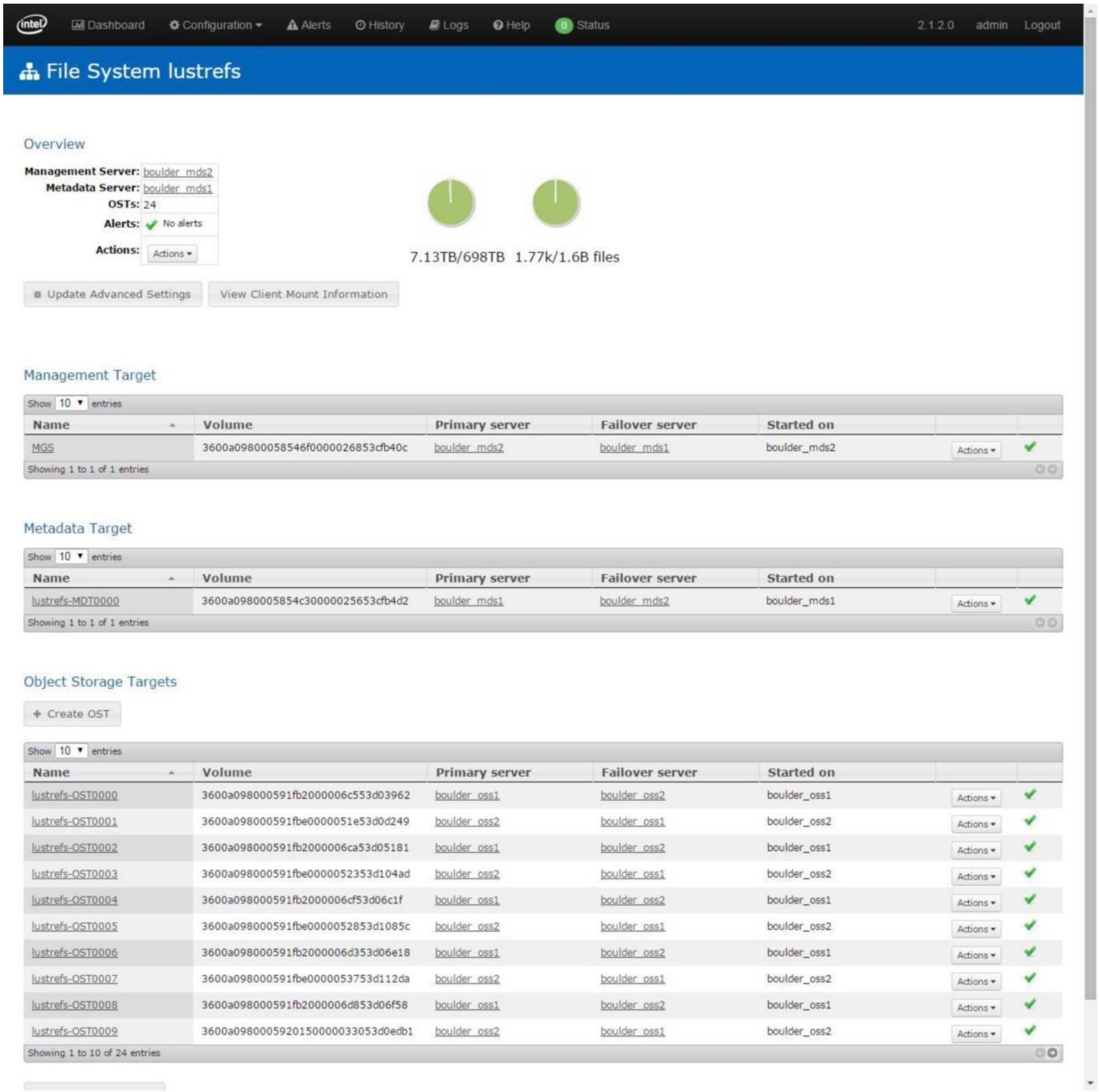
Using the Intel Manager for Lustre (IML) GUI interface, we verified the MDT and OST objects are healthy and in an optimal state. We also verified that the LNet interface and the Lustre Kernel modules are loaded and Lustre NIDS are accessible.

Verify contents of /etc/modprobe.d/iml_lnet_module_parameters.conf are correct for each MDS and OSS server. Example below.

```
[root@boulder_mds1 ~]# cat /etc/modprobe.d/iml_lnet_module_parameters.conf

# This file is auto-generated for Lustre NID configuration by IML

# Do not overwrite this file or edit its contents directly

options lnet networks=o2ib0(ib0)

### LNet Configuration Data

##  {

##    "state": "lnet_unloaded",

##    "modprobe_entries": [
##     "o2ib0(ib0)"
##   ],
##   "network_interfaces": [
##     [
##       "10.149.255.250",
##       "o2ib",
##       0
##     ]
##   ]
## }
[root@boulder_mds1 ~]#
```

Using the IML GUI, verify status of MDT and OST objects. There should be no file system alerts and all MDT and OST objects should have green status.

Configuration > File Systems > Current File Systems > “lustrefs”



File System lustrefs

Overview

Management Server:	boulder_mds2
Metadata Server:	boulder_mds1
OSTs:	24
Alerts:	✓ No alerts
Actions:	Actions ▾

7.13TB/698TB 1.77k/1.6B files

Management Target

Show 10 entries

Name	Volume	Primary server	Failover server	Started on		
MGS	3600a09800058546f0000026853cfb40c	boulder_mds2	boulder_mds1	boulder_mds2	Actions ▾	✓

Showing 1 to 1 of 1 entries

Metadata Target

Show 10 entries

Name	Volume	Primary server	Failover server	Started on		
lustrefs-MDT0000	3600a0980005854c30000025653cfb4d2	boulder_mds1	boulder_mds2	boulder_mds1	Actions ▾	✓

Showing 1 to 1 of 1 entries

Object Storage Targets

Show 10 entries

Name	Volume	Primary server	Failover server	Started on		
lustrefs-OST0000	3600a098000591fb2000006c553d03962	boulder_oss1	boulder_oss2	boulder_oss1	Actions ▾	✓
lustrefs-OST0001	3600a098000591fbe0000051e53d0d249	boulder_oss2	boulder_oss1	boulder_oss2	Actions ▾	✓
lustrefs-OST0002	3600a098000591fb2000006ca53d05181	boulder_oss1	boulder_oss2	boulder_oss1	Actions ▾	✓
lustrefs-OST0003	3600a098000591fbe0000052353d104ad	boulder_oss2	boulder_oss1	boulder_oss2	Actions ▾	✓
lustrefs-OST0004	3600a098000591fb2000006cf53d06c1f	boulder_oss1	boulder_oss2	boulder_oss1	Actions ▾	✓
lustrefs-OST0005	3600a098000591fbe0000052853d1085c	boulder_oss2	boulder_oss1	boulder_oss2	Actions ▾	✓
lustrefs-OST0006	3600a098000591fb2000006d353d06e18	boulder_oss1	boulder_oss2	boulder_oss1	Actions ▾	✓
lustrefs-OST0007	3600a098000591fbe0000053753d112da	boulder_oss2	boulder_oss1	boulder_oss2	Actions ▾	✓
lustrefs-OST0008	3600a098000591fb2000006d853d06f58	boulder_oss1	boulder_oss2	boulder_oss1	Actions ▾	✓
lustrefs-OST0009	3600a09800059201500000033053d0edb1	boulder_oss2	boulder_oss1	boulder_oss2	Actions ▾	✓

Showing 1 to 10 of 24 entries

Verify that UIDs and GIDs are consistent on Lustre clients. This must be done before installing Hadoop software. In particular, the following users and groups should be checked:

users: hdfs, mapred, yarn, hbase, zookeeper

groups: hadoop, zookeeper, hbase

We used the following script to set up our Hadoop users prior to installing Hadoop:

```
VALUE=10000;

for i in hive hbase hdfs mapred yarn;

do
```

```
    VALUE=$(expr $VALUE + 1);

    groupadd -g $VALUE $i;

    adduser -u $VALUE -g $VALUE $i;

done;

groupadd -g 10006 hadoop;

groupmems -g hadoop -a yarn;

groupmems -g hadoop -a mapred;

groupmems -g hadoop -a hdfs;

usermod -d /var/lib/hive -s /sbin/nologin hive;

usermod -d /var/run/hbase -s /sbin/nologin hbase;

usermod -d /var/lib/hadoop-yarn -s /sbin/nologin yarn;

usermod -d /var/lib/hadoop-mapreduce -s /sbin/nologin mapred;

usermod -d /var/lib/hadoop-hdfs -s /bin/bash hdfs
```

As a sanity check, we verified the nodes we wanted to re-provision as Hadoop nodes that were able to read/write to the Lustre file system.

Once we verified all the pre-requisite items above and established that we had a working Lustre environment, we proceeded with the following steps to build, configure and deploy the Hadoop nodes that mount and use the Lustre file system.

Steps we took to build the hybrid solution:

1) Created a software image ‘ieel-hadoop’ using BCM. You can clone an existing software image.

2) Created a node category ‘ieel-hadoop’ using BCM. You can clone an existing node category.

3) We assigned the selected nodes to be provisioned as Hadoop nodes to the ieel-hadoop node category.

4) Installed Cloudera CDH 5.1.2 and the Intel Hadoop Adapter for Lustre (HAL) plug-in into the ieel-hadoop software image.

5) We installed the Intel EE for Lustre client software onto the ieel-hadoop software image.

6) We prepared the Lustre directory for Hadoop on the ieel-hadoop software image.

Example:

```
#chmod 0777 /mnt/lustre/Hadoop

#setfacl –R –m group:hadoop:rwx /mnt/lustre/hadoop
```

```
#setfacl –R –d –m group:hadoop:rwx /mnt/lustre/hadoop
```

7) Added the Lustre file system mount point to the ieel-hadoop node category for automatic mounting upon bootup.

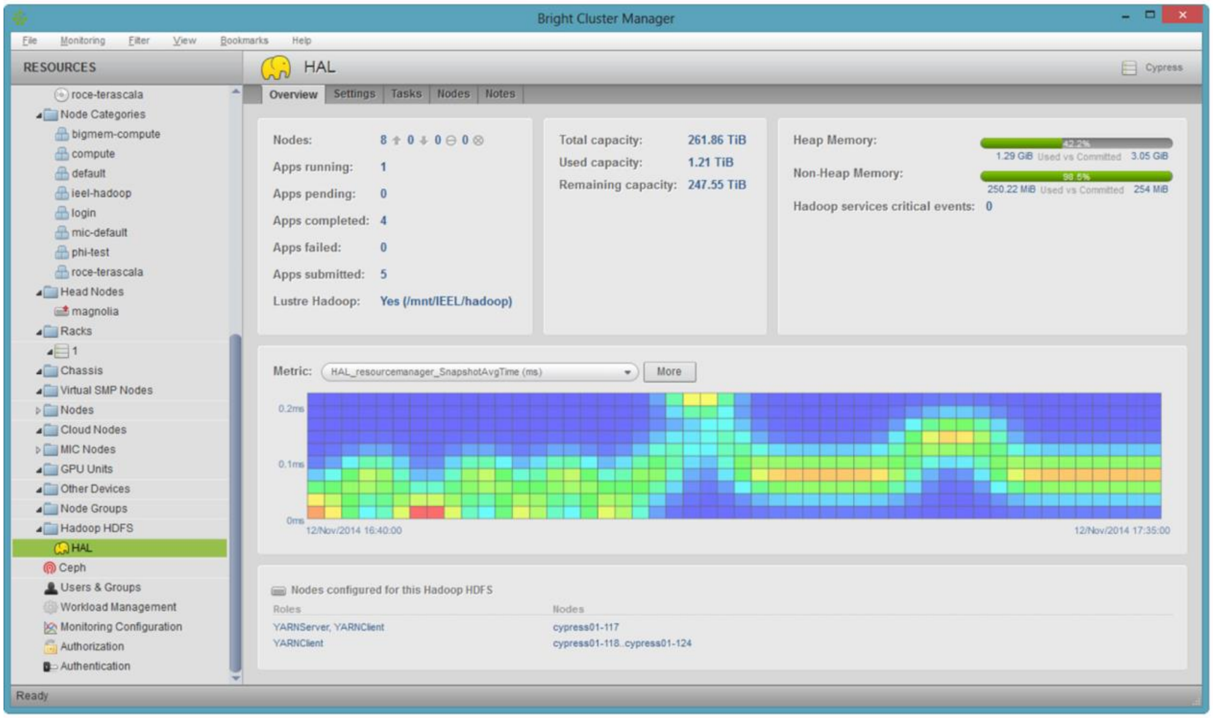
Example: 192.168.4.140@tcp0:192.168.4.141@tcp0:/lustre /mnt/lustre   lustre defaults,_netdev 0 0

8) We added several tuning parameters to /etc/sysctl.conf in the ieel-hadoop software image.

Example:

```
lctl set_param osc.*.max_dirty_mb=512

lctl set_param osc.*.max_rpcs_in_flight=32
```

CDH running on Intel EE for Lustre



Nodes: 8 ↑ 0 ↓ 0 ⊖ 0 ⊗

Apps running: 1

Apps pending: 0

Apps completed: 4

Apps failed: 0

Apps submitted: 5

Lustre Hadoop: Yes (/mnt/IEEL/hadoop)

Total capacity: 261.86 TiB

Used capacity: 1.21 TiB

Remaining capacity: 247.55 TiB

Heap Memory: 42.2% 1.29 GiB Used vs Committed 3.05 GiB

Non-Heap Memory: 98.5% 250.22 MiB Used vs Committed 254 MiB

Hadoop services critical events: 0

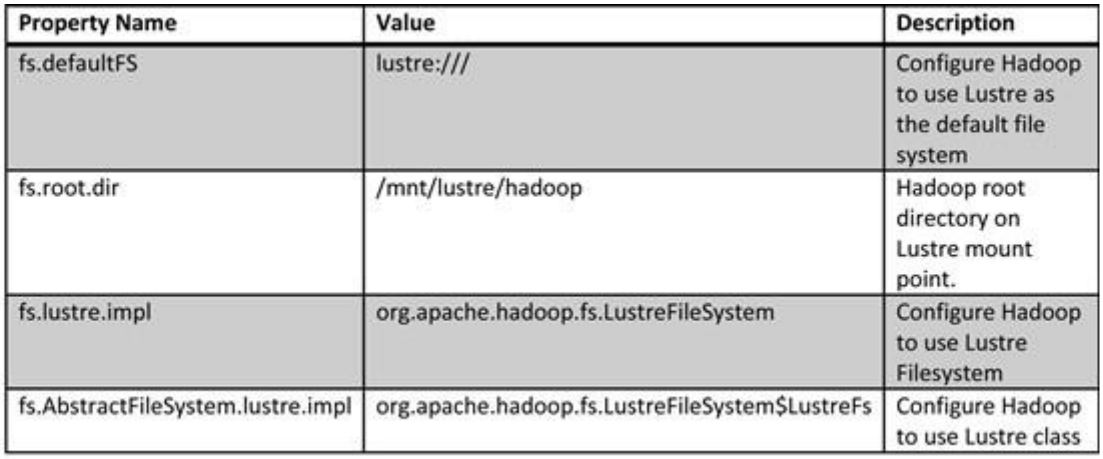
Metric: HAL_resourcemanager_SnapshotAvgTime (ms)

Nodes configured for this Hadoop HDFS

Roles	Nodes
YARNServer, YARNClient	cypress01-117
YARNClient	cypress01-118..cypress01-124

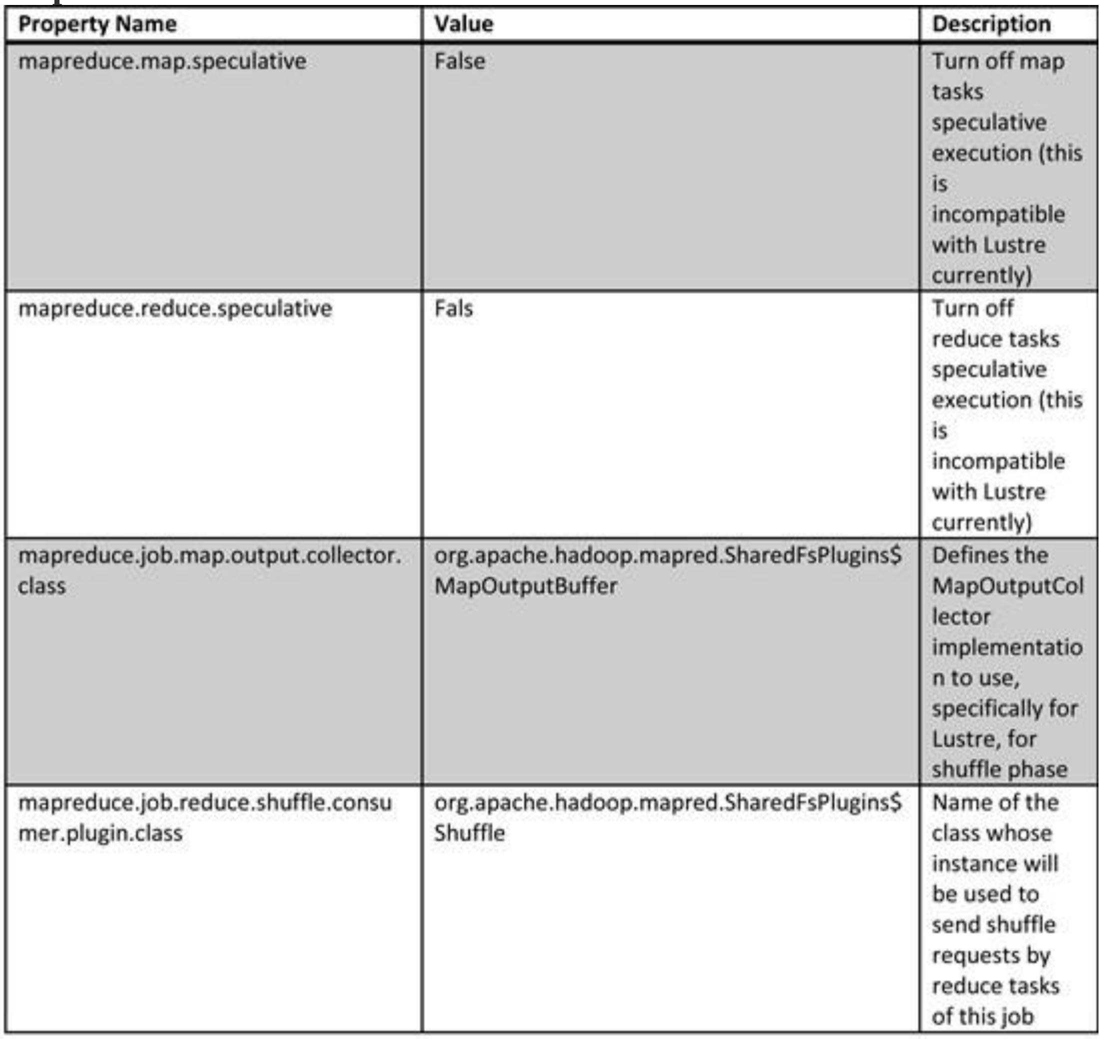
To further optimize the solution, you can edit the core-site.xml and mapred-site.xml with the following Hadoop configuration for Lustre.

- **core-site.xml**



Property Name	Value	Description
fs.defaultFS	lustre:///	Configure Hadoop to use Lustre as the default file system
fs.root.dir	/mnt/lustre/hadoop	Hadoop root directory on Lustre mount point.
fs.lustre.impl	org.apache.hadoop.fs.LustreFileSystem	Configure Hadoop to use Lustre Filesystem
fs.AbstractFileSystem.lustre.impl	org.apache.hadoop.fs.LustreFileSystem$LustreFs	Configure Hadoop to use Lustre class

- **mapred-site.xml**



Property Name	Value	Description
mapreduce.map.speculative	False	Turn off map tasks speculative execution (this is incompatible with Lustre currently)
mapreduce.reduce.speculative	Fals	Turn off reduce tasks speculative execution (this is incompatible with Lustre currently)
mapreduce.job.map.output.collector.class	org.apache.hadoop.mapred.SharedFsPlugins$MapOutputBuffer	Defines the MapOutputCollector implementation to use, specifically for Lustre, for shuffle phase
mapreduce.job.reduce.shuffle.consumer.plugin.class	org.apache.hadoop.mapred.SharedFsPlugins$Shuffle	Name of the class whose instance will be used to send shuffle requests by reduce tasks of this job