Hadoop Virtualization Extensions on VMware vSphere® 5

TECHNICAL WHITE PAPER

Table of Contents

Introduction

Apache Hadoop has emerged as the tool of choice for big data analytics, and virtualizing Hadoop brings many benefits[1], including:

- Rapid provisioning – From the creation of virtual Hadoop nodes to starting up the Hadoop services on the cluster, much of the Hadoop cluster deployment can be automated, requiring little expertise on the user's part. Virtual Hadoop clusters can be rapidly deployed and configured as needed.
- High availability – Reliability is critical for certain mission-critical uses of Hadoop. HA protection can be provided through the virtualization platform to protect the single points of failure in the Hadoop system, such as NameNode and JobTracker.
- Elasticity – Hadoop capacity can be scaled up and down on demand in a virtual environment, thus allowing the same physical infrastructure to be shared among Hadoop and other applications. This consolidation of workloads results in more efficient resource utilization and reduced costs.
- Multi-tenancy – Different tenants running Hadoop can be isolated in separate VMs, providing stronger VM-grade resource and security isolation. With virtualization, mixed workloads that include non-Hadoop applications can run alongside Hadoop on the same physical cluster.

Hadoop Virtualization Extensions (HVE) allow Apache Hadoop clusters implemented on virtualized infrastructure full awareness of the topology on which they are running, thus enhancing the reliability and performance of these clusters.

Topology Awareness in Hadoop

As an open source implementation of MapReduce, the key strategy of Apache Hadoop is “divide and conquer,” so as to leverage maximum local I/O bandwidth and to eliminate the effects of network bottlenecks. Thus, awareness of node-level data locality in scheduling tasks and choosing which replica to read from helps improve the scalability and performance of Hadoop.

Also, Hadoop delivers a reliable storage and MapReduce service on unreliable hardware, so it needs to deal with node or even rack failure which includes keeping data integration from node/rack failure. As a result, awareness of rack level failure groups among nodes and placing replicas across failure groups helps to enhance the reliability of Hadoop.

Virtual Hadoop

Some of the benefits of virtualizing Hadoop—the enablement of elasticity and multi-tenancy—arise from the increased number of deployment options available when Hadoop is virtualized.

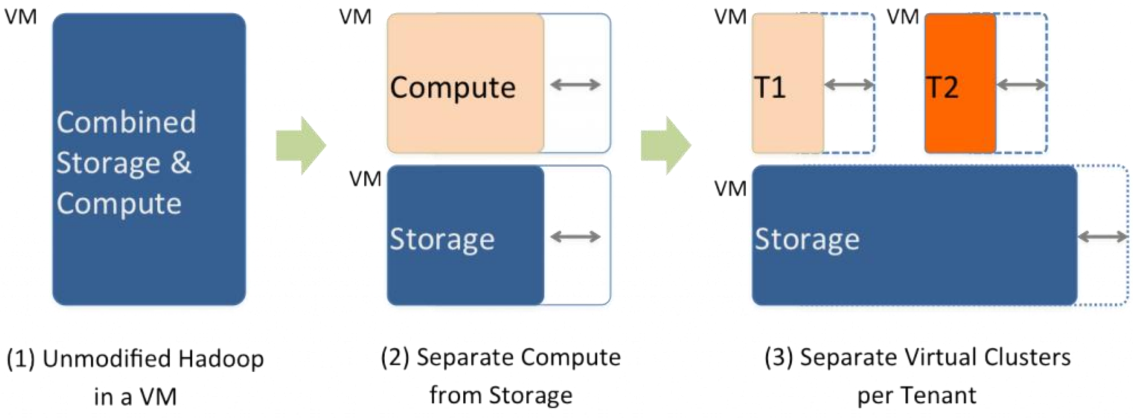
VM
Combined Storage & Compute
VM
Compute
VM
Storage
VM
T1
VM
T2
VM
Storage
(1) Unmodified Hadoop in a VM
(2) Separate Compute from Storage
(3) Separate Virtual Clusters per Tenant

Figure 1: The Evolution of Virtual Hadoop

Compute and data are combined in the traditional Hadoop model. While this implementation is straightforward, representing how the physical Hadoop model can be directly translated into a VM, the ability to scale up and down is limited because the lifecycle of this type of VM is tightly coupled to the data it manages. Powering off a VM with combined storage and compute means access to its data is lost, while scaling out by adding more nodes would necessitate rebalancing data across the expanded cluster, so this model is not particularly elastic.

Separating compute from storage in a virtual Hadoop cluster can achieve compute elasticity, enabling mixed workloads to run on the same virtualization platform and improving resource utilization. This is quite simple to configure using a HDFS data layer that is always available along with a compute layer comprising a variable number of TaskTracker nodes, which can be expanded and contracted on demand.

Extending the concept of data-compute separation, multiple tenants can be accommodated on the virtualized Hadoop cluster by running multiple Hadoop compute clusters against the same data service. Using this model, each virtual compute cluster enjoys performance, security and configuration isolation.

While Hadoop performance using the combined data-compute model on vSphere is similar to its performance on physical hardware[2], providing virtualized Hadoop increased topology awareness can enable the data locality needed to improve performance when data and compute layers are separated. Topology awareness allows Hadoop operators to realize elasticity and multi-tenancy benefits when data and compute are separated. Furthermore, topology awareness can improve reliability when multiple nodes of the same Hadoop cluster are co-located on the same physical host.

Opportunities to optimize the data locality and failure group characteristics of virtualized Hadoop include:

- Virtual Hadoop nodes on the same physical host are grouped into the same failure domain, on which placement of multiple replicas should be avoid.

- Maximizing usage of the virtual network between virtual nodes on the same physical host, which has higher throughput and lower latency and does not consume any physical switch bandwidth.

Thus, virtual Hadoop nodes on the same physical host are put into the same failure and locality group which can be optimized for in existing Hadoop mechanisms, such as: replica placement, task scheduling, balancing, etc.

HVE Solution

HVE is a new feature that extends the Hadoop topology awareness mechanism to account for the virtualization layer and refine data-locality-related policies. It is designed to enhance the reliability and performance of virtualized Hadoop clusters with extended topology layer and refined locality related policies. Test results documented in the following sections of this paper do show that reliability and performance can be optimized by taking advantage of this awareness of the virtualization topology.

HVE was developed by VMware engineers working with Apache Hadoop open source community, and the resulting patches have been contributed back to Apache Hadoop on trunk and branch-1. Umbrella JIRA HADOOP-8468[3] tracks progress of this effort in the Apache community.

Architecture

HVE consists of hooks and extensions to data-locality-related components of Hadoop, including: Network Topology, HDFS write/read, balancer, and task scheduling. It touches all sub projects of Hadoop: Common, HDFS and MapReduce. The architecture is shown below:

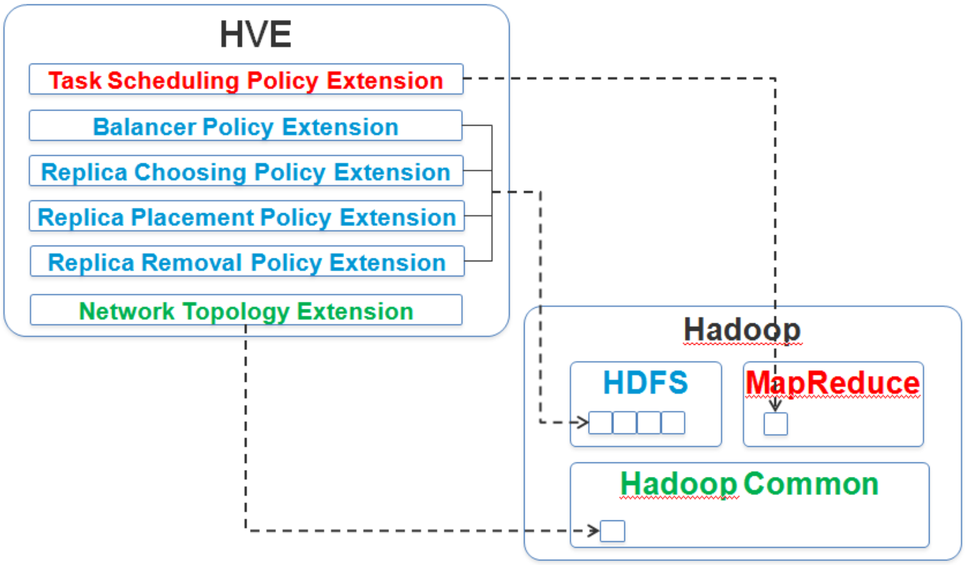
HVE
Task Scheduling Policy Extension
Balancer Policy Extension
Replica Choosing Policy Extension
Replica Placement Policy Extension
Replica Removal Policy Extension
Network Topology Extension
Hadoop
HDFS
MapReduce
Hadoop Common

Figure 2: HVE Architecture

Network Topology Extension

To model the new failure and locality characteristics of running Hadoop in a virtualized environment, HVE introduces a new layer in the topology hierarchy as shown below.

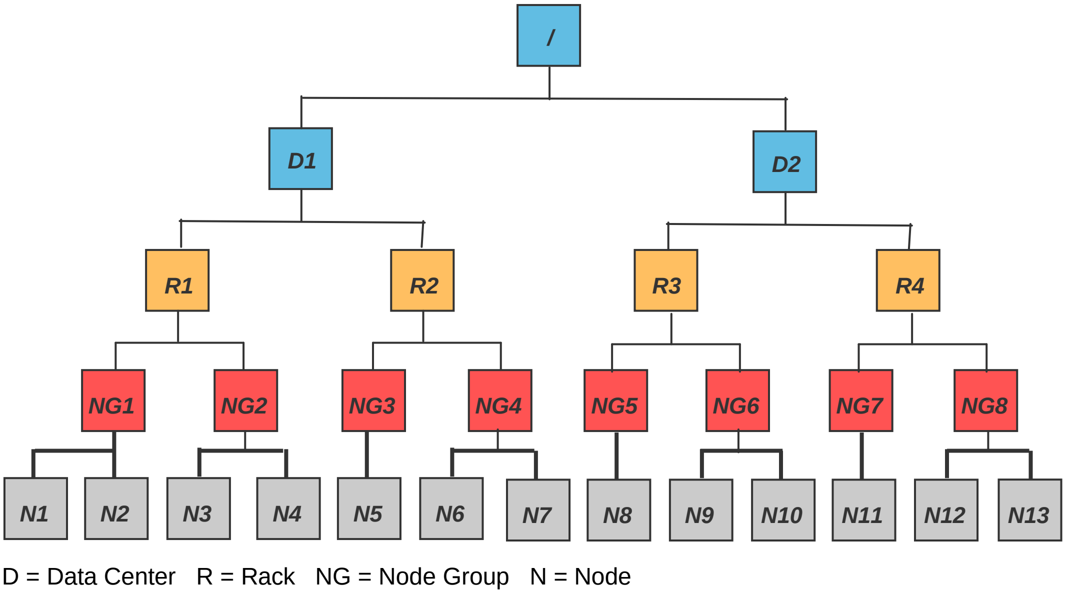

Figure 3: Network Topology Extension

The new layer, marked “NG” in Figure 2, is called the node group layer and represents the virtualization layer. All VMs under the same node group run on the same physical host. With awareness of the node group layer, HVE can refine locality-based polices to optimize performance in a virtualized environment.

HDFS Policies Extension

HVE on HDFS policies include: replica placement/removal policy extension, replica choosing policy extension and balancer policy extension.

Replica Placement/Removal Policy Extension

Replica placement policy in HDFS applies rules in choosing data nodes to place required replicas or to remove excessive replicas.

The existing replica placement policy includes:

- Multiple replicas are not placed on the same node
- 1st replica is on the local node of the writer;
- 2nd replica is on a remote rack of the 1st replica;
- 3rd replica is on the same rack as the 2nd replica;
- Remaining replicas are placed randomly across rack to meet minimum restriction.

With awareness of the node group, the extended replica placement policy includes:

- Multiple replicas are not be placed on the same node **or on nodes under the same node group**
- 1st replica is on the local node **or local node group** of the writer;
- 2nd replica is on a remote rack of the 1st replica;

- 3rd replica is on the same rack as the 2nd replica;
- Remaining replicas are placed randomly across rack **and node group** to meet minimum restriction.

With awareness of the node group and the refined replica placement/removal policy, the reliability of Hadoop running on virtualized environment is enhanced as shown in the following case.

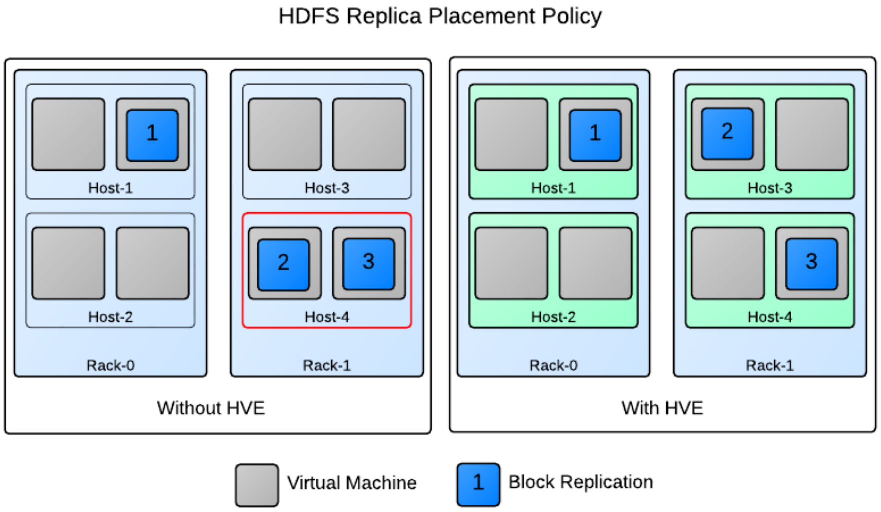

Figure 4: HDFS Replica Placement Policy Extension

In cases with HVE disabled, it is possible to place two replicas on the same physical host (Without HVE – Host-4) although they are in different VMs. HVE, with enhanced replica placement policy extension, will ensure replicas are placed across physical hosts to achieve reliability.

Replica Choosing Policy Extension

With the replica choosing policy, the HDFS client can pick up the nearest replica from a list of replicas for a specific block to achieve the best read throughput. The policy is based on sorting based on distance between each replica node and client machine in the network topology tree. Under the previous policy, the distance from nearest (distance is 0) to farthest (distance is 4) is local node (0), local rack (2), off rack (4). With awareness of the node group, now the distance from nearest (distance is 0) to farthest (distance is 6) is updated to local node (0), **local node group (2)**, local rack (**4**), off rack (**6**).

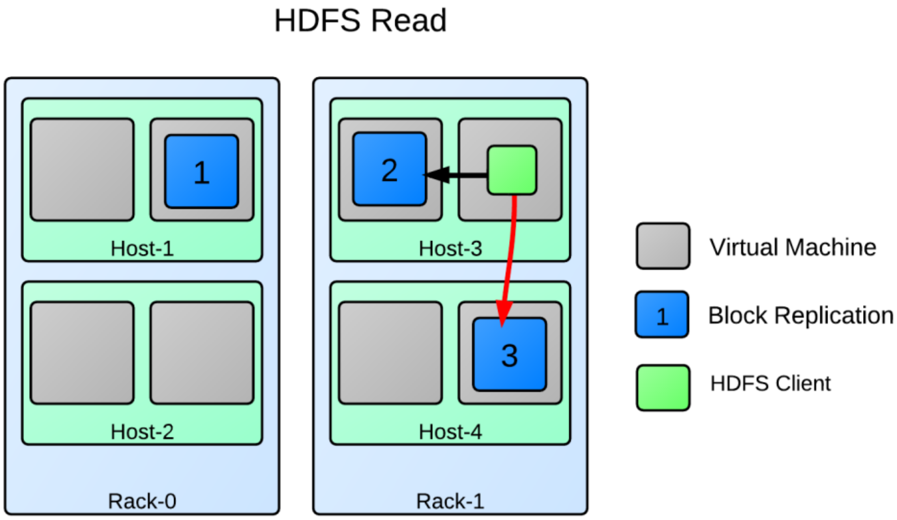

Figure 5: HDFS Replica Choosing Policy Extension

As the above case shows, without HVE in Hadoop, a HDFS client has an equal chance to read the block from ***replica 2*** and ***replica 3***. But with node group awareness and refined policy in replica choosing in HVE, it can be guaranteed to always choose ***replica 2*** for reading the block, which can achieve better read throughput as it benefits from the higher bandwidth and lower latency available between VMs residing on the same physical host.

Balancer Policy Extension

Balancer policy in Hadoop applies rules on choosing blocks to transfer between under-utilized nodes and over-utilized nodes. It contains rules in two levels: At node level, it sets up pairs of under-utilized node and over-utilized node based on distances between nodes to limit network traffic as much as possible in doing balancing. At the block level, it checks qualified blocks to move without violating replica placement policy. With awareness of node groups, the balancer policy extension to Hadoop includes the following update (marked in **Bold**) compared to the previous balancer policy:

- At the node level, the choice of target and source for balancing follows the sequence: **local node group**, local rack, off rack.

- At the block level, a replica block is not a good candidate for balancing between source and target node if another replica is on the target node **or on the same node group of the target node.**

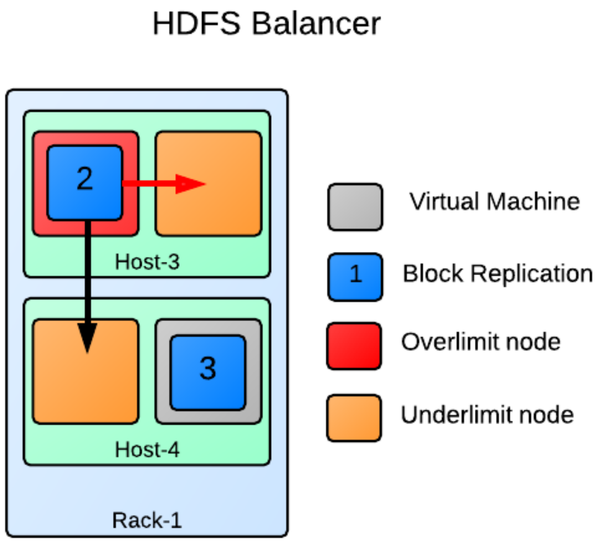

Figure 6: HDFS Balancer Policy Extension

In the case shown above, the original balancer policy without HVE has equal chance to move **replica 2** to the empty nodes in Host-3 and Host-4. Obviously, moving the block to the Host-3 node is a much better choice, which reduces network traffic and keeps the same reliability as before (avoiding placing two replicas on the same physical host). The new balancer policy with HVE will guarantee it always makes the better choice.

Task Scheduling Policies Extension

Hadoop MapReduce task scheduling policy takes data locality information into consideration in assigning Map tasks to benefit from local data access. When a MapReduce job is submitted, JobTracker splits the input data into block-sized pieces and creates tasks to work on each piece. In this process, each task is tagged with the location info of replicas. Then, when a TaskTracker requests new tasks to fill its free slots, the JobTracker will pick up a task, from the task list of a specific priority, with the nearest input data split from TaskTracker.

Under the previous task scheduling policy, JobTracker assign tasks to TaskTracker in order of: data local, rack local and off switch. But with the HVE task scheduling policy extension, the order is changed to: data local, **node group local**, rack local and off rack. The following case shows the advantage of this change.

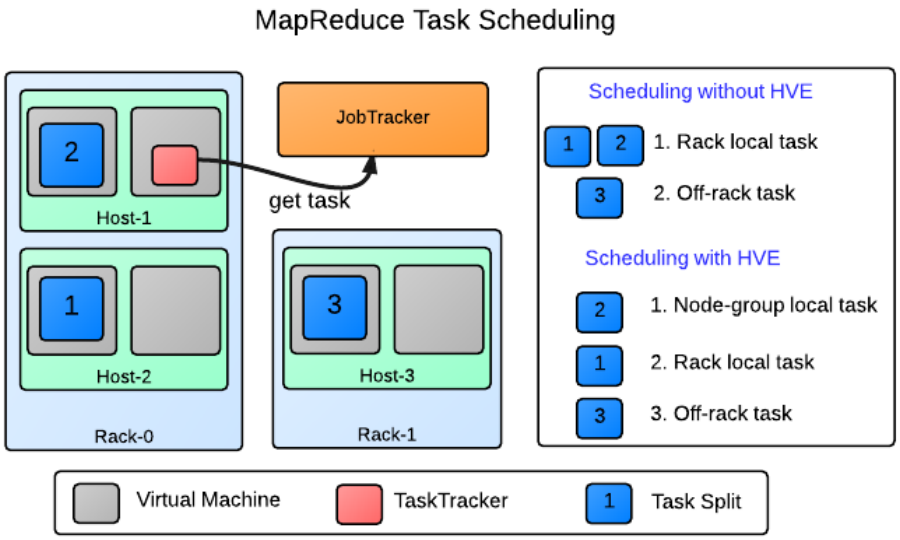

Figure 7: Task Scheduling Policy Extension

A TaskTracker with one free slot requests a task from the JobTracker and there are three task candidates, which work on different data blocks, to be scheduled. Under the previous task scheduling policy, JobTracker has equal chance to schedule task1 and task2 to TaskTracker. With the HVE task scheduling policy extension, JobTracker will always schedule task2 to this TaskTracker, which is a smart choice to enjoy physical host level data locality.

One thing need to note here is the HVE task scheduling policy extension is not a replacement of existing Hadop job schedulers but is tested to work well with existing job schedulers, including: FIFOScheduler, FairScheduler and CapacityScheduler.

Deployment options

Unlike when operating on physical hardware, there are several options for deploying Hadoop clusters on virtualized infrastructure, as discussed previously. Among them, four representative deployment options are listed below.

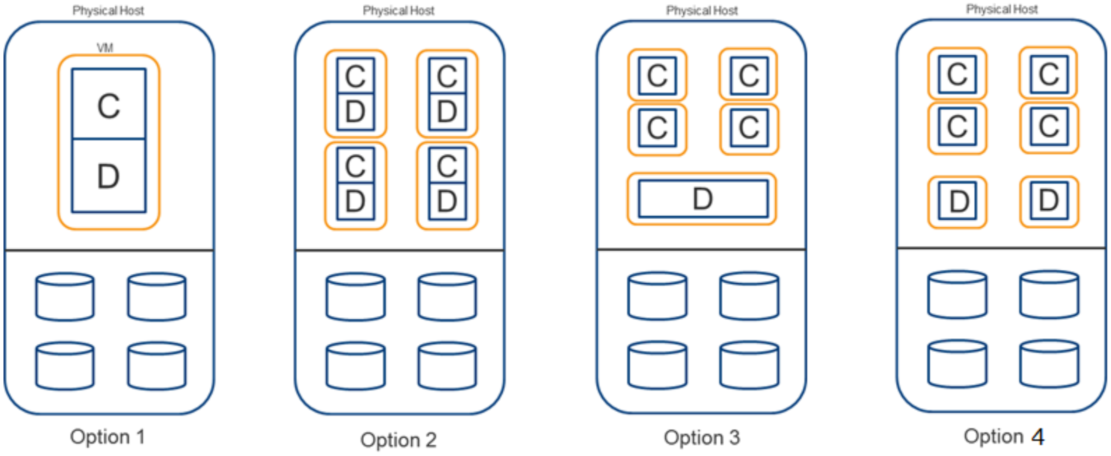

Figure 8: 4 Deployment Options

The outermost rounded rectangles represent physical hosts that can host one or more virtual machines. The inner rounded rectangles, contained within the physical hosts, represent virtual Hadoop nodes running within a virtual machine (VM) on a physical host. The “C” and “D” boxes represent Hadoop compute (TaskTracker) and data (DataNode) processes.

Option 1: one Hadoop node per host

In option 1, a Hadoop cluster is deployed on some physical hosts with equal or less than one virtual node (with compute and data processes) per host. Running a Hadoop cluster in this configuration does not require any changes to Hadoop topology or locality algorithms, but it underutilizes the virtualized infrastructure because it does not allow for elastic addition or removal of VMs. HVE has no impact on the related policies when enabled.

Option 2: multiple Hadoop nodes per host

Option 2 shows multiple virtual nodes per host, each of which is running with both compute and data processes. This is a common case when deploying Hadoop clusters on virtualized infrastructure. With HVE enabled, all locality-related policies are applied with enhancement.

Option 3: multiple compute nodes and one data node per host

Option 3 indicates the deployment of multiple virtual compute nodes and one virtual data node per host for a Hadoop cluster. The original design goal of Hadoop does not take cluster elasticity into consideration because it considers physical infrastructure as dedicated resources for the Hadoop cluster. It is expensive to add data nodes to or remove data nodes from the Hadoop cluster because both will take significant data movement traffic (or the cluster will be unbalanced or unreliable). With the decoupling of TaskTrackers and DataNodes into separated VMs, the virtual compute nodes of the Hadoop cluster can have an independent life cycle and are easily scaled in and out across the physical infrastructure according to running workloads.

This deployment enhances elasticity and improves the resource efficiency of Hadoop clusters but does not leverage the benefit from data locality as Hadoop is not aware of which virtual compute nodes are on the same host as a virtual data node. However, enabling HVE applies enhancements in the policies of task scheduling and replica choosing, enabling this deployment to achieve the same locality as non-separated cases.

Option 4: multiple compute and data nodes per host

This option is similar to the deployment in option 3 but has multiple data nodes per host. Besides elasticity in compute cluster, this deployment also shows extensibility for storage capacity. For clusters using this deployment option, with HVE enabled, all locality-related policies are applied with reliability and performance enhancements.

Summary of HVE Effect for Options 1 - 4

The below table shows which of the HVE extensions are effective (marked “X”) for each option:

	Option 1	Option 2	Option 3	Option 4
Network Topology Extension	-	X	X	X
Replica Placement Policy Extension	-	X	-	X
Replica Removal Policy Extension	-	X	-	X
Replica Choosing Policy Extension	-	X	X	X
Balancer Policy Extension	-	X	-	X
Task Scheduling Policy Extension	-	X	X	X

Table 1: Applicable enhancement of policies with enabling HVE in options 1 – 4

Based on the above analysis, we can draw a HVE impact table for virtual Hadoop clusters under options 1 – 4 as below:

	Option 1	Option 2	Option 3	Option 4
Achieve Reliability	-	X	-	X
Improve Data Locality	-	X	X	X

Table 2: HVE Reliability and Performance Effect in options 1 - 4

Physical layer between host and rack

HVE tends to support different failure and locality topologies which is primarily driven from the perspective of virtualization, however, it is also possible to use the new extensions to support other failure/locality changes, such as those relating to failures of power supplies or network, arbitrary sets of physical servers, or a set of blades in the same enclosure, etc.

Reliability Evaluation

As previously discussed, with HVE enabled, the reliability of a virtualized Hadoop cluster (in options 2 and 4) is enhanced due to the replica placement policy extension, which guarantees all replicas of a block are placed on different physical hosts. In this section, we describe a test to evaluate the effect of HVE on reliability.

Test Methodology

We provision two virtual clusters on the same physical Testbed, with each cluster having the same virtual topology (two Hadoop nodes per host) but with different configurations to enable or disable HVE. We used Greenplum HD 1.2, the first commercial distribution to support HVE, for our evaluation. When writing data to the Hadoop cluster in both cases, we list the replica placement info of each data block and check if it conforms to reliability requirements—no more than one replica on the same physical host.

In both cases, if we find any result that violates block reliability, then we perform a negative test, suddenly shutting down 2 physical hosts to simulate a catastrophic failure, and observe the effect of this failure on the Hadoop cluster.

The setup for the two clusters for reliability evaluation is as below:

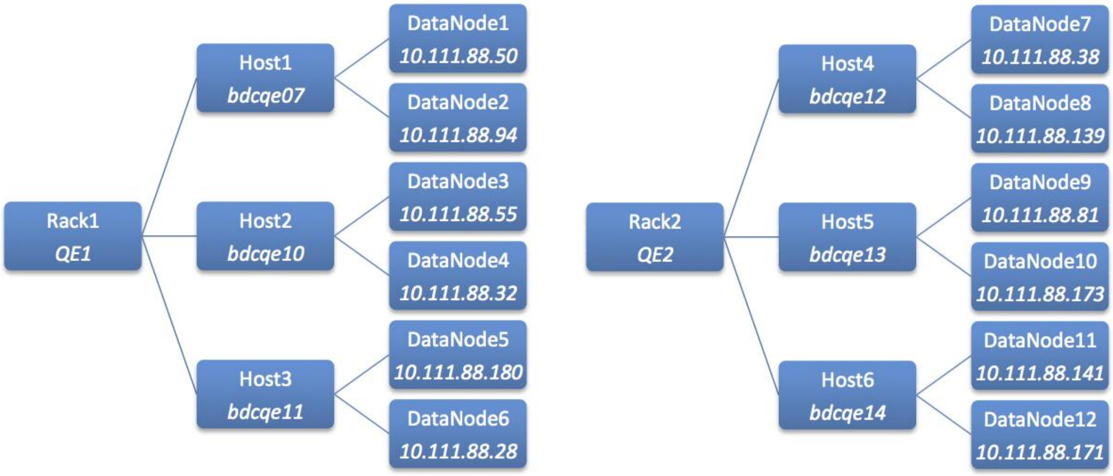

Figure 9: Cluster Setup with HVE Disabled

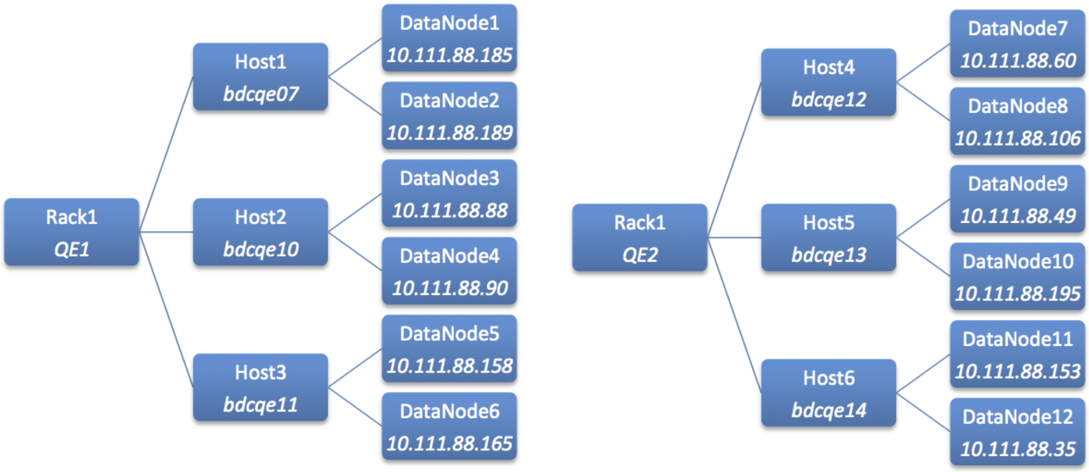

Figure 10: Cluster Setup with HVE Enabled

Each setup is a typical cluster under option 2 (please refer previous section - “Deployment Options”), with 6 physical hosts across two racks, each with 2 virtual nodes.

Test Cases and Results

Case 1: HVE disabled

On the cluster with HVE disabled, we write a file of 2G size into Hadoop cluster and list the following block replica info:

```
 [root@10 hadoop]# hadoop fsck /tmp/input -files -blocks -racks -locations
FSCK started by root from /10.111.88.111 for path /tmp/input at Wed Oct 17 09:34:05 UTC 2012
/tmp/input 2000000000 bytes, 30 block(s):  OK
0. blk_-1617183248786265922_1038 len=67108864 repl=3 [/QE1/10.111.88.94:50010, /QE1/10.111.88.180:50010,
/QE2/10.111.88.139:50010]
1. blk_1638498577860448171_1038 len=67108864 repl=3 [/QE1/10.111.88.94:50010, /QE2/10.111.88.173:50010,
/QE1/10.111.88.180:50010]
2. blk_-303252709987136463_1038 len=67108864 repl=3 [/QE1/10.111.88.94:50010, /QE1/10.111.88.28:50010,
/QE2/10.111.88.141:50010]
……
19. blk_-4419296691388858244_1038 len=67108864 repl=3 [/QE2/10.111.88.139:50010, /QE1/10.111.88.28:50010,
/QE2/10.111.88.38:50010]
20. blk_3787794892177567631_1038 len=67108864 repl=3 [/QE2/10.111.88.173:50010, /QE2/10.111.88.81:50010,
/QE1/10.111.88.180:50010]
……
```

Checking all 30 blocks, we identify several blocks that are not robustly replicated and utilize 2 physical hosts only. The block marked in bold is an example: according to previous setup figure, VMs 10.111.88.139 and 10.111.88.38 are on the same physical host, which means the file would be corrupt if we shut down these 2 specific hosts. The following test result confirms our expectation after we shutdown host3 and host4, wait, and check the health of the file.

```
[root@10 hadoop]# hadoop fsck /tmp/input
FSCK started by root from /10.111.88.111 for path /tmp/input at Wed Oct 17 10:03:41 UTC 2012
.
/tmp/input: CORRUPT block blk_-4419296691388858244
```

```
/tmp/input:  Replica placement policy is violated for blk_-4419296691388858244_1038. Block should be additionally replicated on 2
more rack(s).

/tmp/input: MISSING 1 blocks of total size 67108864 B.Status: CORRUPT
 Total size:    2000000000 B
 Total dirs:    0
 Total files:   1
 Total blocks (validated):    30 (avg. block size 66666666 B)
  ********************************
  CORRUPT FILES:    1
  MISSING BLOCKS:   1
  MISSING SIZE:         67108864 B
  CORRUPT BLOCKS:       1
  ********************************
 Minimally replicated blocks:     29 (96.666664 %)
 Over-replicated blocks:     0 (0.0 %)
 Under-replicated blocks:    0 (0.0 %)
 Mis-replicated blocks:      1 (3.3333333 %)
 Default replication factor:  3
 Average block replication: 2.9
 Corrupt blocks:          1
 Missing replicas:        0 (0.0 %)
 Number of data-nodes:            8
 Number of racks:       2
FSCK ended at Wed Oct 17 10:03:41 UTC 2012 in 3 milliseconds


The filesystem under path '/tmp/input' is CORRUPT
```

Case 2: HVE enabled

On the cluster with HVE enabled, we repeat the same operation to write a 2G file into the Hadoop cluster and list the block replica info:

```
/tmp/input 2000000000 bytes, 30 block(s):  OK
……
0. blk_4957178687966820185_1017 len=67108864 repl=3 [/QE2/sin2-pekaurora-bdcqe013.eng.vmware.com/10.111.88.49:50010,
/QE2/sin2-pekaurora-bdcqe014.eng.vmware.com/10.111.88.35:50010, /QE1/sin2-pekaurora-
bdcqe011.eng.vmware.com/10.111.88.165:50010]
1. blk_2444195495823050935_1017 len=67108864 repl=3 [/QE2/sin2-pekaurora-bdcqe012.eng.vmware.com/10.111.88.60:50010,
/QE2/sin2-pekaurora-bdcqe013.eng.vmware.com/10.111.88.195:50010, /QE1/sin2-pekaurora-
bdcqe011.eng.vmware.com/10.111.88.165:50010]
2. blk_-7154786734574204306_1017 len=67108864 repl=3 [/QE2/sin2-pekaurora-bdcqe014.eng.vmware.com/10.111.88.153:50010,
/QE2/sin2-pekaurora-bdcqe012.eng.vmware.com/10.111.88.60:50010, /QE1/sin2-pekaurora-
bdcqe011.eng.vmware.com/10.111.88.165:50010]
3. blk_-4131295536131824040_1017 len=67108864 repl=3 [/QE2/sin2-pekaurora-bdcqe013.eng.vmware.com/10.111.88.49:50010,
/QE2/sin2-pekaurora-bdcqe012.eng.vmware.com/10.111.88.106:50010, /QE1/sin2-pekaurora-
bdcqe011.eng.vmware.com/10.111.88.165:50010]
4. blk_-4710113200037022226_1017 len=67108864 repl=3 [/QE2/sin2-pekaurora-bdcqe012.eng.vmware.com/10.111.88.60:50010,
/QE2/sin2-pekaurora-bdcqe014.eng.vmware.com/10.111.88.35:50010, /QE1/sin2-pekaurora-
bdcqe011.eng.vmware.com/10.111.88.165:50010]
……
```

Checking all 30 blocks, we don’t see any block that has reliability issues, and replicas of each block are distributed across 3 node groups (physical hosts) and 2 racks. We randomly perform a shutdown at the same time, and the file status remains healthy:

```
[root@10 sdc1]# hadoop fsck /tmp/input
FSCK started by root from /10.111.88.165 for path /tmp/input at Tue Oct 16 14:49:34 UTC 2012.
/tmp/input:  Under replicated blk_4957178687966820185_1017. Target Replicas is 3 but found 1 replica(s).
/tmp/input:  Replica placement policy is violated for blk_4957178687966820185_1017. Block should be additionally replicated on 1
more rack(s).
/tmp/input:  Under replicated blk_2444195495823050935_1017. Target Replicas is 3 but found 2 replica(s).
/tmp/input:  Under replicated blk_8535899477350651788_1017. Target Replicas is 3 but found 2 replica(s).
/tmp/input:  Under replicated blk_5179780317649805927_1017. Target Replicas is 3 but found 1 replica(s).
/tmp/input:  Replica placement policy is violated for blk_5179780317649805927_1017. Block should be additionally replicated on 1
more rack(s).
/tmp/input:  Under replicated blk_-3418140709046928885_1017. Target Replicas is 3 but found 1 replica(s).
/tmp/input:  Replica placement policy is violated for blk_-3418140709046928885_1017. Block should be additionally replicated on 1
more rack(s).
/tmp/input:  Under replicated blk_1375172523958903407_1017. Target Replicas is 3 but found 2 replica(s).
/tmp/input:  Under replicated blk_-5759774566150740558_1017. Target Replicas is 3 but found 2 replica(s).
Status: HEALTHY
 Total size:    2000000000 B
 Total dirs:    0
 Total files:   1
 Total blocks (validated):      30 (avg. block size 66666666 B)
 Minimally replicated blocks:   30 (100.0 %)
 Over-replicated blocks:        0 (0.0 %)
 Under-replicated blocks:       7 (23.333334 %)
 Mis-replicated blocks:         3 (10.0 %)
 Default replication factor:    3
 Average block replication:     2.6666667
 Corrupt blocks:                0
 Missing replicas:              10 (12.5 %)
 Number of data-nodes:          8
 Number of racks:               2
FSCK ended at Tue Oct 16 14:49:34 UTC 2012 in 4 milliseconds
The filesystem under path '/tmp/input' is HEALTHY
```

Results Summary

From the above test, we demonstrate that HVE enhances the reliability of virtualized Hadoop clusters under option 2 (multiple nodes per host), which meets our previous expectation. It is also straightforward to deduce the same reliability impact for HVE on a virtual Hadoop cluster under option 4 (multiple separated compute and data nodes per host).

Data Locality Evaluation

Some studies were conducted to evaluate the impact HVE had on data locality in different cases, including: option 2 – multiple combined data-compute nodes per host (labeled the “normal” case) and option 3 – multiple compute nodes and one data node per host (labeled the “data/compute separation” case). For each test case, the following benchmarks were tested with HVE enabled and disabled for comparison: Terasort and TestDFSIO. Again, Greenplum HD 1.2, with support for HVE, was used for our evaluation.

This evaluation is meant to demonstrate the tangible effect HVE has on improving data locality in cases where multiple Hadoop nodes exist on each physical host. It does not make any comparisons against running Hadoop on physical hardware or in the case where a single Hadoop VM is deployed per physical host; HVE has no impact on these deployment models.

Test Scenarios and Benchmarks

The performance test consists of 2 test scenarios: normal setup and data/compute separation.

Normal setup

In this scenario, input data for the test are distributed evenly throughout the clusters, and jobs are expected to read input data locally. This is the classic MapReduce scenario on physical hardware, where most map tasks get input from the local Datanode. Because of the high locality rate, it is expected that HVE will have less impact on performance when enabled.

Data/Compute separation

In this scenario, Datanode and TaskTrackers are separated into different VMs but still in the same physical host. There is no data locality in this case. If HVE is disabled, all tasks are expected to be scheduled randomly and will read input from random sources. But if HVE is enabled, most tasks are expected to get node-group locality and read input data block from nearest source.

In each scenario, two standard Hadoop benchmarks, TestDFSIO and Terasort, will be tested.

Terasort

Terasort is a well known benchmark which sorts a large number of records. Due to resource situation in this cluster, input data size is scaled down to 100GB rather than 1TB. Teragen is used to generate 100GB of data with a replication factor of 3 which has 400 map tasks, each writing a 250MB file. Terasort is then run to sort the input data. The Terasort job has 400 map tasks and 200 reduce tasks.

TestDFSIO

TestDFSIO is used to test HDFS throughput, it consists of 2 tests TestDFSIO-write and TestDFSIO-read. Just like Terasort, input data size is also scaled down to 100GB. We first write 100GB of data, run TestDFSIO-write with 400 map tasks, with each task writing a 250MB file, and then run TestDFSIO-read with 400 map tasks, with each task reading a 250MB file.

Test Results

Test DFSIO Read Throughput

The following chart shows the HDFS read throughput number generated by TestDFSIO-Read, normalized to 1 for the HVE Disabled case.

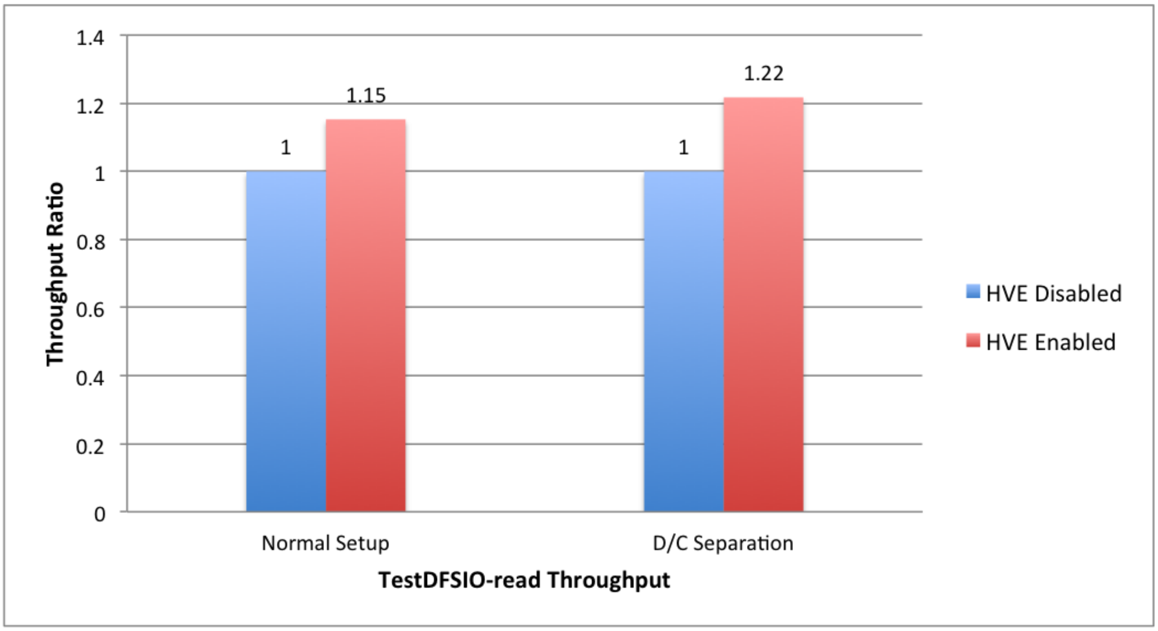

Figure 11: TestDFSIO-read Throughput (Higher is Better)

In both scenarios, HDFS read throughput is higher when HVE is enabled. HVE is 15% and 22% better in the Normal and D/C Separation test scenarios, respectively. This result shows that by leveraging additional information from virtualized environment, HVE can choose faster read paths, thus enjoy better data locality, especially for the Data/Compute Separation environment.

Terasort Locality Count

The following table shows the data locality statistics of Terasort jobs in all test scenarios.

Terasort locality	Data Local	Node-group Local	Rack Local
Normal	392	-	8
Normal with HVE	397	2	1
D/C separation	0	-	400
D/C separation with HVE	0	400	0

Table 3: Terasort Task Locality

The above statistics show that by adding a new scheduler locality layer of node-group, the HVE-enabled scheduler can do a better job at scheduling computation near the data, which is important for MapReduce jobs.

Conclusion

Virtualization brings many benefits to Hadoop, including ease of operation, high availability, elasticity and multi-tenancy. The elastic and multi-tenant aspects of virtualized Hadoop are derived from the deployment flexibility available to operators in a virtual environment. Separating the data and compute components of Hadoop enables the easy and rapid scaling of compute nodes on demand and the ability to allocate different virtual compute clusters to tenants.

HVE is a critical feature that makes data-compute separation and the deployment of multiple nodes per physical host possible by achieving reliability against data loss and ensuring data locality is fully exploited, even under these alternative deployment models. HVE, in conjunction with data-compute separation, makes elastic, reliable, performant Hadoop clusters a reality for users wishing to leverage virtualization for Hadoop.

References

1. Virtualizing Hadoop: http://www.vmware.com/hadoop

2. A Benchmarking Case Study of Virtualized Hadoop Performnace on VMware vSphere 5. http://www.vmware.com/files/pdf/VMW-Hadoop-Performance-vSphere5.pdf

3. Apache Hadoop Umbrella JIRA for HVE: https://issues.apache.org/jira/browse/HADOOP-8468. Apache Hadoop JIRA for rack awareness: https://issues.apache.org/jira/browse/HADOOP-692.