Deploying Virtualized Hadoop® Systems with VMware vSphere® Big Data Extensions™

A DEPLOYMENT GUIDE
Table of Contents

Introduction ................................................................. 4
Overview of Hadoop, vSphere, and Project Serengeti ......................... 4
  An Overview of Hadoop ............................................. 4
  VMware vSphere ...................................................... 6
  Why Virtualize Hadoop? ........................................... 6
  Open-Source Work in the Hadoop Arena ................................ 8
  Project Serengeti .................................................... 8
  VMware vSphere Big Data Extensions .................................. 8
  The Big Data Extensions Architecture and Deployment Process ........... 9
Deploying Hadoop Clusters with Big Data Extensions - An Overview .......... 10
  vSphere Big Data Extensions Requirements ............................. 10
  Deploying the Big Data Extensions Virtual Appliance .................... 11
  Using the Serengeti Command-Line Interface Client .................... 11
  Deploying a Customized Hadoop Cluster .................................. 11
Reference Architecture ....................................................... 12
  Understanding the Application Requirements ........................... 12
  Deployment Models for Hadoop Clusters ................................ 13
    The Combined Model .............................................. 13
    The Separated Model ............................................. 15
  Network Effects of the Data–Compute Separation ....................... 16
  A Shared-Storage Approach to the Data–Compute Separated Model ...... 18
  Local Disks for the Application’s Temporary Data ...................... 19
  Virtual Machines, VMDKs, Datastores, and Local Disks.................... 20
  A Network-Attached Storage (NAS) Architecture Model .................. 20
  Deployment Models Summary ........................................ 21
Storage Sizing ..................................................................... 22
Availability ......................................................................... 22
Hardware Considerations ..................................................... 24
Systems Architecture Examples ............................................. 25
  Example Configurations .............................................. 25
Deploying Virtualized Hadoop Systems with VMware vSphere Big Data Extensions

Best Practices .......................................................... 28
  Big Data Extensions Best Practices .................................. 28
  Changing the Guest Operating System of the Big Data Extensions Template ... 28
  Upgrading the Virtual Hardware in the Big Data Extensions Template ........ 28
  Warm Up the Disks After Provisioning a Hadoop Cluster with Big Data Extensions ........................................... 30
  Determining the Disk Format to Use .................................. 30

General Best Practices for Hadoop on vSphere .......................... 31
  Disk I/O. .................................................................. 31
  Sizing the Data Space .................................................. 31
  Create Aligned Partitions on Disk Devices ............................ 32
  Virtual CPUs ............................................................ 32
  Memory ................................................................. 33
  Networking ............................................................... 33
  Details on Virtualized Networking ..................................... 33
  Virtual Distributed Switch ............................................. 34
  Examples of Various Networking Configurations .................... 34
  Customizing the Hadoop Configuration Parameters .................. 35

Conclusion ............................................................... 36
About the Authors .......................................................... 36
Appendix A: Big Data Extensions
  Configuration File – Example ........................................... 37
Appendix B: Hadoop Configuration for a Sample Test .................... 39
References ................................................................. 41
  VMware Resources ....................................................... 41
  Hadoop Resources ...................................................... 42
  Networking Resources .................................................. 42
Introduction

This document provides an introduction to the concepts contained in the VMware vSphere® Big Data Extensions™ technology. Big Data Extensions is part of the vSphere product family as of vSphere 5.5 and can also be used with installations of vSphere 5.0. This document also provides a set of different deployment patterns and reference architectures that adopters of Big Data Extensions can implement to use the technology in their own Hadoop® work.

Big Data Extensions provides an integrated set of management tools to help enterprises deploy, run, and manage Hadoop workloads executing in virtual machines on a common infrastructure. Through the VMware vCenter™ user interface, users can manage and scale Hadoop clusters more easily. Big Data Extensions complements and integrates with the vSphere platform through use of a vCenter plug-in and a separate VMware Serengeti™ Management Server running within a virtual appliance.

This document can be used as a starting point for a new installation or for rearchitecting an existing environment. The examples given here can be adapted to suit the environment, based on requirements and the available resources.

In this document, we confine the scope of discussion to versions of Hadoop preceding the 2.0 version, so Hadoop 2.0 YARN technology is not covered here.

This document addresses the following topics:

• An overview of the Hadoop and Big Data Extensions technologies
• Considerations for deploying Hadoop on vSphere with Big Data Extensions
• Architecture and configuration of Hadoop systems on vSphere

Overview of Hadoop, vSphere, and Project Serengeti

This section presents a primer on Hadoop and on VMware® virtualization technologies to readers who are unfamiliar with them. This information will support the more detailed deployment discussions in later sections of the document.

An Overview of Hadoop

Hadoop originated in work done by engineers at Google and later at Yahoo to solve problems involving data on a very large scale—Web indexing and searching in particular. It later was contributed to the Apache community as an open-source project, in which development continues. Apache Hadoop is a programming environment with a software library and a runtime that together provide a framework for reliable and scalable distributed computing. Along with the distributions available directly from Apache, there are several commercial companies that provide their own supported distributions based on Apache, including Cloudera, HortonWorks, Intel, MapR, and Pivotal. These are often integrated with other components, both open source and proprietary. In this document, we will refer to all such implementations as Hadoop.

All of the previously mentioned Hadoop distributions can be deployed using Big Data Extensions and run well in virtual machines on vSphere. Examples of testing done on a Hadoop distribution when deployed on vSphere are given in [4] and [5]. VMware works with the distributors to provide validation and support for their Hadoop products on vSphere.

Hadoop typically is used for processing large data sets across clusters of independent machines. Hadoop clusters can scale up to thousands of machines, each participating in computation as well as file and data storage. Hadoop is adopted by companies for a wide range of custom-built and packaged applications that are designed to help businesses make better decisions through analysis of the data sets. Hadoop has become one of the leading “big data” platforms for storing large quantities of unstructured data, supporting a range of extract, transform, and load (ETL) tools, MapReduce, and other analytic functions.
Hadoop 1.0 systems primarily comprise two basic components: the Hadoop Distributed File System (HDFS), used to store data, and the MapReduce computational framework. Other technologies that use Hadoop as a basis—Hive™, Pig, YARN, and others—are outside the scope of this document.

The Hadoop MapReduce component uses a “divide and conquer” approach in processing large amounts of data to produce a set of results. MapReduce programs are inherently parallel in nature, so they are well suited to a highly distributed environment. In the Hadoop 1.0 architecture, the main roles within Hadoop are the JobTracker, TaskTracker, NameNode, and DataNode.

A JobTracker process schedules all the jobs in the Hadoop cluster, as well as individual tasks that make up the job. A job is split into a set of tasks that execute on the worker nodes. A TaskTracker process, which runs on each worker node, is responsible for starting tasks on its node and reporting progress to the JobTracker.

HDFS is managed by one or more NameNodes that manage the file system namespace and data block placement. The NameNode is the master process for HDFS and works in conjunction with a set of DataNodes that place blocks of data onto storage and retrieve them. HDFS manages multiple replicas of each block of data, providing resiliency against failure of nodes in the cluster. By default, three replicas are kept, but the administrator can change this number.

The NameNode manages the file system namespace by maintaining a mapping of the filenames and blocks of the file system onto all of the DataNodes. The machines that run the JobTracker and NameNode processes are the “master” nodes of the cluster; all others are the “worker” nodes.

The first generation of Hadoop had two single points of failure: the NameNode and JobTracker processes. This has been addressed in later versions of the first release and in the Apache Hadoop 2.x YARN technology. In some Hadoop 1.0 deployments, there is a secondary NameNode that keeps snapshots of the NameNode metadata, to be used for recovery in the event of a primary NameNode failure.

A small Hadoop cluster includes a single master node and multiple worker nodes, as is shown in Figure 1.

![Figure 1. Major Components of the Hadoop Architecture - Contains a Single Master Node and Multiple Worker Nodes](image-url)
In Figure 1, the master node contains JobTracker and NameNode as the main Hadoop management roles, with a TaskTracker and DataNode as optional extra roles within the master node. The worker node contains the DataNode and TaskTracker roles on a server, though it is also possible to have data-only worker nodes and compute-only worker nodes.

In a native implementation of a larger Hadoop cluster, the HDFS can be managed by a NameNode on a dedicated server to host the file system index. A standalone JobTracker on a server can also be used to manage job scheduling, resource allocation, and other functions.

In some Hadoop cluster types, such as those in which the Hadoop MapReduce operation can be deployed against an alternative to HDFS, the NameNode, secondary NameNode, and DataNode architecture components of HDFS are replaced by the equivalent components that the alternative system provides. This can be an NFS server in certain distributions.

**VMware vSphere**

VMware vSphere has helped organizations optimize their IT infrastructure through consolidation while providing superior availability, scalability, and security. vSphere 5 enables virtualization of the entire IT infrastructure including servers, storage, and networks.

Why Virtualize Hadoop?

The following are among the many benefits of virtualizing Hadoop on vSphere:

- **Rapid provisioning:** vSphere Big Data Extensions enables rapid deployment, management, and scalability of Hadoop in virtual and cloud environments. Virtualization tools ranging from simple cloning to sophisticated end-user provisioning products such as VMware vCloud Automation Center™ can speed up the deployment of Hadoop, which in most cases requires multiple nodes with different configurations in them. A virtualized infrastructure with the automated elasticity features of Big Data Extensions tools enables on-demand Hadoop instances.
• **Better resource utilization:** Colocating virtual machines containing Hadoop roles with virtual machines containing different workloads on the same set of VMware ESXi™ server hosts can balance the use of the system. This often enables better overall utilization by consolidating applications that either use different kinds of resources or use the same resources at different times of the day.

• **Alternative storage options:** Originally, Hadoop was developed with local storage in mind, and this type of storage scheme can be used with vSphere also. The shared storage that is frequently used as a basis for vSphere can also be leveraged for Hadoop workloads. For smaller clusters, the data and compute tiers that make up a Hadoop application can be held entirely on shared storage. As larger clusters are built, a hybrid storage model applies. The hybrid model is one in which parts of the Hadoop infrastructure use direct-attached storage and other parts use storage area network (SAN)-type storage. The NameNode and JobTracker use negligible storage resources and can be placed on SAN storage for reliability purposes. When the goal is to achieve higher utilization, a reasonable approach is to put the temporary data on locally attached storage and to place HDFS data on local or on shared storage. With either of these configurations, the unused shared-storage capacity and bandwidth within the virtual infrastructure can be given to Hadoop jobs.

• **Isolation:** This includes running different versions of Hadoop itself on the same cluster or running Hadoop alongside other applications, forming an elastic environment, or different Hadoop tenants.

• **Availability and fault tolerance:** The NameNode, the JobTracker, and other Hadoop components, such as Hive Metastore and HCatalog, can be single points of failure in a system. vSphere services such as VMware vSphere High Availability (vSphere HA) and VMware vSphere Fault Tolerance (vSphere FT) can protect these components from server failure and improve availability. Resource management tools such as VMware vSphere vMotion® and VMware vSphere Distributed Resource Scheduler™ (vSphere DRS) can provide availability during planned maintenance and can be used to balance the load across the vSphere cluster.

• **Efficiency:** VMware enables easy and efficient deployment and use of Hadoop on an existing virtual infrastructure as well as consolidation of otherwise dedicated Hadoop cluster hardware into a data center or cloud environment, as is shown in Figure 3.

---

**Figure 3.** Virtual Data Center Deployment Including Hadoop Clusters
Open-Source Work in the Hadoop Arena

VMware has made several open-source contributions related to Hadoop:

1. Hadoop Virtualization Extensions (HVE) [11] provides an awareness of the virtualization topology—as well as the hardware topology—as an addition to the management components of Hadoop. The HVE facilities for optimal task placement and for data block placement and retrieval help toward achieving better Hadoop reliability and performance in virtual environments.

2. Project Serengeti™ provided the foundational work for vSphere Big Data Extensions. By simplifying the configuration and deployment while supporting distributions from multiple vendors and their customizations, it offers tools for the creation, sizing, and management of Hadoop clusters and nodes in virtual machines.

Project Serengeti

Project Serengeti [6] is an open-source initiative by VMware to automate the deployment and management of Hadoop clusters in vSphere environments.

Serengeti is delivered as a virtual appliance—that is, a set of virtual machines that can be instantiated together. The Serengeti virtual appliance contains a deployment toolkit and management service that can be used to configure and deploy Hadoop clusters and nodes in a vSphere environment. With Serengeti, users can deploy a highly available Hadoop cluster on a virtual platform in minutes, including the common Hadoop components such as HDFS, MapReduce, Pig, and Hive. Serengeti supports multiple Hadoop distributions such as those available from Apache, Cloudera, HortonWorks, Intel, MapR, and Pivotal.

Figure 4. Serengeti Toolkit and Management Services

VMware vSphere Big Data Extensions

Big Data Extensions is a vSphere feature that expands the platform to support big-data and Hadoop workloads. It is the supported enterprise version of Project Serengeti, providing an integrated set of management tools to help users deploy, run, and manage Hadoop on a common infrastructure. Through the vCenter user interface, Big Data Extensions users can manage and scale Hadoop clusters on the vSphere platform. For more detailed information, go to the Big Data Extensions site [2].
The Big Data Extensions Architecture and Deployment Process

The outline architecture for vSphere Big Data Extensions is shown in Figure 5. The two major components of Big Data Extensions are shown: the template and the Serengeti Management Server. The template is used to create virtual machines that contain different Hadoop roles. The Serengeti Management Server is responsible both for configuration of new virtual machines and for monitoring the Hadoop system after it has been set up and started.

The Serengeti Management Server makes requests to VMware vCenter Server™ to carry out various actions, such as instantiating a virtual machine from the template. Both the Serengeti Management Server and the template are contained in a virtual appliance that is installed when a user imports the Big Data Extensions Open Virtualization Architecture (OVA) file, in which Big Data Extensions is delivered, into vCenter Server. When started, the Serengeti Management Server must be securely connected to vCenter Server so it can carry out its requests. The template contains an agent process that starts in the guest operating system (OS) when a virtual machine conforming to that template is started up. This agent is used for customization of the guest OS as well as for installation and configuration of the Hadoop role that the virtual machine supports.

![Figure 5. The Big Data Extensions Architecture and Deployment Process](image)

Big Data Extensions performs the following steps to deploy a Hadoop cluster:

1. The Serengeti Management Server searches for ESXi hosts with sufficient resources.
2. The Serengeti Management Server selects one or more ESXi hosts on which to place the Hadoop virtual machines.
3. The Serengeti Management Server sends a request to vCenter Server to clone from the supplied template and to reconfigure the resulting virtual machine.
4. The prebuilt agent within the guest OS of the newly cloned virtual machine configures the OS parameters and network configurations.
5. The agent downloads the Hadoop software packages from a repository that has been identified to the Serengeti Management Server.
6. The agent installs the Hadoop software.
7. The agent configures the Hadoop parameters.

These provisioning steps can be performed in parallel across several newly created virtual machines, reducing deployment time.
Deploying Hadoop Clusters with Big Data Extensions – An Overview

For the complete description of the process of building a Hadoop cluster with Big Data Extensions, consult the VMware vSphere Big Data Extensions Administrator’s and User’s Guide [1]. This section provides an outline of the process. As part of the planning for the deployment of a Hadoop cluster with Big Data Extensions, ensure that the following system prerequisites are met. Have good estimates of the disk space and memory space needed for each component of the proposed Hadoop cluster.

vSphere Big Data Extensions Requirements

Before using Big Data Extensions for deployment of a Hadoop cluster, check against the following requirements:

• An installation of VMware vSphere 5.1 or later with VMware vSphere Enterprise Edition™ or VMware vSphere Enterprise Plus Edition™ level licensing
• A compatible vCenter installation to manage the vSphere host servers
• VMware vSphere Web Client 5.1 or later
• A vSphere DRS cluster with a dedicated resource pool for the Hadoop cluster
• Network Time Protocol (NTP) configured and enabled for time synchronization on all ESXi hosts in the cluster
• The following required resources for the Serengeti Management Server and other components
  - 28GB RAM
  - A port group having six uplink ports with connectivity to the Hadoop clusters
  - Connectivity between the network used by the ESXi hosts and the network used by the management server
• Serengeti Management Server with sufficient resources for all virtual machines deployed in the resource pool created in vSphere
• Carefully calculated required datastore space
  - For data stored in HDFS
  - For data stored outside of HDFS (“temporary data” used between phases of the MapReduce algorithms)

HDFS stores the input data to the application and the output data from the application. Calculate the minimum data space required by multiplying the input and output data sizes by their respective numbers of replicas and adding the temporary space needed. The number of data replicas can be different for the input and output data and different for separate applications. The size of the temporary data can be larger than the size of the input or output data. The temporary data most often is stored in a storage area separate from that containing the HDFS data, so they must be sized separately.

• No swapping should occur at either the ESXi level or the guest OS level. To avoid swapping at the ESXi level, ensure that there is adequate physical memory for the demands of all virtual machines on the ESXi host server as well as of the hypervisor itself. For a full discussion of this topic, see the vSphere Resource Management Guide [15].
• To avoid swapping at the guest OS level—within the virtual machine—ensure that the virtual machine is configured with enough memory for all its resident processes and OS needs.
Deploying the Big Data Extensions Virtual Appliance

The components of Big Data Extensions are shipped in a virtual appliance form that is contained in an OVA file, which is an industry standard format for such deliverables. Download the OVA file for the Big Data Extensions virtual appliance from the VMware Hadoop Web site: http://www.vmware.com/hadoop. After downloading the Big Data Extensions OVA file, complete the following steps:

1. Start vSphere Client.
2. Select the File > Deploy OVF Template option to identify the OVA file.
3. Install the Big Data Extensions virtual appliance. When installing it, select a resource pool at the top level of the vSphere Host/Cluster hierarchy.

Download the VMware vSphere Big Data Extensions Administrator’s and User’s Guide from the VMware Hadoop Web site. Follow the instructions provided in the guide to set the Big Data Extensions server networking and resource options.

Using the Serengeti Command-Line Interface Client

After installing the Big Data Extensions appliance, use it through the vSphere Web Client GUI or log in to the Serengeti Management Server virtual machine from an SSH session window—or from the Serengeti Remote Command-Line Interface (CLI) Client—and then run the Big Data Extensions command to enter the Big Data Extensions command shell:

$ BDE

The Big Data Extensions cluster create command quickly creates and deploys a default Hadoop cluster:

BDE>cluster create --name myHadoop

This one command creates a Hadoop cluster with one master node virtual machine, three worker node virtual machines, and one client node virtual machine. The client node virtual machines contain a Hadoop client environment including the Hadoop client shell, Pig, and Hive.

After the deployment is complete, view the IP addresses of the Hadoop node virtual machines. By default, Big Data Extensions can use any of the resources added to its list to deploy a Hadoop cluster. To limit the scope of resources for the cluster, specify resource pools, datastores, or a network in the cluster create command:

BDE>cluster create --name myHadoop --rpNames myRP --dsNames myDS --networkName myNW

The --rpNames parameter identifies the resource pools to use. The value supplied to the --dsNames parameter tells it which data stores to use for creation of the Hadoop cluster.

After the Hadoop cluster has been created, run additional commands, such as listing the Hadoop clusters deployed by Big Data Extensions, running a sample Hadoop MapReduce job, using Pig, and running the Hive utilities.

Deploying a Customized Hadoop Cluster

Big Data Extensions also provides the ability to completely customize a Hadoop cluster by specifying the details of the deployment in a Hadoop cluster specification file. This is a JSON-formatted text file in which users can specify a set of Big Data Extensions options to control the creation of a new Hadoop cluster and the vSphere resources that the cluster is to use. An example of a cluster specification file is provided in Appendix B. After creating the cluster specification file, ask Big Data Extensions to provision the Hadoop cluster according to its contents, using the cluster create command, as shown in the following example:

BDE>cluster create --name myHadoop --specFile /home/BDE/mySpec.txt
The `--specFile` parameter provides the name of the cluster specification file that the user has created for their custom design. Using a Big Data Extensions specification file, specify the precise makeup of all the nodes in a cluster as well as the exact type and size of resources the cluster and individual nodes are assigned. Examples of this include the number of virtual CPUs for any virtual machine, whether the storage type being used is local or shared, and the configured networks to be used.

**NOTE:** Refer to the Big Data Extensions Administrator’s and User’s Guide for more information on available commands and options as well as how to fully customize a Hadoop cluster using a Big Data Extensions specification file.

### Reference Architecture

This section focuses on technical considerations for virtualizing a Hadoop cluster and provides a set of architecture choices for building various sizes of Hadoop systems. This is not intended to provide a prescriptive reference architecture, because the various applications built on Hadoop differ, perhaps widely, in how they consume the resources provided.

That is why it is important to estimate the resource consumption of an existing or a proposed Hadoop cluster. This provides the information to size the system correctly—an essential part of the architecture process. Further technical details on the various deployment models discussed here can be found in [4] and [5].

A deployer of a Hadoop-based system might be required to fit the new system into an existing architecture in the data center and as a result have certain constraints such as preexisting storage considerations. We address some of those issues here also.

When virtualized, these Hadoop NameNode, JobTracker, DataNode, and TaskTracker “roles” are processes in the guest OSs of a set of virtual machines, sometimes referred to as daemons. Some of these virtual machines might be executing on the same vSphere server host. This requires examining the topology of a system in a different way. One such new consideration concerns whether the replicas of data blocks in HDFS might be placed on two virtual machines within the same server host.

This section consists of two parts. The first part describes various mappings of Hadoop roles into virtual machines. A number of storage approaches are also discussed here, including locally attached storage, SAN-based storage, and network-attached storage (NAS). Consideration is given to determining the best trade-off between scalability, performance, and costs when choosing a storage strategy.

The second part discusses system configuration, including storage sizing factors, availability, hardware choices, and network infrastructure, as a basis for forming a reference architecture. Options are presented here, along with their respective trade-offs, because no single architecture addresses every problem.

### Understanding the Application Requirements

To build a suitable Hadoop cluster, first accumulate data about the types of applications that will be using it. This requires conversations with the application design and development teams or with the vendor of the application software. The following are among the types of required data:

- The amount of input data that will be ingested into the Hadoop cluster
- The expected growth rate of that data over time
- The amount of temporary data that will be required to be stored between the stages of the MapReduce processing
- The amount of data that will be written back to HDFS
- The throughput and bandwidth required for read and write of all data (Mb/second read and written)
- The replication factor of the data (how many times the data blocks will be copied and stored)
- How large the data blocks should be
• How the application’s performance characteristics are expected to compare with known CPU-intensive, I/O-intensive, or network-intensive applications such as the TeraGen, TeraSort, TeraValidate benchmark applications
• The mix of applications in the cluster, with the information gathered as previously listed for each application
• The priorities of various applications
• Multitenancy and how resources must be allocated to each tenant
• The elasticity requirements of the system

After this data has been collected, a basis exists for making design decisions.

Deployment Models for Hadoop Clusters

This section presents a set of example layouts for deployment of the main Hadoop components and a set of basic principles for choosing between them. All these models are deployable using vSphere Big Data Extensions. Figure 6 outlines the topology choices. Here, the term “compute” refers to the TaskTracker process in the Hadoop architecture; the term “data” refers to the DataNode process. We assume for now that the NameNode and JobTracker are held in separate virtual machines elsewhere in the system, perhaps on dedicated servers or on virtual machines of their own.

![Figure 6. Choices for Deployment of the Hadoop Roles on Virtual Machines](image)

In options 1 and 2 of Figure 6, the compute and data roles are combined into one virtual machine. In options 3 and 4, they are separated into individual virtual machines. These models have different benefits, which will be discussed in the following sections along with a review of architectural variants of them.

The Combined Model

In this approach to the architecture, the TaskTracker and DataNode Hadoop roles are combined in a virtual machine. This is the traditional, or “standard,” approach and was used in the performance tests described in [4] and [5]. However, as will be discussed later, it is not the only model to consider. Figure 7 shows a deployment of the combined model. For the purposes of illustration, we show one worker virtual machine (“Hadoop virtual node”) on the host ESXi server. There can be many such virtual machines running together on a vSphere host server.
For the purposes of this example, we show a vSphere host server that has eight local disks as well as a connection to a SAN or NAS shared-storage device. If no local disk storage is available, there are alternatives for deploying the model, which will be discussed later in this section.

In our example, there are eight VMDK files present on the eight locally attached disks, with one VMDK per disk. A deployment can include more local disks, as available. The TaskTracker and DataNode processes share the disks. We place the guest OS disk for the virtual machine on the SAN-based shared storage as a thin-provisioned disk. If no shared storage is available, the OS and swap disks can be placed on a vSphere datastore that is on local disks configured with suitable RAID protection to ensure that the OS is protected against disk failure.
Deploying multiple virtual machines concurrently on a vSphere server host enables better resource utilization. Technical investigations into this architecture, described in [4] and [5], have shown that multiple Hadoop nodes per host—with each node contained in a virtual machine—can bring benefits to overall system performance. Figure 8 shows a second deployment of the combined model in which there is more than one worker virtual machine per ESXi server host. The eight locally attached disks are divided across the various workloads—that is, across the virtual machines.

In this example, there are four VMDK files allocated to four of the local disks for each virtual machine shown—one VMDK per physical disk. The TaskTracker and DataNode processes in each virtual machine share the VMDK files and the physical disks. This separation of disk access prevents contention for the same disk between the two virtual machines. If the particular performance requirements of the deployed applications require it, a set number of disks can be allocated in this way exclusively to the virtual machine. The disadvantage of this approach is that there are fewer disk spindles handling the I/O from any one virtual machine at any time.

If the master Hadoop roles—the NameNode and the JobTracker—are placed in virtual machines of their own, they can use vSphere HA and vSphere FT by having their data contained on shared storage. More detail is provided on this subject in [12] and [13]. If shared storage is not available, the OS and swap disks can be allocated from a subset of the local disks. These disks are organized so as to have more fault tolerance by configuring two local disks as RAID 1 and placing a vSphere datastore on those disks. The virtual disks for the NameNode and JobTracker virtual machines are then placed on this datastore. This technique is used in [5].

**The Separated Model**

As an alternative to the combined model, the user can specify that the DataNode and TaskTracker Hadoop roles be contained in separate virtual machines. This separation brings the benefit of disconnecting the lifetime of the TaskTracker from that of the DataNode.

As a result, the DataNodes can exist independently of TaskTracker nodes and vice versa. The number of TaskTracker nodes can be varied separately, regardless of the number of DataNodes present at any given time. This separation proves useful as each role is virtualized. For example, TaskTrackers can be added or removed from the system without affecting the DataNode that in the combined model would share a virtual machine with it.

The same principle applies to increasing the number of DataNodes, but if a new DataNode were to be added to a Hadoop cluster, a data rebalancing procedure would be recommended to spread the data out to the new DataNode. This rebalancing work can be network intensive and time consuming. The more likely scenario is to expand or contract the number of TaskTracker virtual machines over time, to match the compute capability to the requirements of the job that is to be executed.

Elasticity of the compute layer in particular can be achieved in this way. Big Data Extensions has algorithms implemented in it to determine how many TaskTracker virtual machines should be running concurrently to best suit the demands of the application workload. Big Data Extensions does this through constant monitoring of the virtual infrastructure layer, using data received from the vCenter performance subsystem, along with consideration of the loading factors on the Hadoop components. Based on this measurement data, Big Data Extensions can either start a new TaskTracker in a virtual machine and join it to a cluster or remove an existing one from the cluster.
The approach shown in Figure 9 for placement of OS disks and swap disks on shared storage is the same as that for the combined-model deployment. In this architecture, each local disk is shared between a DataNode and a TaskTracker (or compute) node through use of common datastores. The local storage is used here for the needs of both the TaskTracker (the temporary data) and the DataNode (the HDFS data).

An advantage of this approach is that all available disk spindles can participate in the I/O traffic. A disadvantage is that the two processes can compete for access at the same time. This can occur if temporary data is being written by the TaskTracker when HDFS data is being read or written to by the DataNode, for example. The local-storage approach can incur extra effort in managing the individual disks independently, compared to a shared-storage approach. However, local storage can be cost-effective in many situations also.

**Network Effects of the Data–Compute Separation**

The data–compute separated deployment model causes a change in the networking data flow between the Hadoop components, as is shown in Figure 10.
Figure 10. Network Data Flow in the Combined Model Versus the Data–Compute Separated Model

The yellow line on the left side of Figure 10 shows the data flow between the TaskTracker and DataNode processes when they are executing in the same virtual machine, as well as showing disk access by the DataNode. The red line represents communication between this virtual machine and others on the network.

The right side of the diagram shows network traffic between the TaskTracker and DataNode when they are separated into their own virtual machines. When the two processes are separated, network traffic between them must traverse the network stack in the guest OS of each virtual machine, along with the network switch implementation in vSphere, as is shown by the yellow line on the right side.

A virtual switch is a software component in the vSphere virtualization platform that, among other functions, logically connects two virtual machines. A virtual switch might or might not be associated with a physical switch. When two virtual machines are located on the same vSphere host server and are connected to the same virtual switch, network traffic between them traverses the virtual switch in memory. Such traffic does not need to be carried on a physical switch or to be handled by the physical network adapter on the host. This feature enhances the performance of network traffic between those virtual machines.

The differences between the two models cited regarding network traffic have been measured and are described in greater detail in [8]. The trade-off here is the extra flexibility gained by having these Hadoop roles separated from one another, even when their container virtual machines are on the same vSphere host.

Separating the two Hadoop roles into different virtual machines can also provide data-security benefits. Visibility of the HDFS data can be restricted through careful attention to access control on the DataNode virtual machine and by placing the HDFS data into a vSphere datastore separate from that of the TaskTracker data. Further description of the vSphere datastore concept is given in a later section of this guide.

Big Data Extensions can be configured to deploy the TaskTracker role on those host servers that also have DataNodes on them. This is enabled through the use of vSphere host affinity between the virtual machines that contain separate TaskTracker and DataNode processes. In this way, network traffic can be optimized between the two processes when they are contained in separate virtual machines.
We also must consider the storage needs of the temporary data items that are in the process of being mapped or reduced. This data is by default stored in locations on the local file system where the TaskTracker is executing, and it is therefore outside of HDFS control. The orange vertical lines in Figure 10 represent the TaskTracker process that is accessing local temporary data, which often is called “temp data.”

Each compute virtual machine has its own temporary data space, the size of which should be planned ahead of time and is tightly related to the design of the specific applications that are using it. For example, the TeraSort Hadoop benchmark application requires a temp data–space size that is at least twice its input dataset size. On the other hand, the related TeraGen and TeraValidate applications do not require significant temporary data space. This part of the design requires careful measurement before the Hadoop cluster is constructed.

A Shared-Storage Approach to the Data–Compute Separated Model

This section discusses another hybrid model for data storage. This model uses a SAN-type shared storage for certain parts of the design and retains the separation discussed earlier between the compute nodes (TaskTrackers) and the data/storage nodes (DataNodes). One part of the TaskTracker virtual machine, a VMDK file, is stored on the SAN while other parts are kept on local, nonshared storage.

Figure 11 shows a SAN- or NAS-based hybrid topology for the data–compute separated deployment model. The advantage to using shared storage is that it provides reliability for the virtual machines hosted on it. Using shared storage for parts of the Hadoop configuration also increases the storage utilization alongside existing vSphere deployments. This model also can be used to reverse the storage mechanisms that are shown—that is, placing the VMDK files of the DataNode on the shared storage and those of the TaskTracker on the local storage. This is done when the bandwidth required for the temporary data outweighs that required for the HDFS data.

With the deployment model shown in Figure 11, the data disk of the compute/TaskTracker virtual machine can increase to cater to the application’s temporary data set. This temporary data is required for the map-phase output, the shuffle phase of the MapReduce algorithms, and possibly also for spills in both the map and reduce phases. Its size is highly application dependent. Making assessments of data size and growth rate at an earlier phase of design is recommended. Providing adequate space on the SAN, or on local storage, to cater to this
temporary data is a necessary step in this design. The following are alternative approaches to this hybrid model if no SAN or NAS access is available in the user’s data center but scalability is required:

- Use locally attached disks for the temporary data storage.
- Use NFS-based storage.
- Use local solid-state disks (SSDs) for storing the temporary data.

The second and third options here can be used in situations where deployments of virtualized Hadoop are being made on blade servers, where slots are insufficient for direct-attached disks. The NFS approach is discussed in the following sections.

**Local Disks for the Application’s Temporary Data**

Studies on the consumption of storage bandwidth by various Hadoop-based applications have shown that a very significant portion—as much as 75 percent in the case of certain applications—of the overall I/O bandwidth is consumed in writing to and reading the temporary data that is used while the MapReduce sort and shuffle phases are executing. This data typically is written to and read from the node’s file system and is not stored in HDFS. Work is in progress on storage of temporary data within HDFS for future Hadoop releases, but that work will not be discussed in this guide.

Speed of access to both the HDFS data and the temporary data is important for the efficiency of the MapReduce application’s algorithms. This requirement supports an argument for using higher-bandwidth disk access for the HDFS and the temporary data. Those higher levels of bandwidth might necessitate alternate strategies to using SAN or shared-storage technology for these types of data. This is a calculation that uses the total I/O bandwidth divided by the number of consuming servers, and it applies to all the models discussed here.

Each SAN device’s I/O bandwidth characteristics are vendor and model specific, so they will not be discussed in detail here. When there is a higher-specification read/write bandwidth supported by the local direct-attached disk approach, that design can be more suitable for storing the HDFS and temporary data. This model is shown in Figure 12.

![Figure 12. Local Disk Storage for Temporary Data](image)
To further explain this subject, some background information is given on the vSphere method for storing a virtual machine’s data. Refer to [13] for a more detailed discussion of provisioning storage using Big Data Extensions and to [14] for VMware vSphere VMFS architecture details and best practices. The following discussion focuses on considerations regarding local storage. Similar considerations for shared storage are discussed in detail in [14].

**Virtual Machines, VMDKs, Datastores, and Local Disks**

A virtual machine’s data is contained in one or more VMDK files managed by vSphere. The VMDK files are placed on VMFS volumes on disk; one volume equates to exactly one vSphere datastore.

In vSphere, a “datastore” is a logical concept that assigns storage space to virtual machines while insulating the guest OS from the details of the underlying physical storage topology. A datastore is mapped to one or more partitions on the disks.

In the most straightforward case, there might be only one partition on a disk, and a datastore might be mapped solely to that one partition, thereby occupying the full disk. This is one way of isolating I/O traffic from one or more virtual machines to the disk. However, the following are the disadvantages of this approach:

1. The I/O bandwidth for the virtual machine or machines contained in that datastore is then constrained by the bandwidth of that disk device.
2. If many disks are present on a server, or if there are many servers, there are also many datastores. This increases the amount of attention required for storage management. A single datastore is more convenient for management than are several datastores.

When there are multiple partitions on disks, a datastore might use the partitions of just one local disk or it might use partitions from multiple local disks on a single physical host.

**NOTE:** As of March 2014, a datastore on a local physical server can be accessed by only one host server. This situation will change as new technology emerges.

Through use of multiple partitions, some or all of the local disks on a server can be used as one vSphere datastore. The following are the advantages of creating a datastore that uses partitions from multiple local disks:

1. As one partition is filled up, subsequent partitions can be used for more storage capacity.
2. The fewer datastores to manage, the more convenient the system becomes for systems management.

Conversely, multiple datastores can use different partitions on the same physical disks, so the I/O traffic from virtual machines contained in separate datastores is handled by the same disk mechanism. If the disk I/O traffic is heavy from virtual machines on two separate datastores concurrently, and if those datastores share access to a single disk—albeit to different partitions—this can cause interference between them, potentially affecting their I/O performance.

A virtual machine can reside on a datastore of its own to minimize interference at the disk I/O level. This has the disadvantage, however, of leading to more management complexity when a large collection of virtual machines results in a large number of datastores.

Figure 12 shows an implementation model in which the VMDK files for the two sample virtual machines are stored on different vSphere datastores that occupy just one partition that is mapped to one disk. Separating the VMDK files for the two virtual machines onto discrete datastores ensures that their I/O traffic is physically separated and makes performance more predictable.

**A Network-Attached Storage (NAS) Architecture Model**

This example NAS deployment model uses an EMC Isilon NAS device. Isilon storage is an expandable set of file servers that export the HDFS interfaces and contain a set of NameNodes to manage that data within the Isilon OneFS file system. In the hybrid model shown in Figure 13, all of the HDFS-related data is stored on the Isilon NAS device. Other data, such as the system disks for the OSs and other installed items within the guest OSs, are held in VMDK files that are mapped to the preexisting shared-storage mechanisms on the left side of the diagram.
Users in the early stages of implementing Hadoop clusters frequently want to leverage their existing virtual infrastructure, which is very commonly supported by shared storage. Their interest is to minimize change, so keeping their existing servers and making use of an Isilon storage system—or server side flash if available—is an interesting option for them.

To a great degree, this simplifies the design for systems that lack local storage. The concern here is the storage for the JobTracker and TaskTracker, or “compute,” roles. As discussed earlier, when performance is a primary concern and bandwidth demands are significant, ideally the temporary data is placed on local storage. But if no local storage option is available, the following alternatives might be possible:

1. Use SSD storage, if available, for the temporary data.
2. If only shared storage is available, the temporary data is required to be placed on the shared storage, possibly limiting the scalability of the system.

All other DataNode and NameNode application data is securely stored on the Isilon device. Traffic to and from the Isilon storage is carried over dedicated network interfaces in the virtual machines that map to physical switches. The switches provide a front end to the Isilon device. A distinct advantage of this design is that it centrally stores the data while still providing the normal HDFS interfaces to it. This approach reduces the amount of time spent on the management of individual locally attached disks on the host servers.

Figure 13. A Deployment Model Based on Isilon NAS Storage

Deployment Models Summary
In this section, we have seen that there are several different models for deployment of Hadoop on virtual infrastructure. We provide corresponding guidelines for choosing between them and for integrating ideas from several models into your own design. There is no one right answer here, particularly where the applications being deployed have different storage requirements.
Deploying Virtualized Hadoop Systems with VMware vSphere Big Data Extensions

Users should refer to the *VMware vSphere Big Data Extensions Administrator’s and User’s Guide* [1] for detailed deployment steps and parameters to apply in the Big Data Extensions cluster specification file to instantiate these topologies. We give an example of a Big Data Extensions cluster specification file for one example topology in Appendix A.

After choosing a topology capable of meeting application needs, users create a detailed plan for the other computing resources that the server hosts will use, including CPU, memory, storage sizing, availability, networking, and hardware layout configurations.

**Storage Sizing**

A Hadoop cluster deployment often starts small, with a cluster that grows as the load on it increases with more applications. Big Data Extensions provides support for scaling a deployed cluster up or down in size.

Storage capacity is an important initial determination to make. This requires a user to predict data growth over time and to consider temporary data and replicated data for the MapReduce and HDFS algorithms. The required data space can be three times or more than that needed for the input data that is first loaded into the cluster. A storage-sizing exercise example is provided later in the “General Best Practices for Hadoop on vSphere” section.

**Availability**

The design of Hadoop ensures that a Hadoop cluster is resilient to many common failures, such as server or storage failure. The majority of servers in a Hadoop cluster are worker nodes. Hadoop has built-in failure-recovery algorithms to detect and repair a cluster when a worker node fails. Data blocks are replicated in HDFS such that if the host containing one copy of any data block becomes unavailable, the replica for that block can be found on another host in that replica group.

However, for the entire cluster to function correctly, Hadoop clusters depend on the availability of a few master nodes. Early Hadoop releases did not provide an availability solution for its master nodes. These master nodes—the NameNode and JobTracker are the most critical—provide key services that are necessary for Hadoop to function. Without the NameNode, data stored in HDFS cannot be accessed or modified. Without the JobTracker, end-user programs, such as MapReduce jobs, cannot be controlled and new jobs cannot be executed. As a result, there is a need for more-extended high-availability solutions, which the additional tools available in vSphere and in Big Data Extensions can provide.

Solutions on native systems to ensure higher levels of availability have traditionally been very costly, hard to implement, and difficult to manage. vSphere makes it easier for important applications.

First, vSphere HA capability reduces unplanned downtime by leveraging multiple ESXi hosts configured as a vSphere HA cluster, to provide rapid recovery from server or application outages. This provides a cost-effective high-availability solution for applications running in virtual machines.

Second, vSphere FT, a capability that is a function of an individual virtual machine, provides an even higher level of failover for those components of the system that need it. When a virtual machine is configured to be fault tolerant, a shadow or secondary virtual machine is operating in lockstep with it on another host server. If the primary vSphere FT enabled virtual machine fails, the secondary, or shadow, virtual machine takes over from it immediately, without interruption to the execution of the processes within the guest OS. The new primary virtual machine is backed up by a new secondary virtual machine, which maintains the level of fault tolerance. vSphere automatically executes all of these functions.

This helps sustain critical processes that must not stop for even the shortest period of time, such as those found with production systems. The subject of fault tolerance in the protection of critical processes in Hadoop systems is explored in greater detail in [12]. We next focus on the capabilities of vSphere HA.
vSphere HA protects Hadoop application availability in the following ways:

- It protects against hardware failure and network disruptions by restarting virtual machines on active hosts within the vSphere cluster when an original host server fails or the network connection to it fails.
- It protects against guest OS failure by continuously monitoring a virtual machine and restarting it as required.
- It provides a mechanism to react to application failures.
- It provides the infrastructure to protect all workloads within the cluster.

After vSphere HA has been configured for a vSphere cluster, no actions are required to protect new virtual machines. If all of the data storage of the virtual machines is contained on shared storage, those virtual machines that reside on hosts within the vSphere HA cluster are automatically protected. vSphere HA can be combined with vSphere DRS to protect against failures and to provide load balancing across the hosts within a cluster.

The NameNode and JobTracker roles can be single points of failure in some Hadoop clusters. If the hardware, OS, or master software for these hosts fails, the entire Hadoop cluster might become unavailable. This issue can be addressed by deploying these roles in a vSphere HA cluster.

Figure 14 shows an example of a vSphere HA cluster that consists of two ESXi server hosts with the NameNode and JobTracker virtual machines running on them. If a vSphere host server containing a NameNode or JobTracker virtual machine fails unexpectedly, another vSphere host in the vSphere HA cluster can bring up the appropriate virtual machine to become the active NameNode or JobTracker. There is no explicit host in the cluster configured as a hot or cold standby NameNode; that role can appear on various ESXi host servers over time if needed.
To store the data for these protected virtual machines, a shared-storage mechanism such as a SAN is required. Big Data Extensions can configure the requirement for vSphere HA for the NameNode and JobTracker at the time of Hadoop cluster creation, using resources previously set up by the vSphere administrator. vCenter configures and manages the vSphere HA cluster.

Besides the NameNode and JobTracker, several of the nodes in a classic Hadoop cluster are configured to be worker nodes. Hadoop has its own built-in failure-recovery algorithms to detect and repair Hadoop cluster components if a worker node fails. Although these nodes do not always require vSphere HA support, there are advantages to using a uniform approach whereby vSphere manages all the participating hosts, including those that have worker virtual machines on them. Virtualizing the entire Hadoop cluster provides operational benefits regarding automation, datacenter consolidation, rapid provisioning, higher resource utilization, systems management, and having a uniform monitoring and management framework for all nodes.

**Hardware Considerations**

Hardware vendors and Hadoop software distributors provide most of the specific recommendations for configuring the virtualized Hadoop cluster hardware. There are some important points to bear in mind when choosing hardware for virtualizing Hadoop.

The design and sizing of the virtual machines are discussed in various architecture examples that can be found in [4] and [5]. Table 1 provides specifications of certain CPU, RAM, and disk setups that form a baseline for a server to be used in an initial proof-of-concept Hadoop cluster.

Avoid resource contention and the associated additional scheduling overhead through either exact resource provisioning or the use of vSphere reservations, limits, and resource pools [15]. A resource pool is required by Big Data Extensions as a deployment target for a new Hadoop cluster, for example.

For the JobTracker and NameNode roles, consider deploying additional RAM and secondary power supplies to ensure the highest performance and reliability of these critical servers in the cluster. Given the Hadoop data distribution model, however, it is not typically a requirement to deploy power redundancy on the ESXi servers that contain virtual machines with worker nodes in them.

<table>
<thead>
<tr>
<th>CATEGORY NAME</th>
<th>CONFIGURATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>Servers should have two CPUs with at least four to eight cores each. Hyper-threading should be enabled. More-modern, powerful CPUs are preferred, to prevent bottlenecks in computing resources.</td>
</tr>
<tr>
<td>RAM</td>
<td>Provide approximately 6 percent extra memory for the virtualization layer at the host level. Provide at least 4GB to 8GB memory per core.</td>
</tr>
<tr>
<td>Disks</td>
<td>Testing described in [4] and [5] shows that JBOD, 7200RPM SATA, and NL-SAS provide acceptable performance for benchmark programs such as TeraGen, TeraSort, and TeraValidate when executed across multiple virtual machines. As many as 8 to 16 disks were used, depending on the storage-to-computation ratio. Eight disks are appropriate, for example, when there are two quad-core processors as described above. Hadoop software distributors usually recommend 1 to 1.5 disks per core on native systems.</td>
</tr>
</tbody>
</table>

Table 1. CPU, RAM, and Disk Configurations for Worker Nodes

The systems administrator should follow the guidelines given by their Hadoop distributor company when choosing the hardware specification for their particular setup, applying prudent judgment at the same time.
Systems Architecture Examples

In this section, we provide several example high-level configurations to show the systems used in proof-of-concept projects, in performance tests, and in scalability testing work. In a subsequent section, we offer guidelines for configuring Hadoop on virtual machines on vSphere.

For early, lower-scale work on Hadoop, where there are fewer than 30 nodes in the Hadoop cluster, it is likely that all vSphere host servers in the cluster are contained within one physical rack in the datacenter. This means that all network traffic will be determined by the speed of the networking within that rack.

In contrast, Hadoop clusters in larger systems use a two- or three-level architecture made up of several racks of host servers. Each server is connected to the top-of-rack (TOR) switch, and the rack of servers is connected to other racks via an aggregation or core switch.

The data-transfer speed at which these switches operate can be a deciding factor in overall system performance. When there is sizable cross-node traffic for replication of data, interrack traffic can become a limiting factor for system performance. The TOR switches are replicated for redundancy to provide adequate bandwidth for data rebalancing when a TOR switch fails.

Example Configurations

The following discussion includes examples of three system layouts that were used, respectively, for prototyping, performance testing, and measuring the scalability of workloads with Hadoop nodes in virtual machines. These examples serve as patterns that can be considered for deployment, depending on the application requirements and on the organization’s level of Hadoop adoption. Other architectures that are not covered here are also viable, such as using an Isilon NAS storage mechanism for HDFS data handling, but this discussion is confined to a small set of examples. Further information on the Isilon configuration is available at the EMC Web site. We show three different examples, representing small, medium, and larger cluster configurations.

1. A Small, Single-Rack Proof-of-Concept Configuration

This configuration was used at the VMware engineering labs to test a smaller Hadoop cluster architecture. It represents a user in the beginning phase of proof-of-concept testing with Hadoop on vSphere. The architecture, shown in Figure 15, uses five vSphere server hosts, each connected to a 10GbE switch within the same rack. This configuration was not tuned for high-performance results because creating benchmark performance reports from it was not the objective.

One virtual machine in this design was dedicated to the NameNode and JobTracker processes. All other virtual machines contained the combined DataNode and TaskTracker processes.

Figure 15 shows that one of the vSphere server hosts contained the master Hadoop roles—NameNode and JobTracker—as well as the worker roles—DataNode and TaskTracker—in separate virtual machines. Other vSphere server hosts contained Hadoop-specific processes along with virtual machines dedicated to other tasks, such as DHCP servers and vCenter Servers.

This mixture of various workloads is very common in virtualized environments, and it presents an opportunity to utilize the hardware resources more efficiently. When the Hadoop workload requires compute power, it is made available to its virtual machines through vSphere resource management features such as reserved resource pools. When the Hadoop workload becomes less demanding or quiescent, some or all of those resources can be allocated to other tasks. The virtual Hadoop manager component within Big Data Extensions makes determinations regarding competing Hadoop clusters based on data supplied by vSphere DRS.
Deploying Virtualized Hadoop Systems with VMware vSphere Big Data Extensions

Figure 15. Example of a Virtualized Hadoop Topology for a Single-Rack Deployment

The MapReduce test programs that are normally shipped with the Hadoop distributions were used here to establish confidence in the general operation of the system. The architecture shown in Figure 15 was used for testing purposes only and was not considered to be a production-ready setup. See [4] for a setup at similar scale, though with expanded architecture, that was used in a performance-testing context. In the following section, we discuss an example of a larger reference configuration with more resources, optimized for high performance.


While carrying out a performance-testing exercise on various configurations of virtual machines containing Hadoop on vSphere, VMware engineers used a single rack of servers with a 10GbE network switch for connections between the servers. They used this setup to minimize or eliminate any network-based bottlenecks when looking at system performance. We show an architecture outline to demonstrate a useful configuration for performance testing of an end-user’s Hadoop-based application. In this case, 32 vSphere server hosts were used in the testing, each with a 10GbE network card connected directly to a 10GbE switch (Arista 7050S). This hardware setup, shown in Figure 16, along with the benchmark programs used and the full test results, is reported on in [5].

http://www.vmware.com/resources/techresources/10360

Figure 16. Example of a Medium-Sized System Setup Used for Performance Tests
3. A Larger, Multirack Configuration

The larger architecture shown in Figure 17 was used for a set of multiple concurrent-workload tests conducted by VMware engineers with VMware technology partners in late 2013. It illustrates an environment using more than 100 vSphere server hosts and more than 800 virtual machines in four separate Hadoop clusters at one time.

The objective of this test sequence was not to conduct a strict performance benchmarking exercise but rather to gauge the effects of multiple concurrent workloads executing in parallel on different virtual machines on the same host servers. Example applications that depend on both Hadoop and HBase were used in this testing, along with a shared HDFS cluster setup. This technical architecture, as well as the findings of the conducted tests, is documented in [7].

Figure 17. Systems Layout for a Multiple Concurrent-Load Test
Best Practices

In this section, we discuss two separate, though related, best-practice topics:

- Big Data Extensions technology-specific best practices
- Best practices for deploying Hadoop on vSphere

Because Big Data Extensions technology incorporates several elements of the latter best-practice type, we will refer to Big Data Extensions features as part of the second topic as well. This section of the guide should be read as an introduction to these best-practice areas. The subject matter is continually evolving, and further work is being done to understand and document more of these practices.

Big Data Extensions Best Practices

A Big Data Extensions-driven Hadoop deployment can be viewed as a stack consisting of the Hadoop software, the guest OS within one or more virtual machines, the ESXi hypervisor, and the underlying hardware. To get optimal performance from this stack of technology, the layers must operate together efficiently. Big Data Extensions frees the user from the extra work of configuring system parameters when provisioning a Hadoop cluster. Much of this exists in template form in the cluster specification file, or interactively with the user interface for the Big Data Extensions plug-in component, when designing a Hadoop cluster.

Some of these best practices are provided by default when using Big Data Extensions to create a Hadoop cluster; others require additional effort on the part of the administrator. Users will choose whether or not to comply with these practices, in accordance with their own needs.

Changing the Guest Operating System of the Big Data Extensions Template

The Big Data Extensions template that ships with the product is designed for common deployments and for test and development scenarios. For those situations in which the intention is to maximize the performance of the Hadoop cluster, the OS in the Big Data Extensions template virtual machine might require an upgrade to a later version.

As of March 2014, the default Big Data Extensions template contains the CentOS 5.9 guest OS, which is in common use. However, some users prefer CentOS 6.x because it has features that enhance performance on vSphere. One such feature is the Linux Transparent Huge Pages (THP) memory management function. This functionality, along with the Intel Extended Page Table (EPT) technology that is available in contemporary x86 processors, can optimize memory management on vSphere. Using THP and EPT together is recommended for better performance of Hadoop systems on vSphere hosts.

For RHEL 6.1, the THP feature is turned on by default, which is preferable. The RHEL 6.2 and 6.3 versions of Linux, because of a known problem with them, require a particular feature to be disabled in order to enable THP. Consult RHEL documentation for more details on this. RHEL 6.4 also has the THP feature available by default.

Users who want to take advantage of the latest OS features in their systems—such as those in CentOS 6.x—should follow the steps provided in the VMware vSphere Big Data Extensions Administrator’s and User’s Guide [7] to upgrade the Big Data Extensions template to later OS versions. The Big Data Extensions product will incorporate later versions of this operating system also in further releases.

Upgrading the Virtual Hardware in the Big Data Extensions Template

The vSphere hypervisor presents a software abstraction of the hardware that is managed by the hypervisor to the guest OS in each virtual machine. This abstraction is called the “virtual hardware.” There are different versions of this virtual hardware over time, as new releases of vSphere appear and new features are added to it. Each virtual machine on a vSphere host is associated with a particular version of the virtual hardware, and that virtual hardware version determines the set of features that the virtual machine can use.
Different virtual hardware versions have different features and different fixes to problems found in earlier versions. Due to timing considerations, the virtual hardware version that is most current at a particular time might be a later one than that in the Big Data Extensions virtual appliance. Users might want to upgrade that virtual hardware to take advantage of the latest features. For example, the user might want to create virtual machines with more vCPUs than are allowed by an earlier version of the virtual hardware. This section explains how to make that upgrade.

The following are the steps for customizing the Big Data Extensions template to upgrade virtual hardware and the network adapter type:

Step 1: In VMware vSphere Client™ or the vSphere Web Client, right-click the template virtual machine icon in the left navigation tree and choose Upgrade Virtual Hardware.

Step 2: Choose Accept on the dialog box window.

Step 3: To add to the virtual CPU capacity of a virtual machine, perform the following actions: Right-click the template virtual machine icon and choose Edit. Choose the item that is to be increased, such as the number of virtual CPUs, and supply the value for that virtual hardware item. In this example, the number of virtual CPUs is being upgraded to 8 from the original value of 1.

The vSphere Client view of the virtual hardware version and other properties of the virtual machine is shown in Figure 18.
Warm Up the Disks After Provisioning a Hadoop Cluster with Big Data Extensions

Background Explanation

vSphere disks on datastores can be created in one of several forms. We will leave aside the Raw Device Mapping (RDM) form for this discussion and will concentrate on creating disks in one of the three following main forms:

**Zeroed Thick**

All the space is allocated on the datastore at the time of virtual disk creation. It is not prezeroed, so it is quick to create. As the OS in a virtual machine writes to the disk, the space is zeroed as the I/O is committed. Zeroing the disk ensures that no old data from the underlying storage is found on the new disk.

**Eager-Zeroed Thick**

Using this method, as before, all the space is preallocated to the virtual disk on the datastore when the disk is first created. However, with eager-zeroed thick disks, the entire space is zeroed out at this time. These disks can take considerable time to create; but when they’re ready, they exhibit a marked performance improvement over new zeroed thick disks. The performance of zeroed thick disks eventually catches up to that of eager-zeroed thick disks, typically within a few hours.

**Thin**

Similar to thin provisioning on a storage array, VMDK thin disks are allocated space only as they grow from disk I/O activity. The disk starts small and expands as the space is zeroed, ready for disk I/O. It will not grow beyond its allowed size. Despite speculation to the contrary, thin provisioning does not impact performance, which is extremely close to that of zeroed thick disks. The main advantage of thin disks is the space saved by not allocating everything in advance. However, some guest disk operations, such as defragmentation, cause thin disks to inflate.

**Determining the Disk Format to Use**

To avoid incurring the longer provisioning and creation times associated with the eager-zeroed thick form of disk, Big Data Extensions uses the zeroed thick option in vSphere as the standard method of laying out all the data disks when it is creating them.

When compared with zeroed thick, the eager-zeroed thick option takes a significantly longer time in provisioning disks because of the preallocation work that vSphere must do. However, this preallocation work facilitates a better I/O write performance at runtime. Therefore, we suggest that users zero-fill all disks that were created as lazy-zeroed disks, thereby changing them into eager-zeroed disks.

This is done so all write operations can benefit from the disk preallocation work, thereby improving performance on a Hadoop cluster. This zero-filling process is called “warming up the disks.” Based on the results of internal VMware experiments done on a test cluster, it is estimated that this will result in a worthwhile performance gain.

There are a variety of techniques for warming up the disks, from executing custom scripts that zero-fill the disk to running a disk-writing I/O-intensive application, such as the TeraGen sample program, on them. The system administrator can choose which method to use here.
General Best Practices for Hadoop on vSphere

This section introduces best practices for setting up each of the main compute resources—disk I/O, CPUs, memory, and so on—when preparing them to run Hadoop-based workloads. Many of these best practices apply equally to other types of workloads on vSphere.

As a general recommendation, place a virtualized Hadoop cluster on the most up-to-date hardware that is available. This will enable optimal specifications for clock speed, core count, hyper-threading, and caching.

Disk I/O

1. To set up the system for optimal performance, the more locally attached disks available on the servers that support the Hadoop worker nodes, the better. Because the TaskTrackers and DataNodes depend heavily on disk I/O bandwidth, having more available disks decreases the likelihood that requests will execute in parallel on a single device and thereby cause an I/O bottleneck. There are examples earlier in this document of vSphere servers with eight local disks, but it is not uncommon with Hadoop workloads to have servers in use with at least twice that number of local disks. One Hadoop distributor recently recommended using 32 local disks for larger servers. This is the “power user” end of the spectrum and has some disadvantages.

2. Use 1 to 1.5 disks per core or vCPU when possible. This recommendation is consistent with that of the Hadoop distribution vendors. Having more independent disks increases the management attention required as compared with a centralized, shared-disk system. So for the architect, this is a trade-off between the stated performance benefits and the cost of managing the independent disks.

3. Provide approximately 100Mb per second of disk bandwidth per core or per vCPU. For SATA disks, that equates to roughly one disk per virtual CPU.

4. For power-user applications, SATA-type disks at 7200RPM have performed well. This type of disk setup requires more management oversight than does a centralized SAN arrangement.

5. RAID is not required on disks that contain HDFS data, because the HDFS system replicates its data blocks regardless. If RAID controllers are present in the storage subsystem being used, the disks should be configured to use pass-through mode if available. If pass-through functionality is not available for the RAID controller, a single LUN or virtual disk should be created in RAID 0 format on each physical disk.

NOTE: The term “virtual disk” here is a vendor-specific concept and does not refer to a VMDK.

6. In keeping with the recommendations of Hadoop distributors for native systems, provided there is flexibility of choice, the LVM, RAID, and IDE technologies should not be used in the I/O-intensive areas of virtualized Hadoop systems.

7. Ensure that the local disk controller has enough bandwidth to support all the local disks.

Sizing the Data Space

1. Data blocks are replicated three times by default in HDFS, so this must be taken into account when sizing the data space for DataNodes in particular.

2. The HDFS data block replication factor can be configured differently for different Hadoop applications.

The following is an example disk-sizing exercise:

- We make the assumption that the data volume grows by approximately 3TB per week.
- By default, HDFS is set up to contain three replicas of each data block.
- Therefore, 9TB of extra storage space is required per week.
- We add a value of 100 percent of the input data size for the temporary space, resulting in 12TB per week.
- Assuming that we have server machines with 12 x 3TB disk drives, a new machine is required every three weeks.
Create Aligned Partitions on Disk Devices

To store virtual disks, vSphere uses datastores, which are logical containers that hide the specifics of physical storage from the virtual machines and provide a uniform model for storing virtual machine files. Datastores deployed on block storage devices use the VMFS format, a special high-performance file system format that is optimized for storing virtual machines.

Disk-datastore alignment is very important in achieving I/O operational efficiency in the virtualization environment, as it is in the physical one. Alignment is required at two levels: at the vSphere host server level and within the guest OS of each virtual machine.

Alignment at the vSphere Server Host Level

Utilize the vSphere Web Client user interface to achieve this alignment when creating a datastore on a disk. vSphere automatically aligns partitions on the disks that it controls.

There might be situations where alignment must be done over many disks and where using the vSphere Web Client user interface would involve much repetitive and error-prone work. For these situations, custom scripts are developed and used by operations personnel to execute the alignment on a larger scale.

See the arrangement of server datastores by using the following menu combination within the vSphere Web Client:

_hosts and Clusters > Configuration > Storage > Datastores

Figure 19 shows an overview of this.

Figure 19. Datastores Overview in the vSphere Web Client

Guest OS Disk Alignment

Big Data Extensions automatically handles disk alignment at the guest OS level in a virtual machine during the provisioning of a Hadoop cluster. On the Linux guest OS flavors, use the `fdisk -lu` command to verify that partitions are aligned.

Virtual CPUs

1. At least two virtual CPUs are recommended for any virtual machine that is executing a significant Java process, such as the main Hadoop processes. See vSphere Enterprise Java Best Practices [9] for more details.

2. For optimal performance, it is recommended that the physical CPUs on the vSphere host not be overcommitted. Ideally, the total number of virtual CPUs across all virtual machines on a host server should be equal to the physical core count on that server. This ensures that no virtual CPU is waiting for a physical CPU to be available before it can execute. When the latter occurs, we normally see an increase in %Ready time as measured by vSphere performance tools. When hyper-threading is enabled, the total number of vCPUs might be twice the number of physical cores.
3. Virtual machines that fit within the number of cores in a socket, and that use the associated memory for that socket, often perform better than larger ones that span multiple sockets. Therefore, we recommend limiting virtual CPUs in any virtual machine to a number that is less than or equal to the number of cores in a socket on the target hardware. This prevents the virtual machine from being spread across multiple sockets and can help it perform more efficiently. See the related discussion of NUMA in the “Memory” section that follows.

Memory
1. The sum of all the memory configured in the virtual machines on a server should not exceed the size of physical memory on the host server.

2. Avoid exhausting the memory of the guest OS within the virtual machine itself. Each virtual machine has a configured memory size that limits its addressable memory space. When a set of memory-hungry processes, such as the Hadoop roles, is executing in the guest OS, ensure that there is enough guest OS memory space configured to enable those processes to execute without incurring swapping at the guest OS level.

3. To increase speed and efficiency, nonuniform memory access (NUMA) breaks the server’s main memory into parts that are closely associated with individual processors. Each of these parts is a NUMA node and has a particular size on various architectures. By designing a virtual machine’s memory size to fit within the boundary of a NUMA node, the resident Hadoop workload’s performance should not be affected by cross-NUMA node migrations or accesses. Virtual machines with memory space that spans more than one NUMA node can incur performance impacts through use of cross-node memory accesses.

4. In the physical memory of a vSphere host server, allow for the memory requirements of the vSphere hypervisor and for the overhead incurred by each virtual machine. The latter overhead is described in vSphere documentation. A general guideline is to set aside 6 percent of physical memory for the hypervisor’s own use.

5. Use a memory-to-core ratio of at least 4GB of memory to one core or virtual CPU.

Networking
1. Use dedicated network switches for the Hadoop cluster and ensure that all physical servers are connected to a TOR switch.

2. Use a bandwidth of at least 1Gb per second to connect servers running virtualized Hadoop workloads.

3. Provide between 200mb per second and 600mb per second of aggregate network bandwidth per core, depending on the network requirements of the system.

Example: 100-node Hadoop cluster
- 100 x 16 cores = 1,600 cores
- 1,600 x 50MB per second = 80GB per second
- 1,600 x 200Mb = 320Gb of network traffic
- 1,600 x 600Mb per second = 960Gb per second of network traffic

Details on Virtualized Networking
Typical servers have dual network ports. In many cases, two ports are not enough for ESXi hosts. In configuring ESXi host networking, consider the traffic and loading requirements of the following:
- Management network
- Virtual machine port groups
- IP storage (NFS/iSCSI, FCoE)
- vSphere vMotion
- Fault tolerance
The virtual machines within the Hadoop cluster might also require network connectivity to the corporate network. When designing this portion of the infrastructure, there should be no dependencies on a single network adapter or on a connection to a single physical switch. Redundant pathways for virtual machine-to-virtual machine traffic should be planned into the design.

Scalability is another consideration. The larger the environment grows, the more dynamic it becomes and the harder it becomes to manage the network configuration and keep it consistent across all the hosts in a cluster.

**Virtual Distributed Switch**
VMware vSphere Distributed Switch™ (VDS) technology eases the management overhead per host and virtual switch configuration management by treating the network as an aggregated resource. Individual, host-level virtual switches are abstracted into one large VDS that spans multiple hosts at the datacenter level. Port groups become distributed virtual port groups (dvport groups) that span multiple hosts and ensure configuration consistency for virtual machines and virtual ports necessary for such functions as vSphere vMotion and network storage. A VDS is suitable for all types of networking that have been discussed.

**Examples of Various Networking Configurations**
The two following examples show details of the network setup for ESXi host servers with four network adapters (1Gbps) and three network adapters (two at 1Gbps and one at 10Gbps) respectively.

**Example 1**

Figure 20. Example Network Configuration with Four 1Gbps Network Adapters

Figure 20 shows the case of four network adapters. All the virtual machines containing the Hadoop components are connected to virtual switch 1. Therefore, the Hadoop virtual machine network traffic goes through virtual switch 1. The network interfaces vmnic 2 and vmnic 3 are both active and serve this virtual switch. All vSphere management subsystems, such as the VMkernel, vSphere vMotion, management (MGMT), and vSphere FT components, are connected to virtual switch 0. On virtual switch 0, any MGMT and VMkernel networking traffic is carried on vmnic 0, with vmnic 1 acting as a standby. vSphere vMotion and vSphere FT network traffic is carried on vmnic 1, with vmnic 0 acting as a standby.
Example 2

Figure 21. Example Network Configuration with Two 1Gbps Network Adapters and One 10Gbps Network Adapter

Figure 21 shows an example hardware case with a 10Gbps network adapter card on a host ESXi server containing the Hadoop virtual machines. The network traffic from those Hadoop virtual machines goes through virtual switch 1 and is delivered by vmnic 2, which is capable of higher bandwidth than the other network adapters. On virtual switch 0, the MGMT and VMkernel traffic is directed through vmnic 0, with vmnic1 as a standby. vSphere vMotion and vSphere FT network traffic is carried on vmnic 1, with vmnic 0 on standby.

Customizing the Hadoop Configuration Parameters

This is a complex subject area and is treated in much greater detail in [17]. Tuning a Hadoop system for performance is an iterative and emerging field in which best practices are emerging over time, so consider this section a basic introduction to the subject.

When configuring a Hadoop cluster, pay close attention to the Hadoop-specific parameter settings. This provides a balance between efficient scheduling and overhead reduction as well as reaching an acceptable balance between reliability and performance. Users must tune these parameters (in the hdfs-site.xml and Mapred-site.xml files) in accordance with their own requirements and with the recommendations of the Hadoop distributors.

Big Data Extensions has optimized several of these Hadoop settings based on VMware testing of virtual machine configurations. Consider the following key points:

- The user can configure HDFS block size. Big Data Extensions currently sets the default block size at 256MB. Consider using 128MB—the Hadoop default—with correspondingly smaller Java heap sizes and other memory parameters if memory needs of the applications are less than 8GB per core.
- mapred.child.ulimit settings should be proportional to the Java maximum heap size (-Xmx) value in the mapred.child.java.opts.
• Setting the log level of log4j to be less verbose will help prevent filling up the available disk space within the virtual machine.
• Setting the MapReduce session management functions will help prevent the “out of time” error.

In general, Hadoop parameter tuning should be done in the same way as with native implementation, while also applying suitable judgment from the virtualization perspective.

Conclusion

This document discusses VMware vSphere Big Data Extensions technology for deploying Hadoop workloads on vSphere. This technology reduces the time and effort that the virtual infrastructure administrator and the Hadoop administrator spend in configuring and managing a cluster of Hadoop nodes in virtual machines.

This guide also examines applicable reference architectures and establishes guidelines for choosing one in deploying a Hadoop cluster on vSphere. Several options apply here, ranging from separation of the compute and data functions within Hadoop to storing the various data items on locally attached storage or other mechanisms. The type of Hadoop system deployed determines the choice between these alternatives. There is no single answer that fits all needs.

Using this document, the reader will understand that vSphere is a viable platform for successfully running both development and production Hadoop workloads and that there are distinct advantages to be gained by combining the technologies.

About the Authors

Xinhui Li is a senior member of the development staff at VMware. Her work focuses on the design and optimization of distributed systems, most recently in the HPC and big data fields. She works on the adoption of new technologies on vSphere and provides technical contributions for the vSphere Big Data Extensions product. She writes technical documents, blog articles, and academic papers regularly.

Justin Murray is a senior technical marketing manager at VMware. He has worked at the company since 2007 in various roles, with a main focus on helping customers and partners use VMware products for deploying their applications on Hadoop and other platforms. To this end, he creates technical material for consumption by architects in the virtualization and application spaces and gives talks regularly on these subjects.
Appendix A: Big Data Extensions
Configuration File – Example

When testing the Big Data Extensions facilities for the virtualization of a Hadoop cluster, the tests started with the configuration of master and worker nodes—as shown earlier in Figure 15—a “single rack proof of concept lab test configuration” that would be deployed in a typical small Hadoop cluster. More worker nodes can be cloned and added to the cluster to scale out to the size wanted for a specific environment. The following specification was used as input to Big Data Extensions to provision this cluster. The individual Hadoop roles that were provisioned are shown on their respective hosts in Figure 15.

```json
{
  "nodeGroups": [
  {
    "name": "master",
    "roles": [
      "hadoop_namenode",
      "hadoop_jobtracker"
    ],
    "instanceNum": 1,
    "cpuNum": 2,
    "memCapacityMB": 7500,
    "storage": {
      "type": "SHARED",
      "sizeGB": 50
    },
    "haFlag": "on",
    "configuration": {
      "hadoop": {}
    }
  },
  {
    "name": "worker",
    "roles": [
      "hadoop_tasktracker",
      "hadoop_datanode"
    ],
    "instanceNum": 5,
    "cpuNum": 24,
    "memCapacityMB": 77824,
    "storage": {
      "type": "LOCAL",
      "sizeGB": 7200
    },
    "haFlag": "off",
    "configuration": {
      "hadoop": {}
    }
  }
  ]
}
```
// we suggest running convert-hadoop-conf.rb to generate "configuration" section and paste the output here
"configuration": {
  "hadoop": {
    "core-site.xml": {
      // check for all settings at http://hadoop.apache.org/common/docs/r1.0.0/core-default.html
      // note: any value (int, float, boolean, string) must be enclosed in double quotes and here is a sample:
      // "io.file.buffer.size": "4096"
    },
    "hdfs-site.xml": {
      // check for all settings at http://hadoop.apache.org/common/docs/r1.0.0/hdfs-default.html
    },
    "mapred-site.xml": {
      // check for all settings at http://hadoop.apache.org/common/docs/r1.0.0/mapred-default.html
    },
    "hadoop-env.sh": {
      // "HADOOP_HEAPSIZE": "",
      // "HADOOP_NAMENODE_OPTS": "",
      // "HADOOP_DATANODE_OPTS": "",
      // "HADOOP_SECONDARYNAMENODE_OPTS": "",
      // "HADOOP_JOBTRACKER_OPTS": "",
      // "HADOOP_TASKTRACKER_OPTS": "",
      // "HADOOP_CLASSPATH": "",
      // "JAVA_HOME": "",
      // "PATH": ""
    },
    "log4j.properties": {
      // hadoop.root.logger": "INFO,RFA",
      // "log4j.appender.RFA.MaxBackupIndex": "10",
      // "log4j.appender.RFA.MaxFileSize": "100MB",
      // "hadoop.security.logger": "DEBUG,RFA"
    },
    "fair-scheduler.xml": {
      // check for all settings at http://hadoop.apache.org/docs/r1.0.0/fair_scheduler.html
      // "text": "the full content of fair-scheduler.xml in one line"
    },
    "capacity-scheduler.xml": {
      // check for all settings at http://hadoop.apache.org/docs/r1.0.0/capacity_scheduler.html
    },
    "mapred-queue-acls.xml": {
      // check for all settings at http://hadoop.apache.org/docs/r1.0.0/cluster_setup.html#Configuring+the+Hadoop+Daemons
      // "mapred.queue.queue-name.acl-submit-job": "",
      // "mapred.queue.queue-name.acl-administer-jobs": ""
    }
  }
}
Appendix B: Hadoop Configuration for a Sample Test

Host CPU and memory:
2x Intel® Xeon® Processor X5650 2.66GHz, 12MB cache, 6.4 GT/s QPI, turbo, HT enabled,
6C heatsinks,
48GB memory (6x 8GB), 1,333MHz, dual-ranked LV RDIMMs for two processors

Host storage controller:
PERC H700 integrated RAID controller
512MB cache write-through

Host hard disks:
2x 500GB 7.2K RPM SATA 3.5” hot-plug hard drive,
two mirrored disks divided into LUNs for native OS and ESXi root drives and virtual
machine storage
8x 1TB 7.2K RPM SATA 3.5” hot-plug hard drive, configured as RAID 0 for Hadoop data

Host hard network:
Broadcom NetXtreme II 57711 dual-port SFP+ direct attach 10GbE network adapter PCIe
x8 with TOE and iSCSI

Network switch
Arista 7124s, 24 ports, 10GbE

Host file system:
Single Linux partition aligned on 64KB boundary per hard disk
Partition formatted with EXT4:
4KB block size
4MB per node

Host BIOS settings:
Intel Hyper-Threading Technology: enabled
C-states: disabled
Enhanced Intel SpeedStep Technology: disabled

Linux
OS version: CentOS6.2 x86_64, kernel 2.6.32-220.e1. x86_x64
Kernel parameters:
nofile=650000
nproc=650000
Java: Sun Java 1.6.0_25

Virtual machines
Hypervisor: vSphere 5.0
VMware Tools: installed
Virtual network adapter: vmxnet3
Disks: VMFS5
One worker virtual machine per host:
77,824MB, 24x vCPU (HT enabled), eight data disks
one master virtual machine together with one worker virtual machine on host
1,024MB, 2x vCPU, one data disk
Cluster topology:
Native, 1 virtual machine per host, all nodes in the default rack
Five hosts, five worker virtual machines and one master virtual machine

Hadoop:
Distribution: CDH 4.2.1
Nondefault parameters
  mapred.map.tasks.speculative.execution=false  
  mapred.reduce.tasks.speculative.execution=false  
  mapreduce.tasktracker.outofband.heartbeat=true  
  mapred.child.ulimit = 3* Xmx  
  dfs.replication=2  
  dfs.block.size=268435456  
  dfs.socket.timeout=3000000  
  dfs.datanode.socket.write.timeout=3000000  
  dfs.balance.bandwidthPerSec=10485760  
  dfs.datanode.drop.cache.behind.writes=true  
  dfs.datanode.drop.cache.behind.reads=true  
  dfs.datanode.sync.behind.writes=true  
  hadoop.root.logger=INFO, RFA  
  log4j.appender.RFA.MaxFileSize=100MB  
  log4j.appender.RFA.MaxBackupIndex=10  
  mapred.child.java.opts=-Xmx855m -Xmn256m  
  mapred.tasktracker.map.tasks.maximum=24  
  mapred.tasktracker.reduce.tasks.maximum=24  
  mapred.reduce.parallel.copies=150  
  io.sort.factor=150  
  io.sort.mb=600  
  io.sort.record.percent=0.14
References

Further information about virtualization of Hadoop on VMware vSphere and using vSphere Big Data Extensions can be found in the following documents:

VMware Resources

[1] VMware vSphere Big Data Extensions User’s and Administrator’s Guide

[2] vSphere Big Data Extensions
   http://www.vmware.com/hadoop

[3] Apache Hadoop 1.0 High Availability Solution on VMware vSphere


[5] Virtualized Hadoop Performance with VMware vSphere 5.1
   http://www.vmware.com/resources/techresources/10360

[6] Project Serengeti
   http://www.projectserengeti.org/

[7] Scaling the Deployment of Multiple Hadoop Workloads on a Virtualized Infrastructure


[9] Enterprise Java on VMware vSphere – Best Practices
   http://www.vmware.com/resources/techresources/1087

[10] Best practices for Performance Tuning of Latency-Sensitive Workloads in vSphere Virtual Machines
    http://www.vmware.com/resources/techresources/10220


[12] Protecting Hadoop with VMware vSphere 5 Fault Tolerance


[14] VMware vSphere VMFS: Technical Overview and Best Practices

Hadoop Resources
[16] Apache Hadoop Web site:
http://hadoop.apache.org/


Networking Resources
[18] Big Data in the Enterprise – Network Design Considerations
