Technical Report

NetApp Hadoop Solution for IBM InfoSphere BigInsights

Prem Jain, NetApp and Stewart Tate, IBM
October 2012

ABSTRACT

Today, businesses need to manage and control the unprecedented complexity, speed and volume of data. To succeed, they must find ways to manage this big data affordably, while extracting greater business insights from it. They need to empower their employees, innovate, serve their customers, and increase revenue. They must also find a way to manage their rapidly expanding data growth sustainably while finding new ways to get the most possible value from their data. NetApp® and IBM have extended their partnership to develop a reference architecture, based on Hadoop®, to help businesses get control over their data, meet tight SLAs around their data applications, and turn data-driven insight into effective action. NetApp's big-analytics storage solutions deliver the capabilities for ingesting, storing, and managing large datasets with high reliability. IBM InfoSphere® BigInsights™ provides an innovative analytics platform that processes and analyzes all types of data to turn large complex data into insight.

This technical report provides basic guidelines and best practices on how to size and configure the NetApp and IBM reference architecture based on IBM InfoSphere® BigInsights™ on NetApp® Open Solution for Hadoop®.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 INTRODUCTION

Today, businesses need to manage and control unprecedented complexity, speed and volume of their data. To succeed, they must find ways to manage this big data affordably, while extracting greater business insights from it. They need to gain competitive advantage, innovate, serve their customers better, and increase revenue. They must also find a way to manage their rapidly expanding data growth sustainably while finding new ways to get the most possible value from their data.

NetApp and IBM have extended their partnership to develop a reference architecture to help businesses get control over their data, meet tight SLAs around their data applications, and turn data-driven insight into effective action. This reference architecture is built on Hadoop, a powerful, open big data platform that offers the flexibility and scale you need for the long-term.

NetApp's big-analytics storage solutions deliver the capabilities for ingesting, storing, and managing large datasets with high reliability. IBM BigInsights provides an innovative analytics platform that processes and analyzes all types of data to turn large complex data into insight.

The NetApp-IBM Hadoop combination delivers "big analytics" with:

- Quick insights from unstructured, polystructured and structured data with ready to deploy integrated hardware stack and pre-built queries for instant analysis
- Resilient implementation of Hadoop for the enterprise with lower downtime, higher data availability, all of which reduces risk of deploying Hadoop
- Cost effective with higher storage efficiencies, lower operational expenses, and full set of services

NetApp and IBM have collaborated to deliver a unique Hadoop reference design that firms in any industry can deploy for business advantage.

1.1 MEETING THE CHALLENGE OF BIG DATA ANALYTICS WITH IBM INFOSPHERE BIGINSIGHTS

THE VALUE OF THE IBM INFOSPHERE BIGINSIGHTS ANALYTICS

The IBM InfoSphere BigInsights software platform helps firms discover and analyze business insights hidden in large volumes of a diverse range of data – data that is often ignored or discarded because it is too impractical or difficult to process using traditional means. Examples include log records, click streams, social media data, news feeds, emails, electronic sensor output, and even some transactional data. To help organizations process these and other forms of data that are increasing in volume, variety, and velocity, IBM developed BigInsights, a software platform that includes the open source Hadoop framework from Apache.

ANALYSIS AND DISCOVERY

BigInsights helps firms analyze large volumes of document and message content with its built-in text processing engine and library of annotators. Developers can quickly query and identify items of interest – such as persons, email addresses, street addresses, phone numbers, URLs, and business alliances – to understand the context and content of relevant business information hidden in unstructured text data.

In addition, programmers can use the BigInsights Eclipse-based plug-in to create their own text analytic functions. Built-in pattern discovery, expression builders, and a test environment promote rapid prototyping and validation of custom text annotators.

To help business analysts and non-programmers benefit from big data, BigInsights provides a spreadsheet-like discovery and visualization facility. Through the BigInsights Web console, users can collect and integrate a variety of data into a spreadsheet structure, explore and manipulate that data using built-in functions and macros, create charts, and export the results, if desired.

The Web console also features a catalog of applications that authorized users can launch immediately or schedule for future execution. IBM provides pre-built "apps" for Web crawling, importing and exporting data, accessing a public forum search engine, querying BigInsights data, and performing other work. Users can also publish their own applications in this catalog.

ENTERPRISE SOFTWARE INTEGRATION

Integrating BigInsights – and its analysis of big data – with existing enterprise software is one of IBM's key initiatives. That's why BigInsights provides connectivity to popular data warehouse platforms, including DB2, Netezza, and non-IBM offerings. As a result, BigInsights developers can join reference data from a relational database with data managed by BigInsights to refine and expand their analysis.

IBM's approach ensures that your big data and traditional enterprise data won't exist in isolation. Instead, you can use your existing enterprise software platforms to do what they do best: leverage BigInsights for analytical workloads that aren't appropriate or practical for these platforms, and broaden your business analysis capabilities in an integrated fashion.

Using technologies such as Hadoop MapReduce, data analytics can process very large amounts of both structured and unstructured data. In contrast, the traditional relational database (RDB) with structured data is a different tool for a different job. Relational databases are designed for many concurrent users, with many small transactions (such as inventory events, reservations, and banking), with all of the related Structured Query Language (SQL), table, row, column, and join design assumptions. Hadoop and RDB solutions can (and often do) work together in commercial tasks to reduce an ever-expanding ocean of data into useful information.

The NetApp Open Solution for Hadoop was designed to meet several (often conflicting) technical and market demands. The enterprise Hadoop ecosystem is a hybrid of open source and enterprise sensibilities, and many aspects of this solution take its hybrid nature into account.

Hadoop has its origins in distributed computing. This form of computing divides the dataset into thousands of pieces that can be analyzed without intervention from any of the other pieces. This programming or computing style is often called *shared nothing,* and shared-nothing programs run best on hardware platforms that also share little or nothing.

The "divide and conquer" strategy of distributed computing is not new to enterprise customers. A relational database management system (RDBMS) such as Oracle® or DB2 has used parallel technology to parallelize the scanning of large tables to satisfy a particular class of queries. This technology parallelizes the scan, filters and sorts the intermediate results, and merges them into a query result set that is returned to the user. What they care about is time. Hadoop uses *map* (filter and sort) and *reduce* (merge) to accomplish the same effect.

The two components of Hadoop are MapReduce and the Hadoop Distributed File System (HDFS). MapReduce, as already discussed, is the Hadoop framework for parallel processing. HDFS is the distributed file system that provides petabyte-size storage and data management.

1.2 HADOOP DISTRIBUTED FILE SYSTEM

HDFS is optimized to support very large files that typically range in size from megabytes to petabytes. It's better to have a relatively modest number of these large files than to have a large number of smaller files.

HDFS is designed for streaming data access. At load time, HDFS automatically spreads data across nodes in the cluster to facilitate parallel processing of that data at read time. Unlike many traditional file systems, HDFS is designed to minimize the time required to read an entire data set rather than the time required to read the first record or a small subset of the data. Thus, HDFS isn't suitable for applications that require low-latency data access; instead, it's designed to provide efficient, batch access to large volumes of data. HDFS has a simple concurrency model that presumes data will be written once (by one writer) and read many times by various applications. There is no support for multiple concurrent writers or for updating a subset of data at an arbitrary point within a file.

HDFS was designed to provide fault tolerance for Hadoop. By default, data blocks (i.e., portions of a file) are replicated across multiple nodes at load or write time. The degree of replication is configurable; the default replication is three. Replication protects the data and helps Hadoop recover from hardware failures that can periodically occur when using large clusters of low-cost commodity hardware.

A traditional Hadoop cluster consists of a few basic components:

- The NameNode, which manages the Hadoop Distributed File System (HDFS) namespace.
- The Checkpoint node, a secondary NameNode which manages the on-disk representation of the NameNode metadata.
- The JobTracker node, which manages all jobs submitted to the Hadoop cluster and facilitates job and task scheduling.
- Hundreds of slave nodes that provide both TaskTracker and DataNode functionality. These nodes perform all of the real work done by the cluster. As DataNodes, they store all of the data blocks that make up the file system, and they serve I/O requests. As TaskTrackers, they perform the job tasks assigned to them by the JobTracker.

Since Hadoop is built on top of a distributed file system, HDFS, this arrangement allows every DataNode and TaskTracker node to see and access all data across the cluster. The NameNode houses and manages the metadata, so whenever a TaskTracker must read or write an HDFS block, the NameNode informs the TaskTracker where a block exists or where one should be written.

HDFS is a feature-rich, integrated, and distributed file system that offers data protection, fault tolerance, and the ability to balance workloads. Data protection is implemented by using a cluster-aware, software-based mirroring scheme. Unlike a classic triple hardware mirroring scheme, HDFS dynamically determines where a given block's mirror block (replica) will be located. Another innovation of HDFS is that the mirroring scheme can be set on a file-by-file basis. The replication parameter is used to set the mirroring for a given file. The default practice in Hadoop is to set all data files to a replication count of three. Another innovation of HDFS is the ease with which the replication or mirroring level can be adjusted. To change the replication count of a file, you simply issue the `hadoop fs –setrep` command, and Hadoop schedules the additional copies or reclaims the space from mirror copies that are no longer used.

Because HDFS is distributed, all data resides on individual DataNodes. If a DataNode or sections of the HDFS residing on a DataNode become inaccessible or corrupt, then actions are taken to redirect access to the affected data to one of the other copies, and to initiate repairs in the event of block corruption.

HDFS can provide job completion in the face of hardware failures because copies of the data are located on other nodes. If a node or disks in a node fail where a portion of a MapReduce job (task or set of tasks) was running, the affected tasks are all relocated by the JobTracker to other nodes where the data does still exist and are then re-executed. In addition to data protection, HDFS mirroring ensures that MapReduce jobs complete successfully regardless of disk or node failures. Loss of service rarely involves the destruction of data, but HDFS handles both forms of loss.

1.3 TRIPLE-MIRROR TRADE-OFFS

Although HDFS is distributed, feature rich, and flexible, there are both throughput and operational costs associated with server-based replication for ingest, redistribution of data following recovery of a failed DataNode, and space utilization.

Costs of triple replication include:

- For each usable terabyte (TB) of data, 3TB of raw capacity are required.
- Server-based triple replication creates a significant load on the servers themselves. At the beginning of an ingest task, before any data blocks are actually written to a DataNode, the NameNode creates a list of DataNodes where all three of the first block replicas will be written, forming a pipeline and allocating blocks to those nodes. The first data block arrives over Ethernet; traverses the server Southbridge, Northbridge, and memory bus; and is written to RAM. After being passed to the storage drivers, the data then traverses back across the Northbridge and the Southbridge and is eventually written out to drives. That DataNode forwards the same block to the next node in the pipeline and the preceding procedure is repeated. Finally, this second DataNode sends the block to the third DataNode and the write to storage follows the same I/O path as previous writes. This results in a significant load on server resources.
- Server-based triple replication also creates a significant load on the network. For ingest, three network trips are required to accomplish a single block write. All compete for and consume the same network, CPU, memory, and storage resources allocated to process the Hadoop analysis tasks. The NetApp solution can provide better reliability with double replication or needing only two copies of data.

With the advent of more powerful entry-level servers, modern Hadoop clusters are evolving into a hybrid of symmetric multiprocessing (SMP) and massively parallel processing (MPP) supercomputing clusters. A typical low-cost DataNode now comes with 8 to 12 CPU cores, following a trend toward increased capability that is driven by Moore's Law (which observes that the number of transistors on a given chip doubles approximately every two years) and by the highly competitive x64 server market. A cluster-wide MapReduce job might have 16 tasks running on each of 64 nodes. This modest cluster has a degree of parallelism of 1,024 (16 x 64). This is a core concept to Hadoop, so most customers quickly understand how much computing they can bring to bear on the analysis of their data. What they often overlook is the fact that this computing power can (and must) be used to load the data as well.

Anything done to reduce the consumption of server and network resources during the loading of data increases cluster efficiency.

1.4 WHEN DRIVES FAIL

It is inevitable that the disk drives in a Hadoop cluster will fail at some point, and the larger the cluster, the more likely it is that a drive failure event will occur. Although most customers do not deploy 5,000-node clusters, a cluster of this size could contain 60,000 disks. Even though means are not a good metric for drive failure, mean time between failure data suggests that a drive will fail every few hours in a 60,000-spindle deployment. In a modest cluster, the number of failures is much lower, but if one or more drives fail, the entire DataNode may be removed from service. At the very least, all tasks accessing data on that drive will fail and be reassigned to other DataNodes in the cluster. This in itself can result in severe degradation of completion time for running jobs.

In addition to that, the drive must be physically replaced, mounted, partitioned, formatted, and reloaded with data from nearby copies before HDFS can make full use of that drive. Repopulating the new disk with data can be accomplished by using the Hadoop balancer, which redistributes HDFS data across all DataNodes, by moving blocks from over-utilized to under-utilized DataNodes. The balancer runs in the background and, by default, it copies only 1 MB/sec. That default copy rate can be modified by setting the `dfs.balance.bandwidthPerSec` property. Based on that setting,

rebalancing to repopulate a replaced 2TB disk could take hours to days, depending on allowed resource utilization. Either way, the process diverts network bandwidth away from running jobs. Also, remember that HDFS is able to generate only about 30MB/sec per SATA disk, making disk I/O bandwidth another limited resource, which is affected by the balancer.

For every full 2TB SATA disk that fails, approximately 70 hours or more is required for rebalancing the distribution of blocks following replacement of that disk, depending on tolerance for network resource consumption. The tradeoff is between rebalancing time, network utilization, and disk I/O bandwidth utilization. Remember that higher resource utilization affects MapReduce job performance, especially during data ingest operations. In some cases, rebalancing may also require some manual movement of blocks between disk drives.

1.5 REFERENCE ARCHITECTURE FOR BIG DATA ANALYTICS

As enterprises begin to move Hadoop deployments out of the lab and into production, they typically expect a certain degree of scalability and efficiency. NetApp Open Solution for Hadoop uniquely complements the IBM feature-rich BigInsights software suite to create an end-to-end solution that focuses on functionality, efficiency and quality that are so critical.

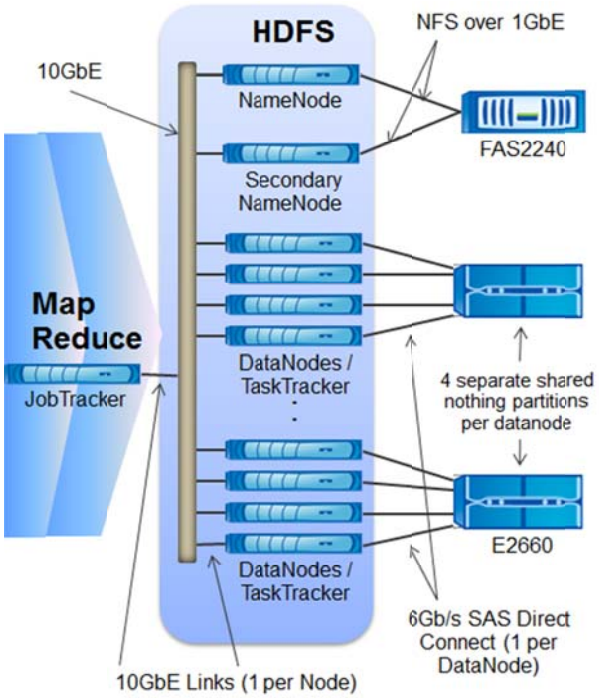


Figure 1) NetApp Integration with Hadoop

The NetApp Open Solution for Hadoop (Figure 1) addresses two operational limitations of current Hadoop clusters by:

- Reducing the operational and functional costs associated with distributed, software-based mirroring (block replication) with a replication count of two, instead of three.
- Addressing the most significant single point of failure in Hadoop, the NameNode, by maintaining an up-to-date copy of all HDFS metadata, external to the NameNode server. This enables fast recovery of HDFS after a NameNode server crashes.

The NetApp Open Solution for Hadoop includes the following components:

Hardware

- **Hadoop data storage array.** One NetApp E2660 storage array serves data for four DataNodes. Using RAID 5, the storage array is able to protect the data and minimize the performance impact of disk drive failures.
- **Hadoop metadata storage.** NetApp FAS2240 or other FAS2000 series storage array provides a NetApp Data ONTAP® grade of data protection for the NameNode and a single, unified repository for Linux, Hadoop, and other software (and scripts) to simplify deployment, updates, and maintenance.

Software

- **NFS license** for FAS2000 series storage system
- Hadoop distribution included in IBM InfoSphere BigInsights

1.6 BENEFITS OF THE NETAPP IBM INFOSPHERE BIGINSIGHTS SOLUTION

The core benefit of the NetApp storage and IBM BigInsights combination is an enterprise-grade HDFS platform for implementing Hadoop that quickly and easily converts all their types of data into business insights:

- Lower cluster downtime with transparent RAID operations and transparent rebuild of failed media
- Less storage needed with replication count of two
- Robustness of HDFS via reliable NFS server based Metadata Protection
- Performance for the next generation of DataNodes (SMP) and networks (10GbE)
- Highly reliable and fully manageable BigInsights Hadoop Platform
- Proven server, storage and networking hardware based on enterprise-grade designs, offering enterprise-grade performance, uptime, and support
- Fully online serviceability
- Industry-leading analytics tooling
- Stable, scalable and supported production-ready code from the open source community
- Business process accelerators (“Apps”)
- Comprehensive set of analytical tools including leading edge text analytics, and spreadsheet-style analysis tools
- RDBMS, warehouse connectivity
- Easier and simpler administration with a Web-based console, flexible job scheduler, LDAP authentication

2 SOLUTION SUMMARY

The combination of NetApp Open Solution for Hadoop and IBM BigInsights provides industry-standard enterprise data protection through the NetApp Data ONTAP-based NFS storage and IBM InfoSphere BigInsights Management layer.

The NetApp E-Series storage arrays provide storage services to DataNodes. The E2660 is not a shared-storage infrastructure as Hadoop is a shared-nothing programming paradigm. Each DataNode accesses its own dedicated set of 15 disks from the E2660 array. In Hadoop, the compute is brought to the data. The NetApp and BigInsights reference architecture described in Section 3 of this document acknowledges the shared-nothing nature of Hadoop by providing each DataNode with two dedicated RAID-protected volumes with enterprise-grade capabilities often found in a traditional storage area network (SAN) environment.

With support for the latest Intel Xeon processor technology, the IBM System x3630 M4 offers cost-optimized performance and memory with better energy efficiency. The x3630 M4 provides a simpler, more affordable alternative to traditional enterprise offerings without sacrificing performance.

The x3630 M4 offers added flexibility and an innovative design to help you more easily and cost effectively add new features as your requirements change. Depending on the model you select, these can include graphic card support, advanced RAID and light path diagnostics, hot-swap hard disk drives, rich remote-management capabilities and a shared or dedicated port for system management. Expandable PCIe slots and NIC ports let you pay for new capabilities as you need them.

2.1 DATA PROTECTION AND JOB COMPLETION

HDFS provides data protection and other benefits by implementing host-based, software triple mirroring. For every block of data loaded into HDFS, two copies are written to two other nodes in the cluster. This is a replication count of three, and it is the default or standard practice for most Hadoop deployments that use standard "just a bunch of disks" (JBOD) SATA DAS disks co-located in a DataNode. HDFS is one of the first file systems to implement a replication strategy on a file-by-file basis. This flexibility allows customers to adjust the protection level to suit particular datasets.

HDFS replication provides data protection, job completion in the event of disk failure or DataNode failure, and load balancing. Replication for an entire cluster can be set by using a single parameter, `dfs.replication`, which has a default value of three. It can also be set for an individual file at the time of file creation, and existing files can be changed by using the **`hadoop fs –setrep`** command. The higher the HDFS replication count, the more copies of the data must be stored, but the more resilient and balanced the access is to that data. The limitation is the same found with all data-based (versus parity-based) protection schemes: storage utilization efficiency. When a customer deploys with the default replication count of three, for every 1PB of data loaded into a Hadoop cluster, another 2PB must be copied across the network fabric to other nodes. High-replication-count files consume a lot of space and take a long time to ingest because 2PB of the 3PB must be written over the network. The NetApp E2660 storage platform offers a more efficient approach to significantly reduce the overhead of data protection, making HDFS more efficient and faster (certainly on loads).

This is the single most important feature of NetApp's solution: offloading some or even all of the data protection aspects of HDFS into enterprise-grade, industry-proven E-Series controllers. The second most important reason for having replicated copies is for job completion. If job tasks on a given DataNode fail for any reason, the JobTracker must reassign the failed tasks to other nodes in the cluster. Eventually the JobTracker may blacklist the node and the NameNode may recognize the node as being in a failed state. If task reassignment is not possible, then the MapReduce job cannot complete.

It is important to understand that this risk window is open only during a MapReduce job. This job-completion mechanism can function only if there are one or more copies of the data in the cluster. NetApp advises that customers can run safely with a replication count of two. The data is already protected by using RAID 5 on the E2660 logical volumes used for data storage

There may be use cases where a replication count higher than two benefits MapReduce job performance, especially when multiple jobs that are using the same data run simultaneously. In that case, higher replication counts may offer greater parallelization of job tasks while leveraging HDFS data locality. Higher replication counts also enable the data scientist to better leverage the Hadoop "speculative execution" feature, which can help balance task execution across resources in a cluster.

Speculative execution enables the JobTracker to assign instances of a relatively slow-running task to other TaskTracker nodes. In this scenario, when the first instance of the task finishes, the other

instance is killed and its results are discarded. This feature can benefit MapReduce performance in clusters where unused node resources exist, but it may actually be bad for performance in very busy clusters. Given all the pros and cons of various replication counts discussed earlier, with the NetApp Open Solution for Hadoop, a replication count of three is no longer a requirement for data protection. A setting of two is sufficient and represents a good balance between performance and data protection for a very large number of Hadoop use cases. As with all database and analytics systems, MapReduce jobs often benefit from tuning at both the system and job code levels. Replication count is one of many tuning parameters to consider.

The combination of removing and reducing most (but not all) of the risk means that MapReduce jobs can run successfully at HDFS with a replication count of one. Keep in mind, however, that NetApp strongly recommends a base replication count of two. A replication count of one greatly increases the risk that a MapReduce job will fail and introduces a very small amount of risk that data could actually be lost. That risk should be carefully weighed against any perceived advantages of disabling replication. A replication count of one also greatly affects the ability of job tasks to leverage data locality, resulting in significantly higher network utilization, because more data blocks must be accessed over network fabric.

3 SOLUTION ARCHITECTURE

3.1 HARDWARE REQUIREMENTS

The NetApp Open Solution for Hadoop and IBM BigInsights reference architecture is comprised of the following hardware and software components

SERVERS

IBM System x3630 M4 Servers

- Support all Intel Xeon E5-2400 Series processors
- Up to two 2.4GHz (6-core) or 2.3GHz (8-core) processors
- Memory: 12 DIMMs (up to 192GB) Up to 1600Mhz
- Hard Disks: 2 x 3.5" HS HDDs
- Network: 2 x 1 Gbps Ethernet ports, 10 Gb Network port
- LSI 9200-8e SAS HBA
- SFF-8088 external mini-SAS cable (for maximum signal integrity, cable should not exceed 5m)

STORAGE

NetApp E-2660 Storage Subsystem:

- 1 DE6600 4U chassis, with rail kit and power cords
- 2 E2600 series controllers, each with four eSAS ports and 2GB memory per controller
- (60) 2TB or 3TB, 7.2K-RPM, 3.5" near-line SAS disks
- NetApp Engenio® Operating System (EOS CFW 10.80 or later release)
- Base SANtricity® Storage Manager
- Turbo feature enabled

NetApp FAS 2240 NFS Storage System

NETWORK

- IBM BNT G8264-E 10Gb Enhanced Ethernet Switch
- IBM BNT G8052 1-GbE Switch

3.2 SOFTWARE REQUIREMENTS

- IBM InfoSphere BigInsights Hadoop Software Suite
- RedHat Enterprise Linux RHEL5.7 or above

3.3 SOLUTION ARCHITECTURE DETAILS

The NetApp storage and BigInsights reference design has two major hardware components. The FAS2240 provides enterprise-grade protection of the NameNode metadata, and the NetApp E2660 storage array provides the data storage services to four Hadoop DataNodes.

Each DataNode has its own dedicated, nonshared set of disks. This exactly follows the way in which Hadoop DataNodes configure capacity when these drives are physically co-located in the chassis.

In practical terms, Linux is not able to distinguish between a set of logical unit numbers (LUNs) presented by the LSI 1068 controller chip on the HBA, and a set of disks presented by either an embedded SAS or a SATA controller. There is no logical difference. However, there are differences in performance and reliability. This set of disks is configured as two blocks, each using one of two parity-

based protection schemes. Eight volume groups are configured in the E2660, and they are configured so that each DataNode sees only its set of two private blocks. This design is an improved alternative to packaging JBOD DAS media for four DataNodes, offering better performance, reliability, data availability and uptime, and manageability.

Each block is a RAID-protected volume group that contains a single virtual logical volume, which can be created and exported to a given DataNode as a LUN. These LUNs appear as physical disks in Red Hat Enterprise Linux (RHEL). They can be partitioned, and file systems can be created and mounted to store HDFS blocks. (HDFS blocks can be 64MB, 128MB, 256MB, or even 512MB in size.)

NETWORK ARCHITECTURE

The network architecture of a Hadoop cluster is critical. With a default replication count of three, more than two-thirds of all I/O traffic must traverse the network during ingest, and perhaps a third of all MapReduce I/O during processing runs could originate from other nodes. The NetApp E2660 storage modules provide a level of performance that is significantly better than JBOD SATA. The E2660 offers a well-designed, enterprise-class performance platform based on four 6Gb/sec SAS ports, supported by proven caching and best-in-class hardware RAID.

The solution uses two network backplanes. The core of this solution is the HDFS network which is a Gb network backplane served by very high performance IBM BNT /10/40Gb Switch Solution for Top of Rack switch. The other network is a Gb Ethernet network, which caters to the administration network and also connects the NameNode, JobTracker and the NetApp FAS storage systems.

On the 10GbE network, all components should be configured for jumbo frames. This requirement also applies to the switch ports to which the components are connected. Any ports on the 10GbE switches that are configured for GbE connections (such as FAS2240 NFS and uplinks) should not have jumbo frames enabled.

BigInsights Cluster Network

- Data Network
  - Private interconnect between all BigInsights nodes
  - Uses its own VLAN and subnet
  - Primary purpose: Data ingest and movement within cluster

- Admin Network
  - System administration network
  - Isolated VLAN and subnet
  - Primary purpose: Admin via ssh/rsh into cluster

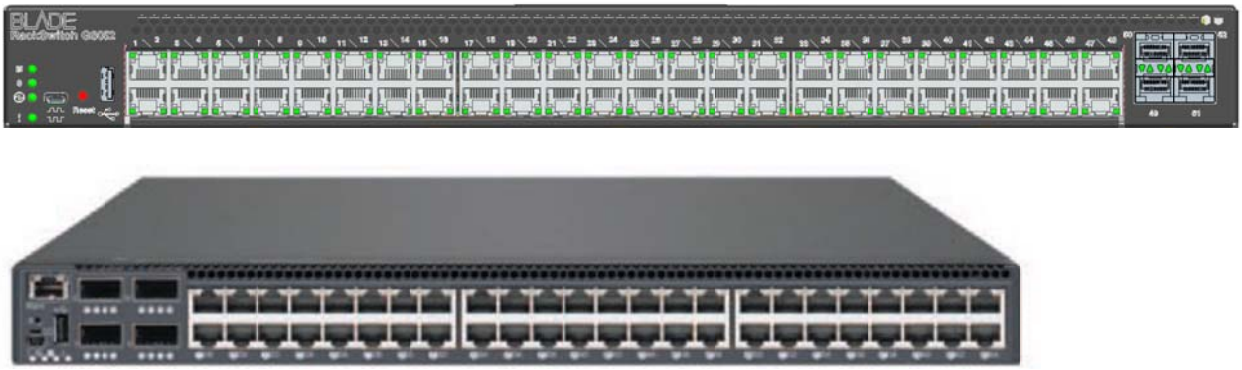
- Management Network
  - IBM Integrated Management Module
  - Isolated VLAN and subnet
  - Primary purpose: Hardware monitoring

NETWORK SWITCH

Designed with top performance in mind, the IBM BNT RackSwitch G8264 is ideal for today's big data optimized workloads. They are enterprise-class and full-featured data-center switches that deliver line-rate, high-bandwidth switching, filtering, and traffic queuing without delaying data. Large data-center grade buffers keep traffic moving. Redundant power and fans along with numerous high availability features equip the switches for business-sensitive traffic. The RackSwitch G8264 is ideal for latency-

sensitive applications such as high performance computing clusters and financial applications. The G8264 supports IBM Virtual Fabric to help clients reduce the number of I/O adapters to a single dual-port 10 Gb adapter, helping reduce the cost and complexity. The RackSwitch G8264T can be leveraged as part of a comprehensive 10GBase-T solution offering from IBM, which includes servers, storage and networking, providing better virtualization, better management, and a cost- effective option for next-generation data centers. Flexible connectivity across distances up to 100m at a low cost makes the G8264T an optimal choice for connecting high-speed server and storage devices.

Figure 2) IBM BNT RackSwitch G8264



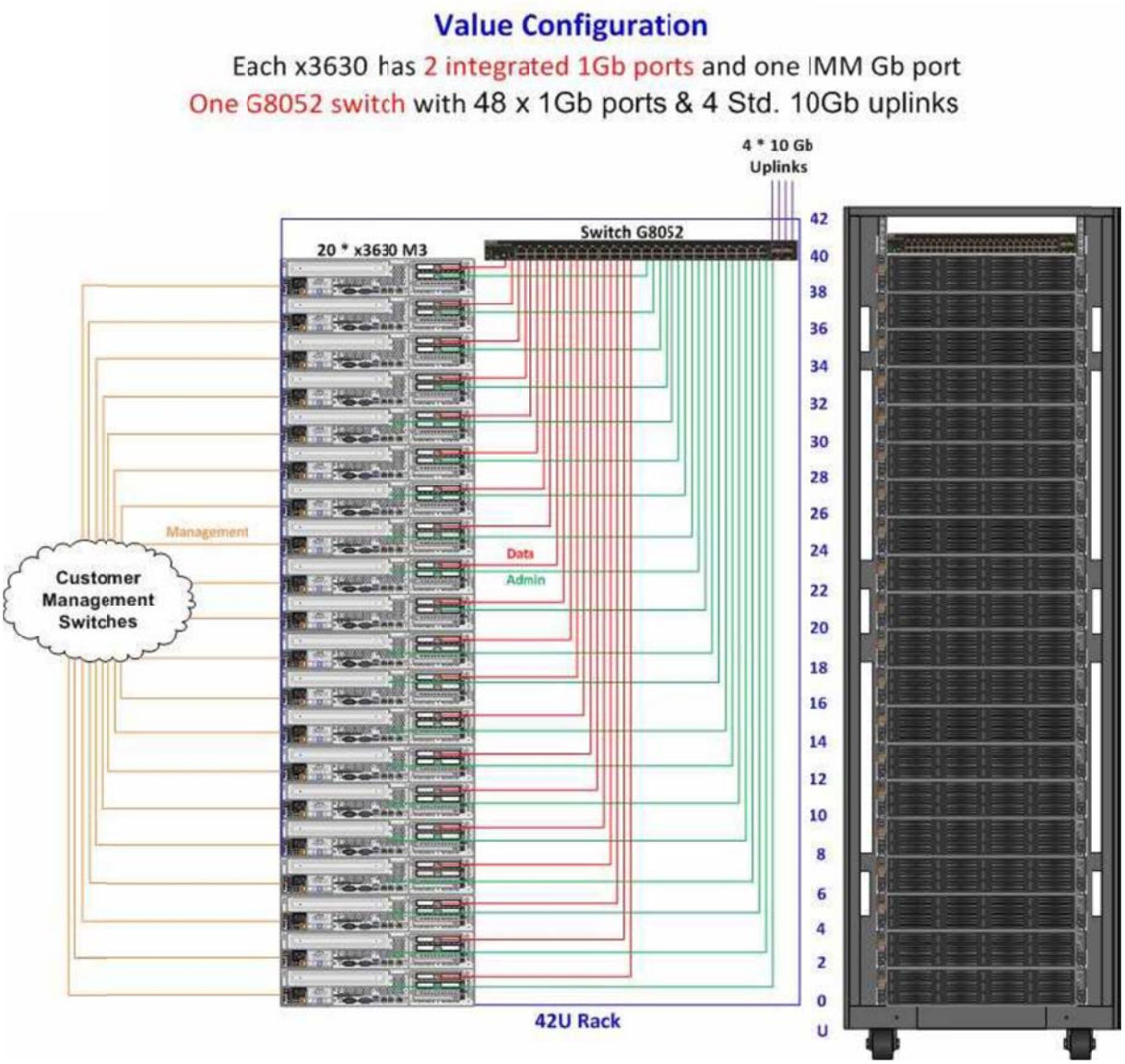
CABLING TOPOLOGY

The server-to-storage-array cabling is very simple. Each DataNode requires a two-port eSAS HBA. Only one of these ports is used to connect the DataNode to the eSAS port on the E2660 that has been designated for this DataNode. The cable length is typically 1m, but it can be up to 5m. The eSAS specification indicates that cables can be up to 10m, but for a variety of manufacturing reasons, the longest cable will probably be 5m.

The solution was designed to be compact and to minimize the colocation rackspace footprint. In a vertical racking deployment, an E2660 would be found in the lower 16U of a rack (because of weight), and the remaining 16U would be for 16-1U servers. Most racks are not populated above the 32U line. In this rack example, 2m cables would be sufficient for all DataNodes to connect to their respective E2660s.

Horizontal deployment is another racking topology. In this topology, the E2660s are in a dedicated rack and the servers are in the adjacent rack. In this case, 0.5m cables would be adequate.

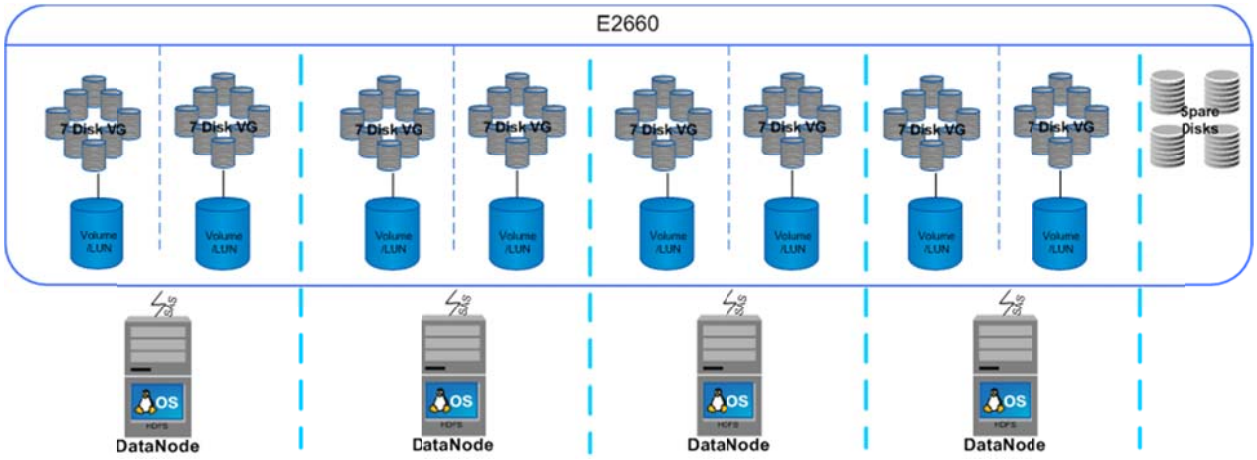
Figure 3) Value Configuration



STORAGE ARCHITECTURE

As much as possible, the E2660 operates as eight completely independent storage modules. Each array is configured as eight independent RAID groups of seven disks that can be set up in a RAID 5 (2 x 6+1)) configuration. This consumes 56 disks (8 x 7). The remaining four disks are global hot spares. If customers deploy all of their files with replication count of two, then using a single-parity drive over six spindles provides a good balance between storage space utilization and RAID protection. As a result, there are constantly two forms of protection when the E2660s are running with files set to replication count of two.

Figure 4) E2660 configured with replication count 2



STORAGE SIZING AND PERFORMANCE CONSIDERATIONS

The amount of usable space in HDFS varies by the replication count for any given set of files. The standard Hadoop cluster usually runs all files at replication count of three. This results in having only one-third of all the file system space available for use. Typically, HDFS clusters are measured by their available space and usable or loadable space. A customer with 9TB of available space mounted as file systems can load or use only 3TB of data when using replication count of three. The NetApp Open Solution for Hadoop allows customers to run at replication count of two, increasing the amount of loadable space per file system by about 50% compared to HDFS configured with replication count of three.

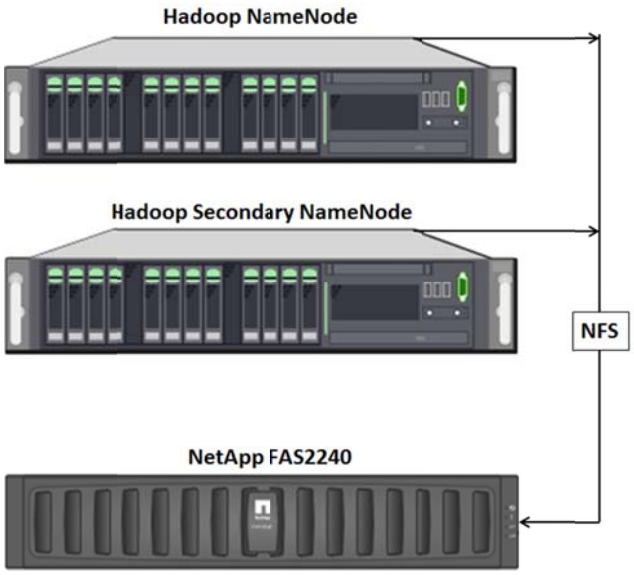
Storage must also be allocated for temporary Map task output. Map task output is usually written to local Linux file systems, not HDFS. A very common practice is to spread the Map I/O across the same disks used by HDFS. The NetApp Open Solution for Hadoop follows this practice as well. The accepted rule of thumb is to allocate 30% of storage capacity for Map output. This must be considered in sizing storage for HDFS.

NAMENODE METADATA PROTECTION

As discussed earlier, this solution offers a unique feature of NameNode metadata protection and the robustness and reliability of an enterprise grade product. This protection is offered by introducing a NetApp FAS2240 NFS storage system. The NameNode is configured to keep a copy of its HDFS metadata (FSImage) into the NFS mounted storage. The same NFS mounted storage is also made available to the secondary NameNode.

In case of a NameNode failure, the critical metadata is still safe on the NetApp FAS storage system. At this point, we can recover the metadata and restart services on the secondary NameNode, to get the HDFS back and running promptly.

Figure 5) NameNode configuration stores metadata on NFS mounted storage



RACK AWARENESS IMPLEMENTATION

We chose to implement rack awareness in our Hadoop solution because of the benefits it offers customers. Using this feature, we are able to configure Hadoop to know the topology of the network.

This is important for two reasons:

- In most rack network configurations, more network bandwidth is available within a single rack than between racks. Rack awareness enables Hadoop to maximize network bandwidth by favoring block transfers within a rack over transfers between racks. Specifically, with rack awareness, the JobTracker is able to optimize MapReduce job performance by assigning tasks to nodes that are "closer" to their data in terms of network topology. This is particularly beneficial in cases where tasks cannot be assigned to nodes where their data is stored locally.
- By default, during HDFS writes, the NameNode assigns the second and third block replicas to nodes in a different rack from the first replica. This provides data protection even against rack failure; however, this is possible only if Hadoop has been configured with knowledge of its rack configuration.

If rack awareness is not configured, all DataNodes in the cluster are assumed to be in the same rack, which is named "default-rack." Follow the steps outlined in the Operational Procedures for assigning rack awareness to each node.

Care must be taken to ensure that for each HDFS block, the corresponding replica is stored on another E2660 controller. In multi-rack clusters, providing the actual rack location of each DataNode will accomplish this. If all the DataNodes are located in the same rack, "rack" mapping must be extended to reflect the E2660 enclosure or E2600 controller used for DataNode storage. Basically, use the following guidelines to prevent data loss in the unlikely event of storage controller failure:

- For multi-rack clusters, define the topology based on the actual rack location of each DataNode.
- For single rack clusters with an even number of E2660 enclosures, define the topology based on the E2660 enclosure used for storage by each DataNode.
- For single rack clusters with an odd number of E2660 enclosures, define the topology based on the E2600 controller used for storage by each DataNode.

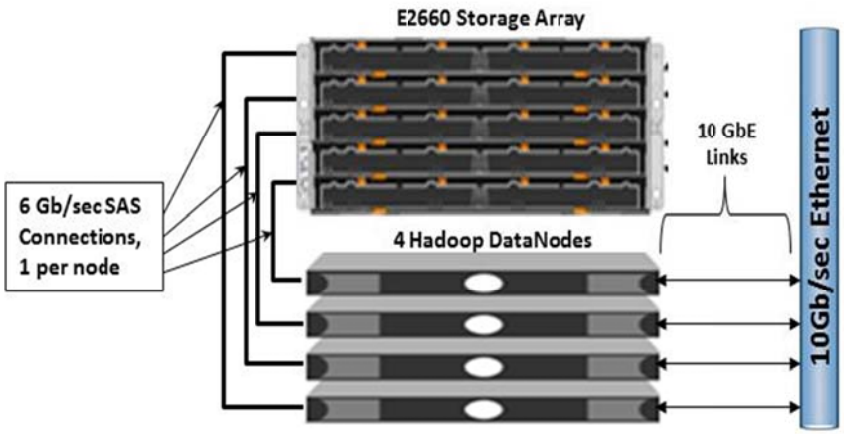
In addition to the above guidelines, care must also be taken not to define network topologies that will result in unbalanced I/O across storage controllers. Consider a scenario where the defined topology results in one or more DataNodes receiving a higher number of block copies than others. For example, the network topology for a four DataNode cluster could be mistakenly configured with three nodes on one rack and one node on another rack. In that configuration, the second replica from three nodes would be copied to the fourth node with the second replica of the fourth node being copied to one of the other nodes. The storage controller supporting the fourth node would quickly become a bottleneck during data ingest. This could also introduce a bottleneck in read performance by impacting the ability of the JobTracker to leverage data locality in the assignment of job tasks. The example given above is very simplistic, and the scenario unlikely for a four DataNode cluster, but as the node count increases, so does the likelihood that this could happen. Also, keep in mind that since the defined network topology determines replica placement, rack awareness must be set up at the very beginning, before any data is ingested. Otherwise, loss of data will likely occur in the unlikely event of storage controller failure.

4 THE NETAPP AND IBM BIGINSIGHTS REFERENCE DESIGN AS TESTED

The tested configuration for the NetApp Open Solution for Hadoop consisted of the following:

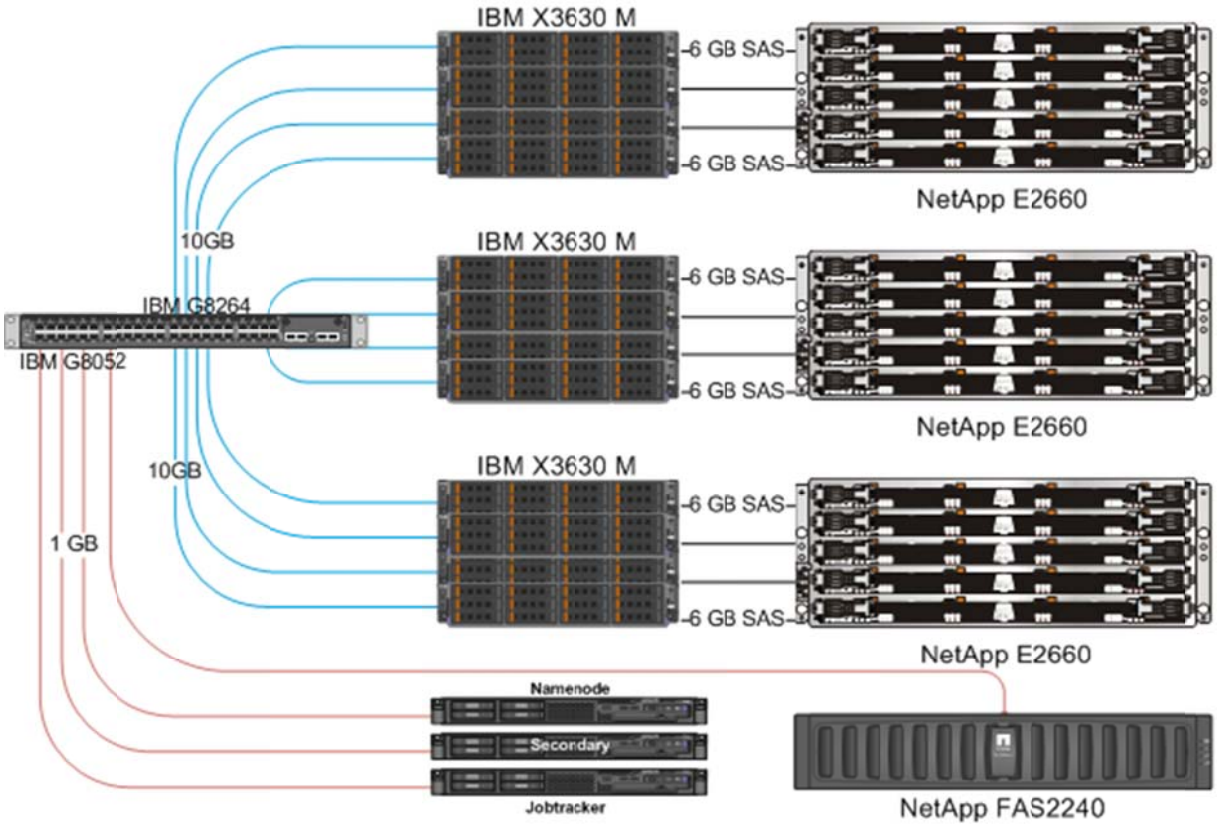
- Three HDFS building blocks (single building block illustrated in Figure 6, each consisting of
  - One NetApp E2660 storage array containing 60 2TB or 3TB NL-SAS disk drives
  - Four IBM x3630 M4 servers, each connected to the E2660 with a single 6Gb/sec SAS connection (one connection per DataNode)
- One NameNode server, IBM x3630 M4
- One secondary NameNode server, IBM x3630 M4
- One JobTracker server IBM x3630 M4
- One NetApp FAS2240 using a single FAS controller equipped with 6-1TB SATA disks
- All nodes (DataNodes) interconnected with 10GbE
- NameNode and secondary NameNode. Jobtracker and FAS2240 connected via GbE (1Gb/sec)

Figure 6) Single HDFS building block



Testing for the NetApp Open Solution for Hadoop was conducted based on the objectives detailed in section 4.1 below. The test cases specific to these objectives are described in section 4.2, "Solution Testing." An underlying objective was to demonstrate the scalability of our solution. Therefore, we chose to test a Hadoop cluster containing six HDFS building blocks. Figure 3 shows the configuration that we actually tested.

Figure 7) The NetApp Open Solution for Hadoop tested configuration



4.1 SOLUTION OBJECTIVES

Quality assurance testing was performed to meet the following key objectives:

- Verify that HDFS could be successfully implemented on servers using NetApp E-Series storage arrays as disk storage
- Demonstrate the ability to capture a backup copy of the HDFS (NameNode) metadata, in real time, on a NetApp FAS2240 storage array through a mounted NFS file share
- Test the resiliency of the HDFS during simulated E-Series storage array disk failure
- Demonstrate the ability to restore the HDFS metadata from the FAS2240 storage array, thereby restoring the proper state of the HDFS after a simulated Hadoop NameNode crash

4.2 SOLUTION TESTING

The pass/fail criteria for test cases 1-A, 1-B, and 2 were successful Hadoop MapReduce job completions with no Hadoop task failures. Because task failures are common in the Hadoop processing environment, Hadoop was designed to recover gracefully from transient task failures. Zero tolerance was chosen so that even failures that might look routine would not be dismissed without proper root cause analysis.

The pass/fail criteria for test case 3 was to have a complete file system metadata recovery from the backup copy on the FAS2240 storage array, and to successfully bring the HDFS back online.

Testing was divided into four test cases to verify that the objectives (described in section 4.1, "Solution Objectives") were met.

TEST CASE 1

This test case was used to initially verify that all solution components and related SAS and network interconnects were working according to the architectural design. It was also used to verify that all of the Hadoop software components were properly installed and configured. This test case was used as a baseline test to verify that Hadoop MapReduce jobs could be repeatedly run and rerun with consistent results. Baseline job timing was recorded.

TEST CASE 2

This test case was performed by running the same MapReduce jobs performed in test case one. For this test, a disk was failed in each E2660 Array, resulting in a total of three failed drives, one drive on each of three E2660 storage arrays. MapReduce job timing was recorded for comparison to those of the baseline runs.

TEST CASE 3

This test case was performed to demonstrate that it was possible to recover the Hadoop file system after the data corruption at the HDFS NameNode. The loss was simulated by halting the NameNode server while a TeraSort MapReduce job was running and deleting the metadata directory. An HDFS check was performed before and after the recovery efforts and compared to verify file system integrity. A complete TeraSort MapReduce job was rerun to demonstrate that the entire file system was intact and that results consistent with previous runs would be achieved.

4.3 TEST ENVIRONMENT CONFIGURATIONS

This section summarizes the configurations of the various components in the Hadoop solution, as tested.

E-SERIES STORAGE ARRAYS

Each of three E2660 storage arrays had the following configuration:

- **Vendor:** NetApp E-Series model E2660
- **Enclosure:** DE6600 with 5 drawers holding 12 disks each (60-disk total capacity)
- **Controllers:** E2600 (quantity 2)
- **Array-to-host interface:** 6GB/sec SAS 4-port, 1 per controller
- **Disk type:** 2TB SAS dual-port enterprise drives (quantity 60)
- **Controller firmware:** 07.77.19.00 (minimum requirement for solution)
- **Controller management:** AMW version 10.77.G0.17 (minimum requirement for solution)
- **Required E2660 premium features:**
  - High-performance tier
  - SANShare storage partitioning

Each E2660 storage array supports four Hadoop DataNode servers through a single SAS connection.

Each array allocates eight volume groups (RAID sets). The supported RAID type is RAID 5. Each volume group is composed of seven disks. Two volume groups are assigned to each DataNode server.

The remaining four disks in the array are designated as “hot spares” in case of disk failure.

One volume is configured to use all available storage space in each volume group. Each volume is configured with a segment size of 512KB to maximize use of disk streaming I/O capabilities. Two volumes are mapped as LUNs to each of the Hadoop DataNode servers to which they are assigned.

In this test environment, the seven-disk RAID 5 volume group yields a single LUN with approximately 10.913TB of usable storage. Two LUNs per node give each server about 21.826TB of storage space.

Table 1) E2660 storage configuration parameters

Storage Parameter	Recommended Setting	Default Setting	Description
Cache block size	32KB	4KB	Block size for E2660 read/write cache. 32KB is the largest size available and was chosen to optimize I/O performance for the large block I/O generated by HDFS. This parameter is set for the entire E2660 array.
Read cache	Enabled	Enabled	Enables caching of prefetch data blocks to improve read performance. Read cache is set for each individual volume.
Write cache	Enabled	Enabled	Enables caching of writes, thereby improving write performance. Write cache is set for each individual volume.
Write cache without batteries	Disabled	Disabled	By default, batteries provide backup power to cache memory to avoid data loss during a power outage. This parameter is set at the volume level.
Write cache with mirroring	Disabled	Enabled	Maintains cache consistency between controllers to enable controller failover, but results in less available write cache and increased operational overhead. The NetApp Open Solution for Hadoop does not use the controller failover capability, so this capability is disabled. It is enabled or disabled on a per-volume basis.
Dynamic cache read prefetch	Enabled	Enabled	Provides dynamic, intelligent read-ahead for improved read performance. Enabled at the volume level.
Volume segment size	512KB	128KB	The amount of I/O written to a single disk drive before moving to the next drive in a RAID array. The largest segment size available was chosen to optimize data layout for the large block I/O generated by HDFS. The segment size is set for each individual volume.

HADOOP SERVERS

Fifteen IBM x3630 M4 servers were employed in the solution tested. The server utilization is shown in Table 2.

Table 2) Server utilization

Server Function	Quantity
Hadoop NameNode	1
Hadoop secondary NameNode	1
Hadoop JobTracker node	1
Hadoop DataNode/TaskTracker	12

Each of the servers, as tested, had the following configuration:

IBM System x3630 M4 Servers

- **CPU:** Dual Intel® Xeon® E5620 4C/8T 2.40GHz
- **RAM:** 6x8 GB DDR3 1333 MHz (48GB total)
- **Internal disk:** 1TB SATA disk drive (quantity 2)
- **SAS adapter:** LSI 9200-8e SAS controller 6GB 8-port (used by DataNodes only)
- **Network interfaces:**
  - Two GbE (1Gb/sec) integrated ports
  - One 10GbE 2-port NIC
- **Power:** Dual power supply
- **OS:** Red Hat Linux Enterprise Server (RHEL), version 5 update 6

  Note: All server firmware and BIOS were upgraded to the latest versions. **Note:** LSI SAS adapter minimal firmware and drivers:

  - **Firmware:** 9.00.00.00 (rom525fx.bin)
  - **BIOS:** 6.30.00.00
  - **X86-BIOS:** 07.07.00.00
- **Linux driver:** 9.00.00.00 (mpt2sas.ko)
- **LUN partitioning**
  - The Linux `parted` utility was used for LUN partitioning. We used `gpt` labels because our LUNs are larger than 2TB in size. For proper alignment with the underlying RAID 5 configuration, we started our partitions (using `mkpart`) at 6144 sectors and ended them at 12.0TB to use the entire LUN. We also specified xfs as our file system type, which is discussed in in the next section under "Local File System Configuration." Below is an example of the command used to partition a LUN with device name `/dev/sdc`:

```
# parted –s /dev/sdc mklabel gpt mkpart /dev/sdc1 xfs 6144s 12.0TB
```

Local File System Configuration

As previously stated, HDFS uses native local file systems for actual block storage. For Linux Hadoop configurations, ext3 has been very popular; however, much progress has been made in the area of file system performance. We found xfs to be a good match for our data storage configuration in terms of performance and reliability. Lab tests showed that a log size of 128 MB performed well with our configuration. We also found use of "lazy-count" logging to be good for performance. Lazy-count logging reduces the overhead of file system logging by safely reducing

the frequency of superblock metadata updates. Stripe unit (`su`), stripe width (`sw`), and real-time section size of the file system (`extsize`) were tuned to match our underlying RAID subsystem and I/O patterns.

The following example of the command is used to create a file system with label name `DISK2` on a partition with device name `/dev/sdc1`:

```
# mkfs.xfs -f -L DISK2 -l size=128m,lazy-count=1 -d su=512k,sw=6 -r extsize=256k /dev/sdc1
```

Xfs File System Mount Options (/etc/fstab entries)

Below is a sample of `/etc/fstab` entries used for mounting our xfs file systems:

```
LABEL=DISK1  /disk1    xfs    allocsize=128k,noatime,nobarrier,nodiratime       0      0
LABEL=DISK2  /disk2    xfs    allocsize=128k,noatime,nobarrier,nodiratime       0      0
```

Use of the `noatime` and `nodiratime` options results in access file and directory time stamps not being updated. These options are commonly used, and NetApp recommends them for HDFS.

Write barriers make sure of the file system integrity for devices using write caches, even during a power loss. This is not necessary with battery-backed write cache, which is standard with the E2660 array. That being the case, we were able to improve write performance by using the `nobarrier` option to disable write barriers,

Finally, setting `allocsize` to 128KB helped to optimize performance with HDFS I/O. The `allocsize` option sets the preallocation size for buffered I/O.

NETAPP FAS2240 STORAGE

The FAS2240 storage array, used for backup storage of Hadoop HDFS metadata, was configured as follows:

- **Controllers:** Quantity 1
- **Disks:** 1TB SATA (quantity 6)
- **Network:** Onboard 1GB Ethernet port
- **Licenses:** NFS
- **OS:** Data ONTAP version 8.0.2 (operating in 7-Mode)

Note: We were able to avoid the cost of a cluster site license by using a single FAS2240 controller with the six disks incorporated in it.

MOUNT OPTIONS TO BE USED ON NAMENODE AND SECONDARY NAMENODE SERVERS

Below is a sample `/etc/fstab` entry for the FAS2240 volume, where `stlfas2240-11` is the name of the FAS system:

```
stlfas2240-11:/vol/vol1 /mnt/fsimage_bkp  nfs     rw,rsize=65536,wsize=65536,noatime,soft
```

The `rw` options enable read/write I/O with the mounted volume. `Rsize` and `wsize` were set to 64KB for performance. The `soft` option was used according to Hadoop best practices to make sure that failure of the NFS server (FAS system) would not result in any type of hang-up on the NameNode.

LINUX KERNEL AND DEVICE TUNING

For our tests we used the `/etc/sysctl.conf` settings described in Table 3.

Table 3) Linux /etc/sysctl.conf settings used in tests

Parameter	Actual Setting / Default Value	Description
net.ipv4.ip_forward	0 / 0	Controls IP packet forwarding.
net.ipv4.conf.default.rp_filter	1 / 0	Controls source route verification.
net.ipv4.conf.default.accept_ source_route	0 / 1	Does not accept source routing.
kernel.sysrq	0 / 1	Controls the system request debugging functionality of the kernel.
kernel.core_uses_pid	1 / 0	Controls whether core dumps append the PID to the core file name. Useful for debugging multithreaded applications.
kernel.msgmnb	65536 / 16384	Controls the maximum size of a message, in bytes.
kernel.msgmax	65536 / 8192	Controls the default maximum size of a message queue.
kernel.shmmax	68719476736 / 33554432	Controls the maximum shared segment size, in bytes.
kernel.shmall	4294967296 / 2097512	Controls the maximum number of shared memory segments, in pages.
net.core.rmem_default	262144 / 129024	Sets the default OS receive buffer size.
net.core.rmem_max	16777216 / 131071	Sets the max OS receive buffer size.
net.core.wmem_default	262144 / 129024	Sets the default OS send buffer size.
net.core.wmem_max	16777216 / 131071	Sets the max OS send buffer size.
net.core.somaxconn	1000 / 128	Maximum number of sockets the kernel can serve at one time. Set on NameNode, secondary NameNode, and JobTracker.
fs.file-max	6815744 / 4847448	Sets the total number of file descriptors.
net.ipv4.tcp_timestamps	0 / 1	Turns off the TCP time stamps.
net.ipv4.tcp_sack	1 / 1	Turns on select ACK for TCP.
net.ipv4.tcp_window_scaling	1 / 1	Turns on the TCP window scaling.
kernel.shmmni	4096 / 4096	Sets the maximum number of shared memory segments.
kernel.sem	250 32000 100 128 / 250 32000 32 128	Sets the maximum number and size of semaphore sets that can be allocated.
fs.aio-max-nr	1048576 / 65536	Sets the maximum number of concurrent I/O requests.

Parameter	Actual Setting/Default Value	Description
net.ipv4.tcp_rmem	4096 262144 **16777216 /** 4096 87380 4194304	Sets min, default, and max receive window size.
net.ipv4.tcp_wmem	4096 262144 16777216 / 4096 87380 4194304	Sets min, default, and max transmit window size.
net.ipv4.tcp_syncookies	0 / 0	Turns off the TCP syncookies.
sunrpc.tcp_slot_table_entries	128 / 16	Sets the maximum number of in-flight RPC requests between a client and a server. This value is set on the NameNode and secondary NameNode to improve NFS performance.
vm.dirty_background_ratio	1 / 10	Maximum percentage of active system memory that can be used for dirty pages before dirty pages are flushed to storage. Lowering this parameter results in more frequent page cache flushes to storage, providing a more constant I/O write rate to storage. This gives better storage performance for writes.

Additional tuning on the Linux servers is included in the settings described in Table 4. The sample commands are for block device `/dev/sdb`. These settings must be implemented for all block devices used for HDFS data storage. These changes are not persistent across reboots, so they must be reset after each reboot. A good way to accomplish this is to include the commands in the Linux

`/etc/rc.local` file.

Table 4) Additional tuning for block devices

Description of Setting	Suggested Value	Command Line Example
Block device kernel request queue length	512	echo 512 > /sys/block/sdb/queue/nr_requests
Block device queue depth	254	echo 254 > /sys/block/sdb/device/queue_depth
Block device readahead	1024	/sbin/blockdev --setra 1024 /dev/sdb

We also found that the deadline I/O scheduler performed better than the default CFQ scheduler. This can be set in the `/boot/grub/grub.conf` file on the Linux server. The following sample `grub.conf` file demonstrates how this setting can be implemented:

```
default=0
timeout=5
splashimage=(hd0,0)/grub/splash.xpm.gz hiddenmenu
title Red Hat Enterprise Linux Server (2.6.18-238.el5) root (hd0,0)
    kernel /vmlinuz-2.6.18-238.el5 ro root=/dev/VolGroup00/LogVol00 rhgb quiet
    elevator=deadline initrd /initrd-2.6.18-238.el5.img
```

LINUX SAS CONTROLLER QUEUE DEPTH SETTINGS WITH PROCEDURE

For optimal performance, we increased the queue depth settings on the host-side SAS controllers. This was done using the `lsiutil` utility as described below.

1. Log in to the attached server as the root user and execute the following command:

```
./lsiutil.x86_64
```

You are prompted to choose a device. Choices resemble the following:

```
     Port Name         Chip Vendor/Type/Rev    MPT Rev  Firmware Rev  IOC
  1. /proc/mpt/ioc0    LSI Logic SAS1068E B3     105      011e0000     0
  2. /proc/mpt/ioc0    LSI Logic SAS2008 03      200      09000000     0
```

2. Choose the LSI Logic SAS2008 entry, which in this case is device #2.

```
The following menu will appear.
1.  Identify firmware, BIOS, and/or FCode
 2.  Download firmware (update the FLASH)
 4.  Download/erase BIOS and/or FCode (update the FLASH)
 8.  Scan for devices
10.  Change IOC settings (interrupt coalescing)
13.  Change SAS IO Unit settings
16.  Display attached devices
20.  Diagnostics
21.  RAID actions
23.  Reset target
42.  Display operating system names for devices
43.  Diagnostic Buffer actions
45.  Concatenate SAS firmware and NVDATA files
59.  Dump PCI config space
60.  Show non-default settings
61.  Restore default settings
66.  Show SAS discovery errors
69.  Show board manufacturing information
97.  Reset SAS link, HARD RESET
98.  Reset SAS link
99.  Reset port
 e Enable expert mode in
 menus p Enable paged mode
 w   Enable logging
```

3. On the Main menu, select an option: [1-99 or e/p/w or 0 to quit]
4. Choose item 13, Change SAS IO Unit Settings.
5. When prompted to enter a value for SATA Maximum Queue Depth, enter 255 and press the Return key:

```
SATA Maximum Queue Depth:  [0 to 255, default is 32] 255
```

6. When prompted to enter a value for SAS Max Queue Depth, Narrow, enter 256 and press the Return key:

```
SAS Max Queue Depth, Narrow:  [0 to 65535, default is 32] 256
```

7. When prompted to enter a value for SAS Max Queue Depth, Wide, enter 2048 and press Return:

```
SAS Max Queue Depth, Wide:  [0 to 65535, default is 128] 2048
```

8. Accept the defaults for Device Missing Report Delay and Device Missing I/O Delay by pressing Enter key at each prompt.
9. At the Select a Phy prompt, press Enter to quit.
10. At the Main menu, select an option prompt, enter 99 to reset the port.

    This is required to implement the changes that you just made, as shown in the following example:

```
Device Missing Report Delay:  [0 to 2047, default is 0]
Device Missing I/O Delay:  [0 to 255, default is 0]

PhyNum  Link       MinRate  MaxRate Initiator  Target    Port
    0   Enabled      1.5      6.0   Enabled    Disabled  Auto
    1   Enabled      1.5      6.0   Enabled    Disabled  Auto
    2   Enabled      1.5      6.0   Enabled    Disabled  Auto
    3   Enabled      1.5      6.0   Enabled    Disabled  Auto
    4   Enabled      1.5      6.0   Enabled    Disabled  Auto
    5   Enabled      1.5      6.0   Enabled    Disabled  Auto
    6   Enabled      1.5      6.0   Enabled    Disabled  Auto
    7   Enabled      1.5      6.0   Enabled    Disabled  Auto

Select a Phy:  [0-7, 8=AllPhys, RETURN to quit]
Main menu, select an option:   [1-99 or e/p/w or 0 to quit] 99
```

A message is displayed indicating that the port is being reset.

11. When prompted again for a menu option, enter 0 to quit. You are then prompted to select a device. As before, enter 0 to quit, as shown in the following example:

```
Resetting port...

Main menu, select an option:  [1-99 or e/p/w or 0 to quit] 0

     Port Name         Chip Vendor/Type/Rev    MPT Rev  Firmware Rev  IOC
  1. /proc/mpt/ioc0    LSI Logic SAS1068E B3     105      011e0000     0
  2. /proc/mpt/ioc0    LSI Logic SAS2008 03      200      05000d00     0

Select a device:  [1-2 or 0 to quit] 0
#
```

HADOOP PARAMETER SETTINGS

The `hadoop-env.sh` configuration file typically contains commands to set environment variables for the Hadoop environment.

The only required change to the default settings in this file is to set the `JAVA_HOME` environmental variable for Hadoop, as shown in the following example:

```
export JAVA_HOME=/usr/java/jdk1.6.0_30
```

To improve cluster efficiency, we changed `HADOOP_HEAPSIZE` from the default value of 1000MB to 2000MB with the following `hadoop-env.sh` entry:

```
export HADOOP_HEAPSIZE=2000
```

The `core-site.xml` configuration file contains core configuration information used by all components in the Hadoop cluster. Table 5 below lists the `core-site.xml` parameter settings used in our test environment.

Table 5) Hadoop core-site.xml parameter settings used in test

Option Name	Actual Setting / Default Value	Purpose
fs.default.name	hdfs:// 10.61.189.64:8020/ [Default Value: file:///]]	Name of the default file system specified as a URL (IP address or hostname of the NameNode along with the port to be used).
webinterface.private.actions	true / false	Enables or disables certain management functions in the Hadoop Web user interface, including the ability to kill jobs and modify job priorities.
fs.inmemory.size.mb	200 / 100	Memory in MB to be used for merging map outputs during the reduce phase.
io.file.buffer.size	262144 / 4096	Size in bytes of the read/write buffer.
topology.script.file.name	/etc/hadoop/conf/topology_ script [Default value is null]	Script used to resolve the slave nodes' name or IP address to a rack ID. Used to invoke Hadoop rack awareness. The default value of null results in all slaves being given a rack ID of `/default-rack`.
topology.script.number.args	1 / 100	Sets the maximum acceptable number of arguments to be sent to the topology script at one time.
hadoop.tmp.dir	/home/hdfs/tmp [Default value: /tmp/hadoop-${user.name}]	Hadoop temporary directory storage.

Parameters used to configure HDFS are contained in the `hdfs-site.xml` file. Table 6 below shows the settings used in testing the NetApp Open Solution for Hadoop.

Table 6) Hdfs-site.xml parameter settings used in test

Option Name	Actual Setting / Default Value	Purpose
dfs.name.dir	/local/hdfs/namedir,/mnt/fsimage_bkp [Default value: ${hadoop.tmp.dir}/dfs/name]	Path on the local file system where the NameNode stores the namespace and transaction logs persistently. If this is a comma-delimited list of directories (as used in this configuration), then the name table is replicated in all of the directories for redundancy. **Note:** Directory /mnt/fsimage_bkp is a location on NFS-mounted NetApp FAS storage where NameNode metadata is mirrored and protected, a key feature of our Hadoop solution.
dfs.hosts	/etc/hadoop-0.20/conf/dfs_hosts [Default value is null]	Specifies a list of machines authorized to join the Hadoop cluster as a DataNode.
dfs.data.dir	/disk1/data,/disk2/data [Default value: ${hadoop.tmp.dir}/dfs/data]	Directory path locations on the DataNode local file systems where HDFS data blocks are stored.
fs.checkpoint.dir	/home/hdfs/namesecondary1 [Default value: ${hadoop.tmp.dir}/dfs/namesecondary]	Path to the location where checkpoint images are stored (used by the secondary NameNode).
dfs.replication	2 / 3	HDFS block replication count. Hadoop default is 3. We use 2 in the NetApp Hadoop solution.
dfs.block.size	134217728 (128MB) / 67108864	HDFS data storage block size in bytes.
dfs.namenode.handler.count	128 / 10	Number of server threads for the NameNode.
dfs.datanode.handler.count	64 / 3	Number of server threads for the DataNode
dfs.max-repl-streams	8 / 2	Maximum number of replications a DataNode is allowed to handle at one time.
dfs.datanode.max.xcievers	4096 / 256	Maximum number of files a DataNode can serve at one time.

Table 7 below lists the `mapred-site.xml` settings used in our test environment. The parameters defined in the `mapred-site.xml` file are used to control MapReduce processing and to configure the JobTracker and TaskTrackers.

Table 7) Mapred-site.xml parameters used in testing the NetApp Open Solution for Hadoop

Option Name	Actual Setting / Default Value	Purpose
mapred.job.tracker	10.61.189.66:9001 [Default value: local]	JobTracker address as a URI (JobTracker IP address or hostname with port number).
mapred.local.dir	/disk1/mapred/local,/disk2/mapred/local [Default value: ${hadoop.tmp.dir}/mapred/local	Comma-separated list of the local file system where temporary MapReduce data is written.
mapred.hosts	/etc/hadoop-0.20/conf/mapred.hosts [Default value is null]	Specifies the file that contains the list of nodes allowed to join the Hadoop cluster as TaskTrackers.
mapred.system.dir	/mapred/system [Default value: ${hadoop.tmp.dir}/mapred/system]	Path in HDFS where the MapReduce framework stores control files.
mapred.reduce.tasks.speculative.execution	false / true	Enables the JobTracker to detect slow-running reduce tasks, assign them to run in parallel on other nodes, use the first available results, and then kill the slower-running reduce tasks.
mapred.map.tasks.speculative.execution	false / true	Enables the JobTracker to detect slow-running map tasks, assign them to run in parallel on other nodes, use the first available results, and then kill the slower running map tasks.
mapred.tasktracker.reduce.tasks.maximum	5 / 2	Maximum number of reduce tasks that can be run simultaneously on a single TaskTracker node.
mapred.tasktracker.map.tasks.maximum	7 / 2	Maximum number of map tasks that can be run simultaneously on a single TaskTracker node.
mapred.child.java.opts	-Xmx1024m / -Xmx200m	Java options passed to the TaskTracker child processes (in this case, 2GB defined for heap memory used by each individual JVM).
io.sort.mb	340 / 100	Total amount of buffer memory allocated to each merge stream while sorting files on the mapper, in MB.
mapred.jobtracker.taskScheduler	org.apache.hadoop.mapred.FairScheduler [Default value: org.apache.hadoop.mapred.JobQueueTaskScheduler]	JobTracker task scheduler to use (in this case, use the FairScheduler).
io.sort.factor	100 / 10	Number of streams to merge at once while sorting files.
mapred.output.compress	false / false	Enables/disables output file compression.

Option Name	Actual Setting / Default Value	Purpose
mapred.compress.map.output	false / false	Enables/disables map output compression.
mapred.output.compression.type	block / record	Sets output compression type.
mapred.reduce.slowstart.completed.maps	0.05 / 0.05	Fraction of the number of map tasks that should be complete before reducers are scheduled for the MapReduce job.
mapred.reduce.tasks	40 for 8 DataNodes 80 for 16 DataNodes 120 for 24 DataNodes [Default value: 1]	Total number of reduce tasks to be used by the entire cluster per job.
mapred.map.tasks	56 for 8 DataNodes 112 for 16 DataNodes 168 for 24 DataNodes [Default value: 2]	Total number of map tasks to be used by the entire cluster per job.
mapred.reduce.parallel.copies	64 / 5	Number of parallel threads used by reduces tasks to fetch outputs from map tasks.
mapred.compress.map.output	false / false	Enables/disables map output compression.
mapred.inmem.merge.threshold	0 / 1000	Number of map outputs in the reduce TaskTracker memory at which map data is merged and spilled to disk.
mapred.job.reduce.input.buffer.percent	1 / 0	Percent usage of the map outputs buffer at which the map output data is merged and spilled to disk.
mapred.job.tracker.handler.count	128 / 10	Number of JobTracker server threads for handling RPCs from the task trackers.
tasktracker.http.threads	60 / 40	Number of TaskTracker worker threads for fetching intermediate map outputs for reducers.
mapred.job.reuse.jvm.num.tasks	- 1/1	Maximum number of tasks that can be run in a single JVM for a job. A value of -1 sets the number to unlimited.
mapred.jobtracker.restart.recover	true / false	Enables job recovery after restart.

4.5 Test Software Inventory

Table 8 below summarizes the software used by our Hadoop solution and additional tools used for monitoring.

Table 8) Test software inventory

Software or Application	Version
IBM InfoSphere BigInsights Software	Version 2.0
Hadoop test tools TeraGen, & TeraSort	Part of IBM InfoSphere BigInsights
Java®	1.6.0_30
SANtricity client for Windows®	10.80.G4.51

4.6 Solution Test Results

Table 9 below lists the final test results for each test case described in the appendix.

Table 9) Solution test results

Test Case ID	Test Case Name	Duration	Pass/Fail (P/F)
1	Tuning run	Multiple runs over 3 days	P
2	Full functionality with fault injection	3 iterations	P
3	NameNode metadata recovery	1 iteration	P

5 CONCLUSION

The NetApp reference architecture for IBM BigInsights reference architecture is optimized for node-storage balance, reliability, performance, and storage capacity and density. Organizations that use Hadoop often use traditional server-based storage with inefficient, hard-to-scale, internal direct-access storage (DAS). The NetApp IBM BigInsights reference design employs the managed DAS model, with higher scalability and lower TCO. The lower cost comes from choosing the number of clusters and storage you want and not spending money for features you don't need. It also comes from nonproprietary lower-cost options and by decoupling compute nodes from storage, preventing unnecessary purchases of compute nodes for storage-intensive workloads.

IBM's depth and breadth of expertise spans a wide range of enterprise software, hardware, and services, enabling firms who partner with IBM to approach their big data projects with confidence and clarity. To learn more about what IBM can do for you, visit http://www.ibm.com/bigdata today. To dig into the details of BigInsights and connect with its community, visit http://www.ibm.com/developerworks/wiki/biginsights.

To learn about NetApp's big data solutions, visit http://www.netapp.com/big data and to find out more about NetApp's Hadoop solutions, please visit http://www.netapp.com/hadoop.

Appendixes: Test Case Details

Table 10) Test case 1-A: initial tuning

Test Case ID	Test Type	Execution Type	Duration
1-A	Initial	Manual	Multiple runs, one day total

Description	Prerequisites
Runs short (5—10 minutes) TeraGen and TeraSort jobs to aid in initial component validation and configuration tuning.	HDFS components have been started.
Setup Procedures	**Execution Procedures**
1. Remove any previous data artifacts from the HDFS file system. Do this before each run.	1. Use included Apache TeraGen and TeraSort. Start from the JobTracker node.
Expected Results	**Notes**
TeraGen and TeraSort jobs complete without error.	1. Use integrated Web/UI tools to monitor Hadoop tasks and file system.

Table 11) Test case 1-B: full functionality (baseline)

Test Case ID	Test Type	Execution Type	Duration
1-B	Full function	Automated	Multiple runs, one day total

Description	Prerequisites
Runs a TeraGen job with duration greater than 10 minutes to generate a substantial dataset. Runs a TeraSort job on the previously created dataset.	•HDFS components have been started. •Test case 1-A has been completed successfully.
Setup Procedures	**Execution Procedures**
1. Remove any previous HDFS artifacts before each run.	1. Use included TeraGen and TeraSort. Start from the JobTracker node. 2. Use the same TeraGen and TeraSort parameters during all iterations.
Expected Results	**Notes**
• Proper output from TeraSort reduce stage. • No tasks on individual task nodes (DataNodes) fail. • File system (HDFS) has maintained integrity and is not corrupted. • All test environment components are still running.	1. Use integrated Web/UI tools to monitor Hadoop tasks and file system.

Table 12) Test case 2: full functionality with fault injection

Test Case ID	Test Type	Execution Type	Duration
2	Full functional with fault injection	Manual	Multiple runs, one day total

Description	Prerequisites
Runs a TeraGen job with duration greater than 10 minutes to generate a substantial dataset. Runs a TeraSort job on the previously created dataset.	• HDFS components started. • Test case 1-B has been completed successfully.
Setup Procedures	**Execution Procedures**
1. Remove any previous HDFS artifacts prior to each run. 2. Make sure that all components are restored to a non-faulted condition.	1. Use included Apache TeraGen and TeraSort. Start from the JobTracker node. 2. Use the same TeraGen and TeraSort parameters during all iterations. 3. After Terasort has been running for approximately halfway minutes, fault a disk in each of four controllers on the three E-Series arrays.
Expected Results	**Notes**
1. Proper output from the TeraSort Reduce stage. 2. No tasks on individual task nodes (DataNodes) fail. 3. File system (HDFS) has maintained integrity and is not corrupted.	1. Use integrated Web/UI tools to monitor Hadoop tasks and file system. 2. Use Ganglia to monitor Linux server in general.

Table 13) Test case 3: NameNode metadata recovery

Test Case ID	Test Type	Execution Type	Duration
3	Recovery	Manual	2 days

Description	Prerequisites
Tests failure of Hadoop NameNode, corrupt the metadata and the Procedures to recover.	Previous test cases (1-A, 1-B, and 2) have been Completed, and all are all successful.

Setup Procedures	Execution Procedures
1. Repeat test 1-B. 2. Note the total reduce input record could from the TeraSort job. 3. Run **`hadoop fsck /`** command and note the following: * Healthy status of NDFS * Number of files * Total size of the file system * Corrupt blocks	1 Repeat the TeraSort job. 2 While it is running, halt the NameNode to simulate loss of HDFS metadata. 3 Delete the metadata-directory 4 Shut down all Hadoop daemons on all servers in the cluster. 5 Bring up the namenode again and make sure the HDFS services fail due to unavailability of metadata
	1. Rebuild the master namenode. Modify the `hdfs.site.xml` file to mount the NFS based backup image of fsimage and secondary location. 2. Start the NameNode daemon on the NameNode server. 3. Restart the whole cluster using start-all.sh script. 4. From any node in the cluster, use the `hadoop dfs –ls` command to verify that the data file created by TeraGen exists. 5. Check the total amount of HDFS storage used. 6. Run `hadoop fsck /` command to compareto results recorded before the NameNode was halted. 7. Run TeraSort against the file originally generated by TeraGen and monitor for error-free completion.

Expected Results	Notes
Output from the TeraSort should match the original output from test case 1-B.	1. Use integrated Web/UI tools to monitor Hadoop tasks and file system. 2. Use Ganglia to monitor Linux server in general.