Protecting Hadoop with VMware vSphere® 5 Fault Tolerance

Performance Study

TECHNICAL WHITE PAPER

vmware®

Table of Contents

Executive Summary

VMware vSphere® Fault Tolerance (FT) can be used to protect virtual machines that run vulnerable components of a Hadoop cluster with only a small impact on application performance. A cluster of 24 hosts was used to run three applications characterized by different storage patterns. Various Hadoop configurations were employed to artificially create greater load on the NameNode and JobTracker daemons. With conservative extrapolation, these tests show that uniprocessor virtual machines with FT enabled are sufficient to run the master daemons for clusters of more than 200 hosts.

Introduction

Apache Hadoop provides a platform for building distributed systems for massive data storage and analysis [1]. It was designed to run on large clusters of x86-based servers equipped with standard disk drives without RAID protection. Hadoop uses data replication across hosts and racks of hosts to protect against individual disk, host, and even rack failures. However, a Hadoop workflow is serialized through the NameNode and JobTracker daemons. The NameNode manages the Hadoop Distributed Filesystem (HDFS) namespace, maintains the metadata for all files, keeps track of the locations of the blocks that make up each file, and coordinates block replication. The JobTracker schedules jobs and the map and reduce tasks that make up each job. The JobTracker can also detect a task failure and reschedule it on another worker node. The bulk of the infrastructure consists of worker nodes, each of which is a physical or virtual machine that runs a DataNode and a TaskTracker. The DataNode handles storage and communicates with the NameNode, and the TaskTracker runs the map and reduce tasks under the direction of the JobTracker.

The NameNode and JobTracker are both Single Points of Failure (SPoFs). In Hadoop 1.0 (recently renamed from 0.20) there is a Secondary NameNode that backs up the NameNode data and can be used to recover most of the NameNode state (with manual intervention) in case of failure. If the JobTracker fails, typically the current jobs need to be restarted after a new JobTracker is up and running. In Hadoop 2.0 (formerly 0.23), considerable work is currently going into making the NameNode more robust, including implementing active failover. The tests presented here used a distribution based on Hadoop 1.0.

VMware provides two features to enable increased availability of applications running on vSphere. VMware vSphere® High Availability (HA) provides automated restart of virtual machines in the event of hardware or operating system failures. VMware vSphere® Fault Tolerance (FT) provides continuous availability of virtual machines by running a secondary virtual machine on another host that closely shadows the primary virtual machine. The architecture and usage of these features are described in detail in documents available from VMware [2].

As previously mentioned, it is expected that in the future Hadoop itself will protect at least the NameNode with active redundancy. However, there are many other SPoFs in the Hadoop ecosystem (for example, within Zookeeper, Hbase, and so on), with more being created as new applications are added. Thus there is a current and ongoing need for a more generic and external method of protecting against SPoFs in Hadoop clusters. One possibility was demonstrated at the recent Hadoop Summit: HortonWorks showed that the NameNode can be protected with HA [3]. Some modifications were needed to Hadoop itself to handle recovery and to detect failure. This method has the advantage that it can respond to application and guest operating system software failures, not just hardware failures. The disadvantage is that there will be some loss of data and a delay while a new virtual machine and the NameNode within it are started and synced to the previous state.

Another method to protect vulnerable components of a Hadoop cluster is to place these components in separate VMs and apply FT to them. In the event of a failure of one of these virtual machines or the host on which it is running, the secondary virtual machine running on another host will seamlessly take over. Nothing is required of the application and it is not even aware that a transition has taken place. The disadvantage of FT is that it is not able to detect when the application itself fails or the guest hangs. In those cases, the instructions executed by the virtual machine will be faithfully replicated on the secondary virtual machine.

Maximum application availability will be achieved with a combination of FT to protect against physical and virtual hardware failures with no downtime or loss of data, and application of HA to protect against software failures inside of the virtual machine. An extensive review of the architecture and performance of VMware FT is provided in "VMware vSphere 4 Fault Tolerance: Architecture and Performance" [4]. In the context of VMware hypervisors, FT is equated with the generic term "fault tolerance." However, there are several other somewhat different definitions in use. In a Cloudera blog that gives a good overview of several availability concepts [5], the term is given a weaker definition than here.

This paper examines the performance effects of enabling FT for the virtual machines that run the Hadoop NameNode and JobTracker. Also, since FT is limited to uniprocessor virtual machines, there will be a limit to the load that can be placed on these components, and thus a limit to the cluster size that can be supported. The load on both components can be artificially increased by decreasing the size of the Hadoop File System (HDFS) block size. The resulting data can then be extrapolated to estimate the maximum cluster size.

A previous paper on virtualized Hadoop performance tests on a small cluster [6] included discussions of motivations for deploying Hadoop, its architecture, and reasons for virtualizing a Hadoop cluster.

Configuration

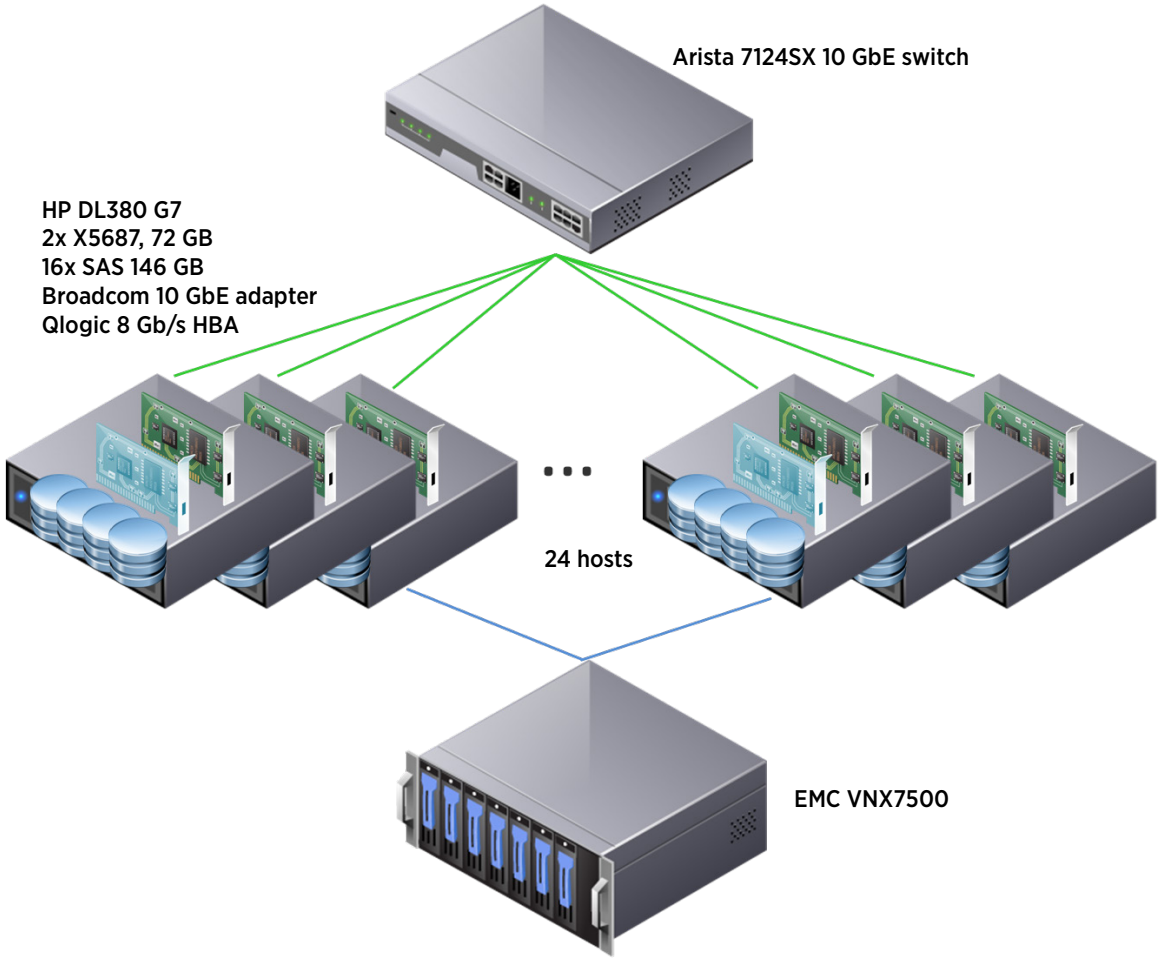

Figure 1. Cluster hardware configuration

Hardware Overview

The hardware configuration is shown in Figure 1. A cluster of 24 host servers was connected to a single Arista 7124SX 10GbE switch. Each host was equipped with two Intel X5687 3.6GHz quad-core processors, 16 internal 146GB 15K RPM SAS disks, a 10GbE Broadcom adapter, and a Qlogic QLE 2560-SP Fibre Channel (FC) host bus adapter (HBA). The internal disks were connected to a PCIe 2.0 x4 storage controller, the HP P410i. This controller has a theoretical throughput of 2000MB/s, but in practice can deliver 1000-1400MB/s. The FC HBA has a single 8Gb/s port. Power states, including TurboMode, were disabled in the BIOS to improve consistency of results. Intel Hyper-Threading (HT) Technology was enabled in the BIOS and used by the operating system in all cases.

Two of the hosts were direct-connected to an EMC VNX7500 SAN over FC. One of these hosts ran virtual machines for the Hadoop NameNode and JobTracker, and the other host ran the secondary copies of these virtual machines when FT was enabled. The SAN was used as shared storage for the root disks of these two virtual machines and not used for Hadoop data. Complete hardware details are given in Appendix: Configuration.

Local Storage

Two of the sixteen internal disks in each host were striped, divided into partitions, and used to store the virtual machine images. The ESXi hypervisor was PXE-booted from a network repository. ESXi has a memory-based file system so does not create any storage I/O itself. During the tests, the I/O to these two "root" disks consisted almost exclusively of just Hadoop logs and job statistics. The other fourteen disks were configured as individual disks (that is, there was no striping or RAID). Twelve were used for Hadoop data, while two were kept as spares. The disks were passed through to the virtual machines using physical raw device mappings (known as pRDM). A single aligned partition was created on each disk and formatted with EXT4.

Shared Storage

Two hosts were directly connected to one of the two storage processors (SPs) on the SAN. Six SSD disks were used to create a RAID-5 RAID group and a single LUN was configured. Both hosts and this LUN were placed in one storage group. This prevented LUN trespassing and allowed the LUN to be visible to both hosts. A datastore was configured on the LUN and the two virtual machines used for the NameNode and JobTracker were stored there.

System Software

RHEL 6.1 x86_64 was used as the guest operating system in the virtual machines, which were run on the GA version of VMware vSphere 5.0 U1 (build 623860). Four virtual machines for Hadoop worker nodes were run on each host, except for the two hosts previously mentioned which ran two worker virtual machines each in addition to the NameNode and JobTracker virtual machines. This gives a total of 92 worker virtual machines. The worker virtual machines were configured with four CPUs and a little over 17GB of memory. The NameNode and JobTracker virtual machines had one CPU and two GB of memory. Except for the hosts that ran these two virtual machines, the CPU resources were exactly-committed and memory was slightly under-committed (5.6% of host memory was held back for the hypervisor and virtual machine memory overhead). The virtual network adapter was vmxnet3. Default networking parameters were used both in ESXi and in the guest. Sun Java 6 is recommended for Hadoop; version 1.6.0_26 was used here. A few Linux kernel tunables were increased.

No hypervisor tuning was done except each of the worker virtual machines was pinned to a NUMA node (which is a processor socket plus the associated memory) for better reproducibility of the results.

These and other operating system and virtual machine details are given in Appendix: Configuration.

Hadoop

The Cloudera CDH3u4 version of Apache Hadoop 1.0 was installed on all virtual machines. Replication was set to two, as is common for small clusters (large clusters should use three or more). For best performance the HDFS block size was increased to 256MB from the 64MB default. This increases the efficiency by creating fewer but longer-running Hadoop tasks. The trade-off is that a larger block size needs more memory and may make balancing the workload across a large cluster more difficult for the Hadoop scheduler. Most of machine memory was used for the Java heap for the task JVMs. The maximum heap sizes were set to 1900MB for both map and reduce tasks. A few other Hadoop parameters were changed from their defaults in order to increase efficiency further and are listed in Appendix: Configuration. Probably the most important Hadoop parameters are the numbers of map and reduce tasks. The optimum depends on the particular application and the hardware configuration, especially the storage capabilities. Setting the number of each task type equal to the number of virtual CPUs in the worker virtual machines worked well overall for the current tests.

As previously noted, all 24 hosts were used to run Hadoop worker nodes. A common recommended practice is to dedicate separate machines for the NameNode and JobTracker. However, this is wasteful of resources for clusters of small to moderate size. Virtualization fixes this by giving the administrator the flexibility to size each virtual machine according to its needs and to run the appropriate number of virtual machines on each host. Here, 22 hosts were fully utilized by running four worker virtual machines each. These virtual machines each contained a Hadoop worker node that ran a DataNode, a TaskTracker, and a set of map and reduce tasks. The other two hosts ran two worker virtual machines each. One of those hosts also ran uniprocessor virtual machines for the NameNode and JobTracker. When FT was enabled, these two virtual machines were shadowed by an identical pair on the second host. The Secondary NameNode was run in one of the worker virtual machines, and the Hadoop Client was run in another. These consume very little resources and had no noticeable effect on the performance of those worker nodes.

The tests presented here were performed with an HDFS replication factor of two. This means each original block is copied to one other worker node. However, when multiple virtual machines per host are used, there is the undesirable possibility from a single-point of failure perspective that the replica node chosen is another virtual machine on the same host. Hadoop manages replica placement based on the network topology of a cluster. Since Hadoop does not discover the topology itself, the user needs to describe it in a hierarchical fashion as part of the input (the default is a flat topology). Hadoop uses this information both to minimize long-distance network transfers and to maximize availability, including tolerating rack failures. Virtualization adds another layer to the network topology that needs to be taken into account when deploying a Hadoop cluster: inter-VM communication on each host. Equating the set of virtual machines running on a particular host to a unique rack ensures that at least the first replica of every block is sent to a virtual machine on another host. Development is currently underway to add virtualization-aware network topology to Hadoop (JIRA: HADOOP-8468).

Hadoop Benchmarks

Several example applications are included with the Cloudera distribution. Three of these are often run as standard benchmarks: TeraGen, TeraSort, and TeraValidate. Table 1 summarizes the number of simultaneous map and reduce tasks across the cluster for these applications. Also shown in the table are the relative amounts of read and write storage operations.

BENCHMARK	MAP	REDUCE	%READ	%WRITE
TeraGen	368	0	0	100
TeraSort	368	368	40	60
TeraValidate	368	1	100	0

Table 1. Total number of simultaneous tasks and storage characteristics of each benchmark

This set of applications creates, sorts, and validates a large number of 100-byte records. It does considerable computation, networking, and storage I/O and is often considered to be representative of real Hadoop workloads. Results are reported for ten billion records, which is often referred to as the "1 TB" case.

TeraGen creates the data and is similar to TestDFSIO-write except that significant computation is involved in creating the random data. The map tasks write directly to HDFS so there is no reduce phase.

TeraSort does the actual sorting, reading the generated data from HDFS and writing the sorted data back to HDFS in a number of partitioned files. The application itself overrides the specified replication factor (sets it to one) for the output so only one copy is written. The philosophy is that if a data disk is lost, the user can always rerun the application, but input data needs replication since it may not be as easily recovered.

TeraValidate reads all the sorted data to verify it is correct (that is, it is in order). The map tasks perform this verification for each partitioned file independently, and then the single reduce task checks that the last record of each partition comes before the first record of the next one.

Benchmark Results

Shown in Table 2 are the measured performance data for the three benchmarks with various HDFS block sizes and for FT enabled or disabled. In the next three subsections the elapsed times, CPU utilizations of the NameNode and JobTracker virtual machines, and the network traffic associated with enabling FT for these virtual machines are discussed.

RunID	FT	HDFS block size, MB	Elapsed Time, sec			NameNode %CPU			JobTracker %CPU			FT logging, Mb/s		
			gen	sort	valid	gen	sort	valid	gen	sort	valid	gen	sort	valid
402b	on	256	104	466	63	10	7	4	6	9	2	13	12	8
401	off	256	103	458	64	8	11	3	6	8	3	-	-	-
403	on	64	101	585	68	21	16	4	12	22	3	16	20	7
419	off	64	107	574	57	17	10	3	12	25	2	-	-	-
404	on	16	109	860	70	41	26	7	12	45	5	32	31	10
418c	off	16	106	848	62	61	16	6	3	45	20	-	-	-
405	on	8	142	1240	62	61	36	8	10	72	26	44	42	12
406	off	8	136	1195	64	43	27	6	10	81	15	-	-	-

Table 2. Performance data. %CPU is peak minus idle utilization based on %run. FT logging is maximum transmit network bandwidth over the VMkernel port.

Elapsed Time

Decreasing the block size proportionally increases the number of blocks the NameNode needs to manage and the number of tasks the JobTracker must keep track of. Varying the number of Hadoop nodes while keeping the application load the same (by using half as many virtual machines which are each twice as large) does not have much effect on the amount of work the NameNode and JobTracker need to do, so the data for these cases are not reported here. Thus a smaller HDFS block size can be used to simulate the effect of a larger cluster size on the NameNode and JobTracker.

To put the performance effect of FT into context, a few points about the data in Table 2 should be noted. First, HDFS block size has a large effect on TeraSort performance: decreasing this parameter from 256MB to 8MB increases elapsed time by almost three times. Even using the default size of 64MB instead of 256MB causes a 25% overhead. At a 4MB block size TeraGen completes, but TeraSort fails halfway through the test with the JobTracker virtual machine pegged at 100% CPU utilization. The block size has a much smaller effect on TeraGen, significantly

affecting its performance only at 8MB or smaller, although the CPU utilization of the NameNode virtual machine becomes large. There is no significant effect on TeraValidate. The latter two tests are less sensitive to block size since the total number of map tasks does not depend on it. For the purposes of this paper, the most important point about the application slowdown with block size is that it effectively reduces the load on the NameNode and JobTracker by stretching their operations out in time by the amount of the slowdown. For instance, TeraSort needs to process 32 times as many blocks for the 8MB block size case compared to the 256MB case. But since it takes 2.6 times as long, the average load on the NameNode and JobTracker increases only by about 12 times. Second, the CPU utilization is difficult to measure and noisy, especially for TeraGen and TeraValidate, and also whenever the utilization is low. Finally, the elapsed time of each test is subject to "long pole" effects, where most of the cluster is idle near the end of the test while a few tasks are still finishing. In light of this, overall trends are more important than individual data points.

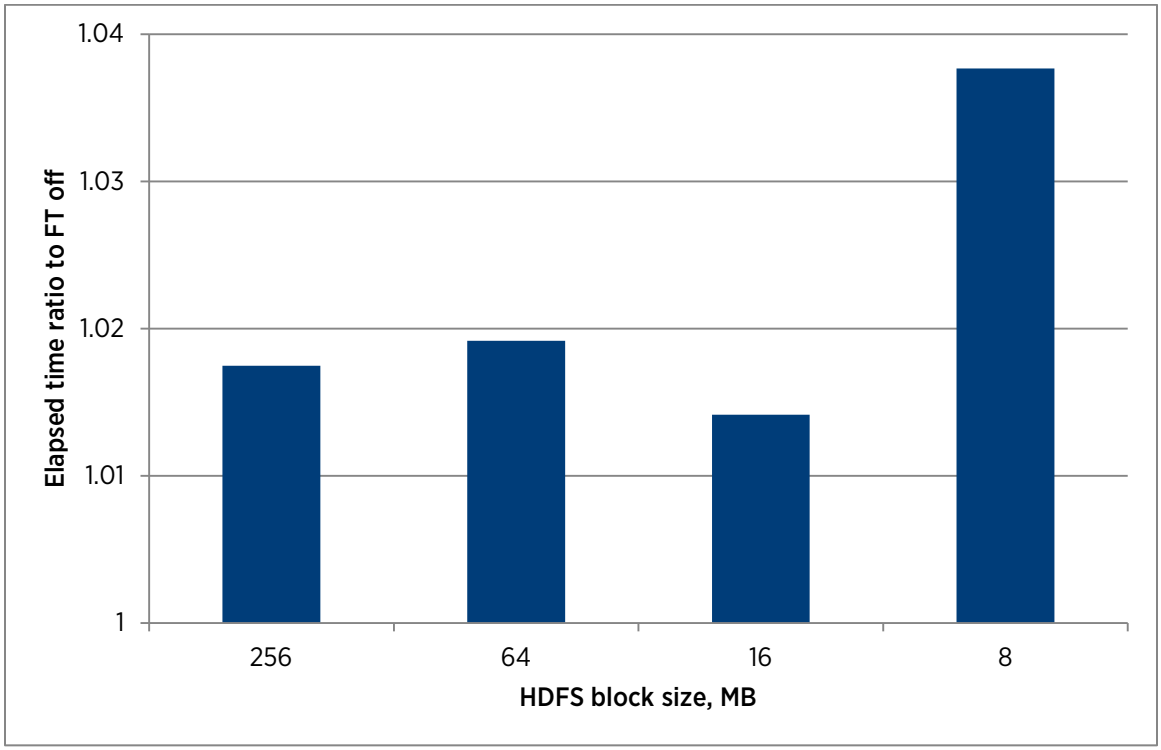

Figure 2. Elapsed time of TeraSort with FT enabled. Ratio is to the corresponding FT disabled cases. Lower is better for FT enabled.

The elapsed time for TeraSort is shown as a ratio of the FT enabled to disabled cases for the various block sizes in Figure 2. There is a performance penalty of up to 4% when FT is enabled for this application, with the largest penalty occurring for the smallest block size where the CPU utilization of the JobTracker becomes large. It is not clear that the latter causes the former. The elapsed times of the other two applications exhibit more measurement noise, so they are not shown in the figure. However, on average they show a similar performance effect of enabling FT.

CPU Utilization

CPU utilization of the NameNode virtual machine for TeraGen is relatively high since new blocks are being created at a rapid rate, while its minimal task management needs results in low JobTracker CPU utilization. TeraSort has the opposite requirements: block management is moderate while a lot of tasks are created. TeraValidate has minimal needs for both block and task management and correspondingly low CPU utilization. In all cases it is evident that the noisiness of CPU utilization means that measuring peak utilization will be difficult. The figure also shows that at idle there is a baseline utilization of about 1% for the NameNode and 2% for the JobTracker. These values were subtracted from the actual measured peak virtual machine utilizations to get the values in Table 2.

CPU utilization of the NameNode and JobTracker virtual machines was measured using esxtop with 10 second collection intervals and the %run (scheduled time) metric. This is shown in Figure 3 for the duration of all three applications for the 8MB block size case with FT enabled (RunID 405).

For TeraSort the NameNode and JobTracker CPU utilizations are normalized by the "block throughput" and shown in Figure 4. The block throughput is defined here as the total number of processed blocks divided by the elapsed time of the test, and as previously mentioned, it increases by about 12 times from the largest to smallest block size. If the normalized CPU utilization is constant, then the CPU utilization of the virtual machines shown would be exactly proportional to the block throughput, and the latter could be considered a good predictor of the former. This is not quite true; both become more efficient as the number of blocks increase. However for smaller block sizes (larger CPU utilizations) the normalized CPU utilization is reasonably constant.

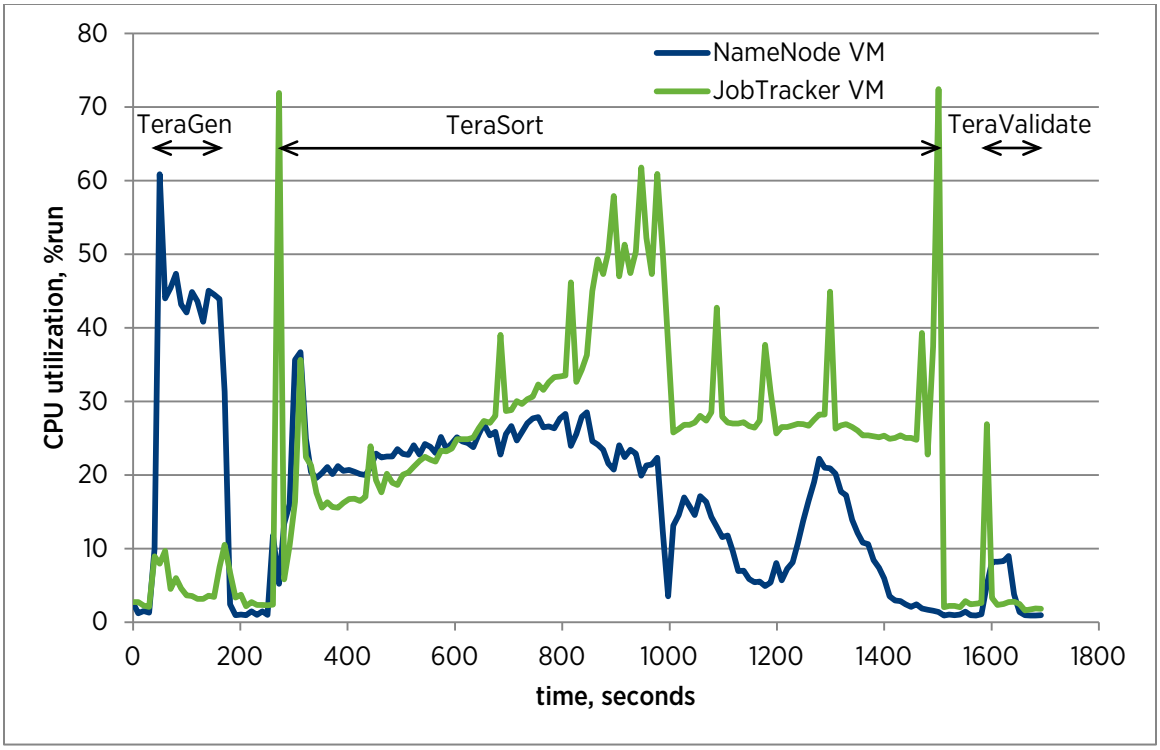

Figure 3. CPU utilization for the NameNode and JobTracker VMs for 8MB block size and FT enabled (RunID 405)

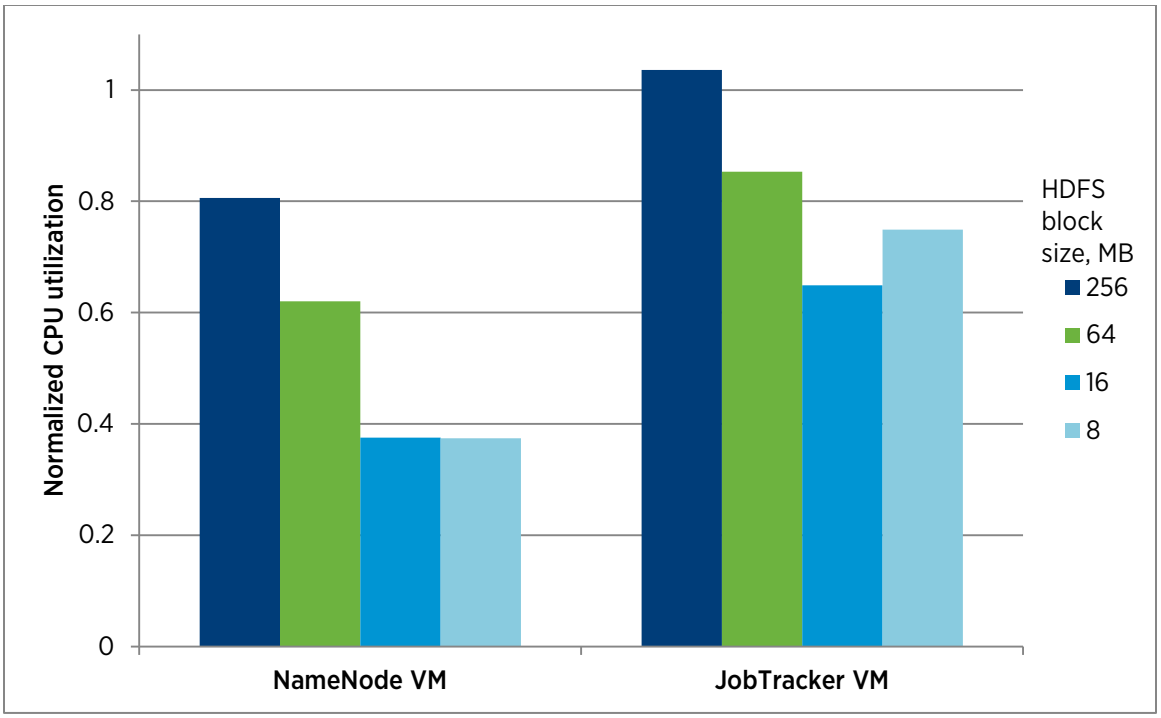

Figure 4. TeraSort percent CPU utilization normalized by block throughput with FT enabled for various HDFS block sizes

Network Traffic

A potential performance limitation of FT is the "logging" traffic required to keep the primary and secondary virtual machines synchronized. Most of the logging bandwidth is associated with incoming networking and storage I/O. Outgoing I/O does not need to be replicated on the secondary virtual machines. Shown in Figure 5 are the receive (Rx) and transmit (Tx) bandwidths for the VMkernel port handling the FT logging traffic, as well as the network traffic of the NameNode and JobTracker virtual machines, for the same case as Figure 3 (RunID 405). The maximum FT logging bandwidth observed was 44Mb/s. A standard best practice is to dedicate a separate network to FT logging to ensure performance; however, on a 10GbE network this is clearly overly conservative for this workload. For comparison, the worker nodes drive up to 2500Mb/s per host in each direction. The maximum FT logging bandwidths for all the block sizes and benchmarks are given in Table 2. The bandwidth increases with decreasing block size, but does not scale linearly with block throughput like the CPU utilization does. Thus the FT logging bandwidth is unlikely to ever be an issue on modern networks.

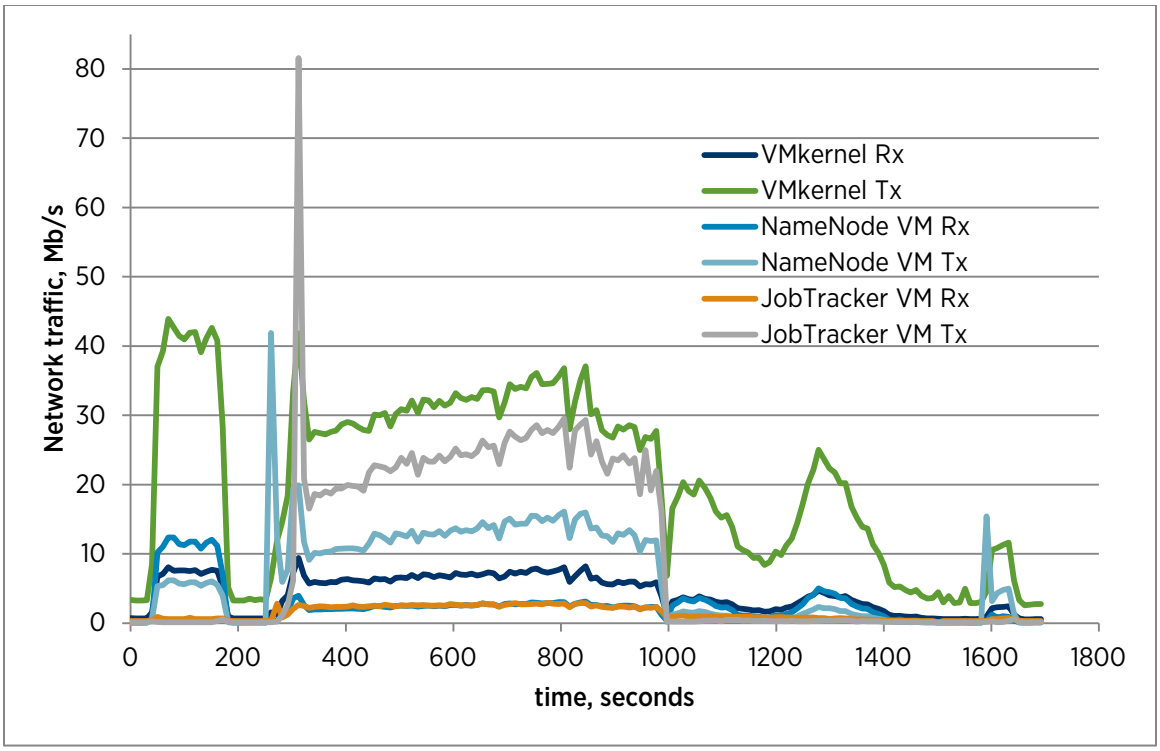

Figure 5. Network bandwidth for FT logging and the NameNode and JobTracker VMs, for 8MB block size and FT enabled (RunID 405)

Extrapolation to Larger Clusters

Assuming the block throughput is the major determinant of the CPU utilization of the NameNode and JobTracker virtual machines, an estimate of the maximum cluster size can be derived where those two virtual machines are protected by FT and the HDFS block size is set to 256MB.

First, consider the NameNode virtual machine. At an 8MB block size, the block throughput for TeraGen is 23 times that of the 256MB case while the CPU utilization is still conservatively well below 100%. A similar CPU utilization and block throughput should be obtained with the larger block size on a cluster that is 23 times larger, or about 550 hosts. TeraSort has a block throughput only 12 times larger for the 8MB case, but at 36% the CPU utilization could safely be doubled, yielding about the same estimate for the maximum number of hosts. An estimate using the data for TeraValidate would be much larger.

Considering the JobTracker virtual machine, neither TeraGen nor TeraValidate create a large load and can be ignored. TeraSort does create a large load and, for an 8MB block size, the CPU utilization of the JobTracker virtual machine is near a safe limit. Therefore, with a 256MB block size, the cluster could be scaled up by 12 times. Since two of the 24 hosts are only half-used for worker virtual machines, the current cluster is effectively 23 hosts. The extrapolation for the JobTracker virtual machine case then yields a maximum of 276 hosts, each running four worker virtual machines.

Best Practices

The largest constraint on using vSphere FT is that it requires uniprocessor virtual machines. To maximize the Hadoop cluster size that can be supported when running the critical components in such virtual machines, a few practices should be considered:

- Use the highest-performance CPU cores available for the hosts running the FT virtual machines. Processors with many cores do not help and often have unfavorable performance trade-offs.
- Use the largest HDFS block size possible. This minimizes the block throughput and the load on both the NameNode and JobTracker, and also maximizes application efficiency. Larger block sizes generally require more memory resources (Java heap, io.sort.mb, and so on) to be effective.
- Ensure the networking infrastructure can handle both worker traffic as well as FT logging traffic, or separate the FT logging traffic onto a dedicated network.
- For larger clusters, protect just the NameNode with FT and place the JobTracker in an SMP virtual machine.

Conclusion

The results presented here for a 24-host cluster show that vSphere FT can be used to protect the Hadoop NameNode and JobTracker from physical and virtual hardware failures with minimal overhead for clusters up to 276 similar hosts running over 1100 worker virtual machines. The assumptions used to arrive at this somewhat extreme extrapolation need to be verified on larger clusters, and also with other applications. However, the extrapolations were conservative with respect to CPU utilization, and two of the applications used probably place larger loads on the NameNode and JobTracker than most applications.

Appendix: Configuration

Hardware

- Cluster: 24X HP DL380 G7
- Host CPU and memory
  - 2X Intel Xeon X5687 processors, 3.6GHz, 12MB cache
  - 72GB, 6X 8GB and 6X 4GB DIMMs, 1333 MHz
- Host BIOS settings
  - Intel Hyper-Threading Technology: enabled
  - C-states: disabled
  - Enhanced Intel SpeedStep Technology: disabled
  - Intel TurboMode: disabled
- Host local storage controller
  - HP Smart Array P410i, PCIe 2.0 x4 host interface
  - Hardware revision: Rev C
  - Firmware version: 5.12
  - 1024MB cache
- Host local storage disks
  - 16X Seagate ST9146852SS, SAS 6Gb/s, 146GB, 15,000 RPM, 16MB buffer (HP EH0146FAWJB)
    - Firmware: HPDF
  - 2 striped disks for VM storage
  - 12 disks for Hadoop data
- Host FC HBA
  - QLogic QLE 2560-SP, 8Gb/s FC
  - Directly connected to SAN (both hosts to SP-A)
- SAN
  - EMC VNX7500
  - Unisphere 05.31.000.5.704

- SAN LUN
  - RAID-5
  - Disks: SATA Flash 180GB (SATA-SAM SS160520 CLAR200)
  - 6 disks per RAID group, 1 LUN per RG
  - SP read and write caches enabled
  - Cache page size: 8KB
- Host network adapter: Broadcom Corporation NetXtreme II BCM57711 10Gb PCIe
- Network switch: Arista 7124SX, 24 ports, 10GbE, optical

Hypervisor

- vSphere 5.0 U1 GA, build 623860
- Single vSwitch on the 10GbE NIC
  - VMkernel port for FT logging traffic
  - Virtual Machine Port Group for VM traffic

Virtual Machines

- Virtual hardware: version 8
- VMware Tools: installed
- Virtual network adapter: vmxnet3
- Virtual SCSI controller: LSI Logic Parallel
- Disks: Physical Raw Device Mappings
- Worker VMs
  - 17400MB, 17,075MB reserved, 4 vCPUs, up to 3 data disks
  - 2 VMs pinned to each socket
  - VM root disk on local storage
- NameNode and JobTracker VMs
  - 2048MB, all reserved, 1 vCPU
  - No pinning
  - VM root disk on SAN

Linux

- Distribution: RHEL 6.1 x86_64
- Kernel parameters in /etc/security/limits.conf
  - nofile=16384
  - nproc=4096
- Kernel parameters in /etc/sysctl.conf
  - fs.epoll.max_user_instances=4096
- Kernel parameters in /etc/selinux/config
  - SELINUX=disabled
- Java: Sun Java 1.6.0_26
- File system
  - Single Linux partition aligned on 64KB boundary per hard disk or LUN
  - Partition formatted with EXT4
    - 4KB block size
    - 4MB per inode
    - 2% reserved blocks

Hadoop

- Distribution: Cloudera CDH3u4 (Hadoop 0.20.2)
- NameNode, JobTracker: Dedicated VMs
- Client and Secondary NameNode: each in one of the worker VMs
- Workers: 92 VMs
- Non-default parameters
  - Number of tasks: Table 1
  - dfs.datanode.max.xcievers=4096
  - dfs.replication=2
  - dfs.block.size: Table 2
  - dfs.datanode.readahead.bytes=4194304
  - dfs.datanode.drop.cache.behind.writes=true
  - dfs.datanode.sync.behind.writes=true
  - dfs.datanode.drop.cache.behind.reads=true
  - dfs.permissions=false
  - io.file.buffer.size=131072
  - mapred.map.child.java.opts="-Xmx1900m –Xmn512m"
  - mapred.reduce.child.java.opts="-Xmx1900m –Xmn512m"
  - mapred.child.ulimit=4000000
  - mapred.map.tasks.speculative.execution=false
  - mapred.reduce.tasks.speculative.execution=false
  - mapred.job.reuse.jvm.num.tasks=-1
  - io.sort.mb=400
  - io.sort.record.percent=0.2
- Cluster topology: each host is a unique rack

References

[1] White, Tom. *Hadoop: The Definitive Guide*. O'Reilly, 2012.

[2] vSphere Availability. VMware, Inc., 2012. http://pubs.vmware.com/vsphere-50/topic/com.vmware.ICbase/PDF/vsphere-esxi-vcenter-server-501-availability-guide.pdf.

[3] "Hortonworks Delivers Proven High-Availability Solution for Apache Hadoop," press release. Hortonworks, 2012. http://hortonworks.com/about-us/news/hortonworks-delivers-proven-high-availability-solution-for-apache-hadoop.

[4] "VMware vSphere 4 Fault Tolerance: Architecture and Performance," technical white paper. VMware, Inc., 2009. http://www.vmware.com/resources/techresources/10058.

[5] Collins, Eli. "Hadoop Availability," blog. Cloudera, 2011. http://www.cloudera.com/blog/2011/02/hadoop-availability/.

[6] Buell, Jeff. "A Benchmarking Case Study of Virtualized Hadoop Performance on VMware vSphere 5," technical white paper. VMware, Inc., 2011. http://www.vmware.com/resources/techresources/10222.

About the Author

Jeff Buell is a Staff Engineer in the Performance Engineering group at VMware. His work focuses on the performance of various virtualized applications, most recently in the HPC and Big Data areas. His findings can be found in white papers, blog articles, and VMworld presentations.

Acknowledgements

The work presented here was made possible through a collaboration with EMC, who provided the VNX7500 as well as extensive technical expertise. The author also thanks many of his colleagues at VMware for their valuable suggestions.

vmware

VMware, Inc. 3401 Hillview Avenue Palo Alto CA 94304 USA Tel 877-486-9273 Fax 650-427-5001 www.vmware.com