code{4}lib JOURNAL

Issue 22, 2013-10-14 ISSN 1940-5758

Fedora Commons With Apache Hadoop: A Research Study

The Digital Collections digital repository at the University of Maryland Libraries is growing and in need of a new backend storage system to replace the current filesystem storage. Though not a traditional storage management system, we chose to evaluate Apache Hadoop because of its large and growing community and software ecosystem. Additionally, Hadoop's capabilities for distributed computation could prove useful in providing new kinds of digital object services and maintenance for ever increasing amounts of data. We tested storage of Fedora Commons data in the Hadoop Distributed File System (HDFS) using an early development version of Akubra-HDFS interface created by Frank Asseg. This article examines the findings of our research study, which evaluated Fedora-Hadoop integration in the areas of performance, ease of access, security, disaster recovery, and costs.

By Mohamed Mohideen Abdul Rasheed

Introduction

Digital content managed by libraries is growing rapidly. Libraries are already moving from terabyte to petabyte scale storage platforms (Altman et al. 2013). This huge volume of digital content makes its storage and maintenance a challenging task. Native filesystems do not scale to the demanding requirements of today's digital libraries. Libraries, in order to meet their terabyte scale storage requirements, commonly use networked storage solutions such as SAN and NAS. These storage platforms isolate the storage from the application server so that the high volume of access requests is not choked by the application server's storage bottlenecks.

The University of Maryland Libraries is currently reevaluating its digital library infrastructure to meet its growing needs. In this research study we investigate a scalable storage platform for the libraries' digital collections repository. Apache Hadoop, an open source big data platform that provides a scalable, robust, and distributed storage and processing framework, is our primary contender for numerous reasons. This article summarizes our evaluation of Hadoop's ability to serve as a backend-storage for Fedora Commons repository.

What is Hadoop?

Hadoop is a distributed computing framework. Hadoop Distributed File System (HDFS) and MapReduce are the two key components of Hadoop. HDFS is a solid storage platform that can easily handle petabyte scale data. Major features of HDFS include data integrity, availability, and fault-tolerance. MapReduce is a distributed processing framework that can distribute tasks to the nodes in the cluster, and aggregate the results efficiently. For any application to utilize Hadoop's distributed processing framework it has to be designed conforming to the MapReduce application architecture.

Why Hadoop?

Hadoop is highly scalable. The ability to scale horizontally without compromising performance is an important trait of Hadoop. Clusters as large as 4500 nodes exist at Yahoo. Hundred node Hadoop clusters are a commonplace (PoweredBy – Hadoop Wiki … [updated 2013]). Fault-tolerance is another noteworthy feature of Hadoop. It can handle disk, node, or network failures elegantly. In case of a node failure, HDFS automatically copies data that fall below the configured replication factor to other nodes to ensure data availability. Failed nodes can be replaced without downtimes or manual restoration operations. HDFS can also be configured to automatically balance data across nodes.

In addition to the Hadoop HDFS data storage, Hadoop's MapReduce serves as a powerful data processing engine.

Digital library processes involving computationally intensive tasks such as image format migration, document recognition, or pattern identification could benefit from MapReduce. For example, MapReduce applications can be designed with facial recognition tools to generate complementary metadata for digital image collections.

Hadoop has an active group of dedicated contributors and it is supported by a host of successful startups. Hadoop's ability to run on commodity hardware, efficient large-file handling, and ever-growing capabilities are other factors that made Hadoop a top candidate for our investigation.

Architecture

Akubra, a pluggable storage module, was developed under a collaborative effort between Fedora Commons and Topaz, an open source content modeling and storage software framework project. It was designed to provide a consistent low-level storage interface across Fedora Commons and other related projects. Akubra was first incorporated into Fedora Commons in version 3.2 and has been the default storage interface since version 3.4. It allows Fedora to connect to any storage platform that has a compatible interface implementation. Popular platforms such as IRODS and Dell DX object storage have Akubra compatible interfaces available [1]. Akubra-HDFS (Asseg et al. 2012), an experimental project by Frank Asseg, provides a basic Akubra interface implementation for HDFS [2]. We used Akubra-HDFS augmented with some customizations for this experiment.

Akubra-HDFS is available as an open source project. Although the Akubra-HDFS interface is functional, it needed a few customizations to successfully integrate Fedora with Hadoop. We enhanced Akubra-HDFS by adding a cache feature, making it more configurable, and enabling support for secure Hadoop clusters. The cache feature can be used to selectively cache files from HDFS to the local filesystem based on the path suffix. We adapted Android Image Cache [3], a library that caches images to an android device's local storage for conserving network bandwidth, to serve as a simple Linux file cache [4]. Next, the enhancements allow us to utilize the standard Hadoop configuration files for configuring the Akubra-HDFS's Hadoop client. All our customizations are available on github [5]. We intend to contribute the enhancements back to the original project.

HDFS is optimized for large objects. However, for small files the performance efficiency could be undermined by the network overhead of HDFS (White 2009). The cache module can overcome this problem to a certain extent. We have configured the Akubra-HDFS to cache image thumbnail files to the Fedora server's local disk. The cache module serves HDFS requests for configured files from the cache on local disk instead of retrieving them from HDFS. On a request for a file that is not yet cached, the requested file is retrieved from HDFS and added to the cache.

Fedora is very flexible in configuring the storage interface. The objects and datastreams can be configured to use different storage platforms. In this case, we configured Fedora to use Akubra-HDFS for the datastreams, and the local filesystem for the objects. This again limits the number of small files that need to go into Hadoop.

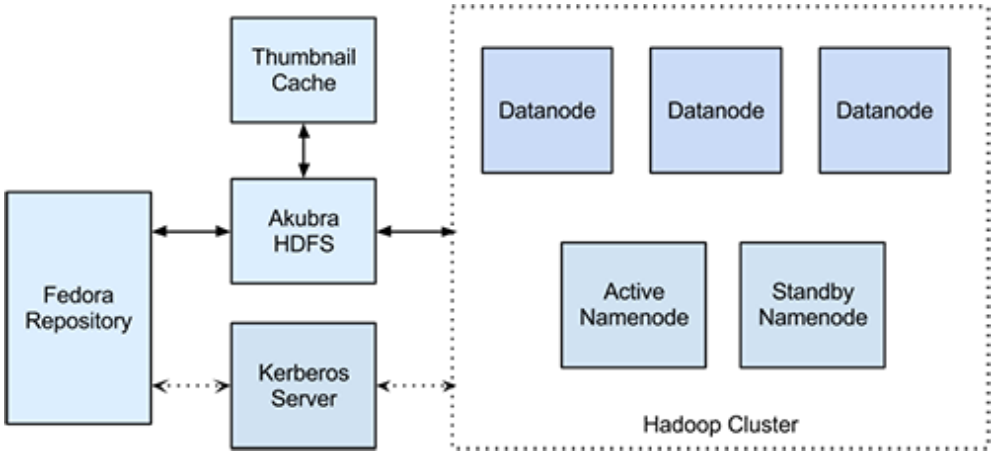


Figure 1. High level architecture of the Fedora-Hadoop integration.

Hadoop Architecture

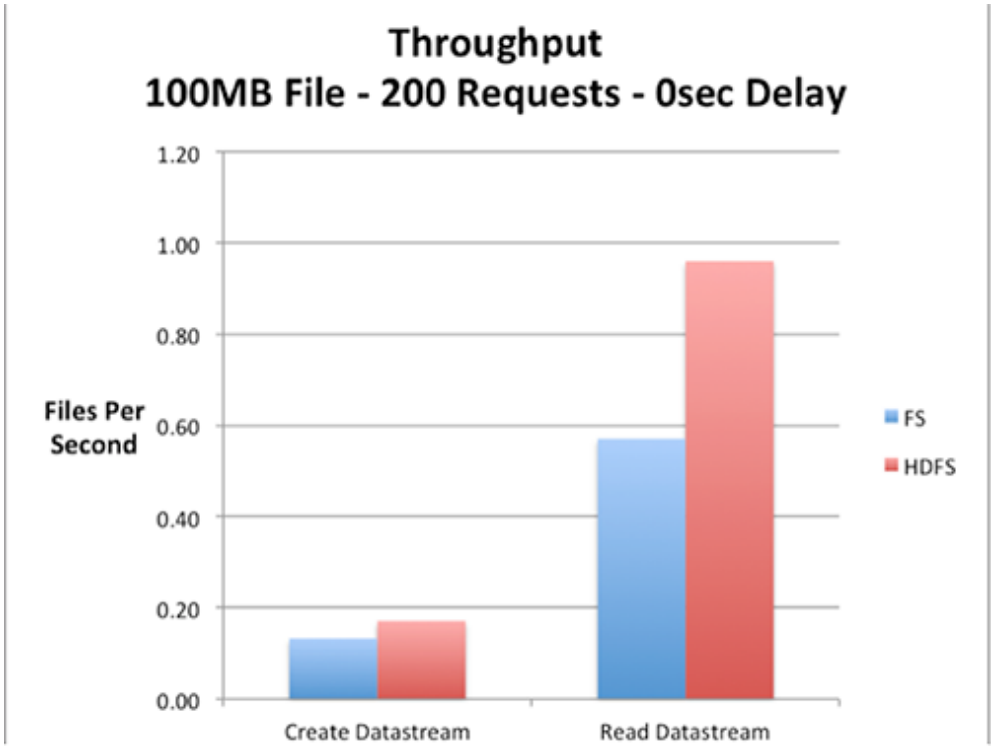
Typically, a Hadoop cluster consists of a single namenode and multiple datanodes. Hadoop can handle datanode failures very well, but if the namenode fails the entire cluster goes down. To overcome such a scenario a hot-standby namenode can be added to the cluster making the namenode Highly Available. The standby node will always be in sync with the active namenode and it can automatically take over the cluster when the active namenode fails.

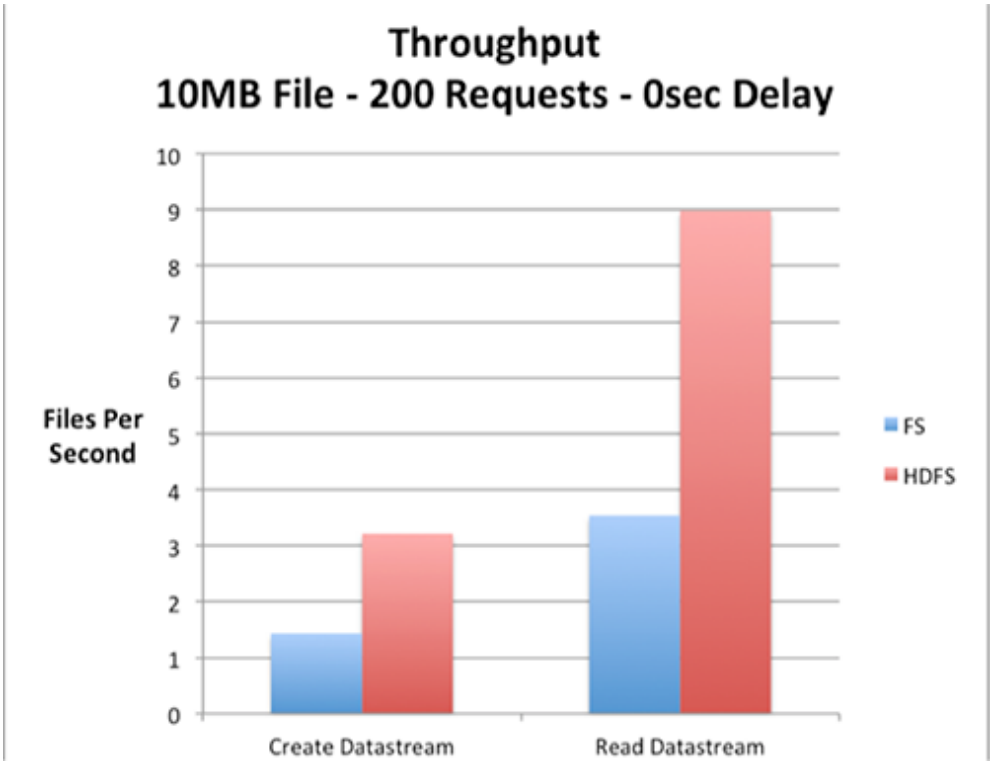
The namenode manages the metadata of the HDFS, whereas the datanodes host the actual content. All client requests to HDFS are first processed by the namenode, and it redirects clients to appropriate datanodes. Hadoop uses redundancy to achieve fault tolerance, storing copies of the same data on different nodes so that when one node fails the data can still be retrieved from other locations. The number of copies Hadoop stores is configurable, with a default setting of three copies. The effective storage capacity of the cluster becomes one-third of the actual capacity under this configuration. Hadoop uses an xml based configuration framework. Every node in Hadoop needs matching, correctly configured xml files for proper functioning of the cluster. We used Cloudera Standard [6], an easy to use tool to deploy and manage Hadoop clusters, from Cloudera [7], to administer the test cluster for this research.

Performance

To understand the performance of Hadoop storage, it needs to be contrasted with the regular storage. We benchmarked the performance of HDFS backed Fedora against Fedora with the default storage platform. The Fedora server application is run on a Linux virtual machine with SAN provisioned storage. See Appendix I for specification of the systems used in this research. We used Apache JMeter [8], a performance-testing tool, to conduct the tests. JMeter can simulate high workload conditions by generating large numbers of parallel requests to the server. JMeter captures different aspects of the response from the server to provide valuable performance information.

Create object, delete object, create datastream, read datastream, and delete datastream operations were benchmarked. As expected the create object and delete object operations had similar performance on both HDFS and local filesystem based servers, since the Fedora objects were configured to be stored in the local storage in both cases. The delete datastream operation also had a close performance on both servers, as it only involved metadata operations. The create datastream and read datastream operations showed significant performance improvement on the HDFS backed server.





Figures 2 and 3. Throughput comparisons of the Fedora server with local filesystem and HDFS.

The charts above depict the throughput of the Fedora server with local filesystem and HDFS for 200 parallel requests generated by JMeter. Multiple JMeter clients were used to maximize the parallelism of the test requests.

The results are dependent on factors such as network bandwidth, disk performance, and SAN performance, so it cannot be generalized that HDFS storage will always provide better performance. Rather, to obtain a more accurate comparison the above-mentioned factors pertaining to the specific environment should be studied. We have hosted our JMeter test plans on github [9]. The test plans can be used to rerun the performance tests on any Fedora server.

Ease of Access

Easy access to the storage is an important requirement for any filesystem. Digital collections staff, and other potential users would need direct HDFS access to perform day-to-day data handling operations. The functionality to natively mount HDFS to users' workstations is currently limited. However, additional services can be installed on the server side that allows easier access to the filesystem.

The available options include

1. Hue File Browser
2. FUSE-HDFS
3. HttpFS
4. NFS and WebDAV

Hue File Browser

Hue [10], Hadoop User Experience, is an open source graphical user interface for Hadoop developed by Cloudera. The

Hue file browser exposes a web-accessible interface to HDFS. It is rich in functionality and very intuitive. The interface can be used to upload, download, or delete files and folders in HDFS. The file upload is simplified by the drag and drop interface. A folder should be zip archived before upload, and Hue will automatically extract the archive once uploaded. The file browser's features like permissions management and text editors are very handy [11].

FUSE-HDFS

FUSE [12], Filesystem in Userspace, allows external filesystems to be mounted to Unix machines. It provides users the convenience of accessing remote filesystems similarly to local filesystems. The FUSE-HDFS module can be used to mount HDFS to Linux machines. An easy guide to setup FUSE-HDFS is available in the Cloudera website [13]. FUSE-HDFS is not recommended as a production ready utility, as there are several disadvantages. The performance concerns of FUSE mounted filesystem and the lack of support for Windows OS users are notable.

HttpFS

HttpFS is a proxy service that allows clients to access HDFS via a HTTP Rest API. File transfer operations in HttpFS can be performed using a command line tool like curl. A custom graphical user interface can be built on top of the Rest API to provide users with a remote file client. Alternatively, HttpFS can also be accessed using the Hadoop FileSystem API for easier integration with other applications. HttpFS service has built-in security to authenticate clients on secure clusters. Also, as a proxy server it improves security by eliminating datanode visibility to clients outside the firewall (Hadoop HDFS over … [updated 2013]).

NFS and WebDAV

NFS and WebDAV plugins for HDFS are in very early stages with NFS support targeted for Hadoop 3.0. The Hadoop project tracker shows an active development effort in progress for the NFS module [14]. The WebDAV module looks like an orphaned project with little or no attention in the past few years [15].

Security

Authentication

Hadoop clusters can be secured using Kerberos authentication. By default, the security option is disabled to facilitate easy deployment and testing, but this leaves the cluster vulnerable to intruders. After successfully deploying the cluster, Kerberos security can be enabled by configuring the necessary properties in the Hadoop xml configuration files. Also, the Kerberos client needs to be installed and configured in all the nodes. Once security is enabled, all Hadoop services will need Kerberos security credentials to communicate with each other. The Cloudera Standard cluster management tool makes this process straightforward [16]. The HDFS clients need to have valid tokens from the Kerberos server to be able to access the secure cluster.

Many libraries might already be using Kerberos for authenticating their users in other applications. The same Kerberos server can be used to authenticate the Hadoop users. However, for authenticating Hadoop processes and services, a dedicated Kerberos server is recommended. This will prevent any accidental overload caused by faulty MapReduce user applications from affecting non-Hadoop services that depend on the same Kerberos server. When two separate Kerberos servers are used, a trust relationship needs to be established between them [17].

Though Kerberos adds reliable security to a Hadoop cluster, Hadoop is still evolving its security infrastructure to make it more flexible, fine-grained, and interoperable. Intel's Project Rhino [18], Cloudera's Sentry [19], and Hortonworks Knox [20] are major security improvisation efforts undertaken by a few vendors.

Access Control

Once Kerberos establishes the identity of users, Hadoop allows users to access the filesystem in accordance with their

permissions. HDFS uses an authorization model similar to POSIX. File and folder permissions can be set using the read, write, and execute flags (Permissions Guide … [updated 2013]).

Users & Groups

By default, Hadoop uses Unix users and groups information for managing access to HDFS. Hadoop expects all user accounts to exist in every node. Creating and managing user accounts in each node of the cluster will not be practical. Instead, Hadoop can be configured to use LDAP for user information. As most institutions already use LDAP or compatible services to manage their users, this will help simplify and centralize Hadoop user management. Hadoop can either use LDAP or Unix user management, but not both simultaneously.

Quotas

HDFS provides capabilities to limit the storage usage on a per directory basis. Space quotas limit the maximum number of bytes a directory and its subdirectories can hold and Name quotas limit the maximum number of files a directory and its subdirectories can have. Unlike other storage platforms, quotas in HDFS are not user or group specific. The dfsadmin tool can be used to set, clear, and monitor quotas in HDFS. More information on quotas can be found at the Hadoop documentation website [21].

Disaster Recovery

Hadoop does not have a solid disaster recovery option yet. Currently, a secondary cluster needs to be used for disaster recovery, which is very inefficient. In using the secondary cluster, Hadoop users can choose between a mirrored cluster and a passive backup cluster. In the mirrored cluster approach, every transaction is duplicated to two clusters, and all the processing happens twice. This option is suitable for highly critical systems where data loss is completely unacceptable. In the backup cluster approach, the data from the main cluster is periodically backed up to the second cluster using DistCp, a MapReduce based parallel copy tool. Systems that are less stringent with data recovery requirements can use the second option, as there is a risk of losing data changes after the last backup (Ranganathan 2013).

Efficient disaster recovery options need to be built into Hadoop. Meanwhile, regular backup options such as offsite tape backups can be used. HDFS can be mounted to Linux using FUSE-HDFS, and then backed up using regular tape backup tools. Similarly, other disaster recovery tools can be used with the mounted HDFS. However, this approach might not be performance efficient.

Snapshots

Snapshot is an essential feature for the digital library storage. It helps recover data lost in file corruptions, accidental user errors, and other such scenarios. Hadoop does not have the snapshot feature in the current version. The snapshot feature is being actively developed, and it will be available in the near future (Sze 2012). The initial versions will support read-only snapshots that are sufficient for the restore functionality. The read-write support will be added consecutively. The status of the feature can be followed at its project tracker page [22]. Fedora's built-in content versioning functionality should be relied upon until the HDFS Snapshot feature is ready.

Costs

Cost is one of the deciding factors for adoption of any technology. This is especially true with budget cautious libraries. Running on low-cost hardware is one of the features Hadoop touts. Hadoop is designed to handle failures, and it does not require high-cost fault tolerant hardware. A well-planned cluster with well chosen hardware can balance between cost and performance.

The primary objective of the Hadoop cluster here is to provide a scalable and robust storage service. For storage-optimized clusters, each node should have multiple high volume storage disks with sufficient memory and processing resources. Though some servers can support a large number of storage disks, there are other factors that limit the number of disks per node. Data distribution will be impaired by too much storage per node. Cloudera recommends staying within twelve 1TB or 2TB disks per node (Loddengaard 2010). Data locality is an important factor for MapReduce

applications to succeed, and it can be compromised on a under-distributed cluster (Lin and Dyer 2010). A well-designed cluster would have maximized resource usage, and reduced investment on under utilized resources.

The chart below represents the cost per gigabyte and the number of nodes of a Hadoop cluster proposed for University of Maryland libraries. The cost for the initial 8TB is very high as it includes the cost for two namenodes and three 8TB datanodes. For the next 16TB, there is no new node added. The existing datanodes would be upgraded to 24TB each. After that, single 24TB datanodes will be used to add additional capacity to the cluster. Please refer to Appendix II for the costs of the individual nodes.

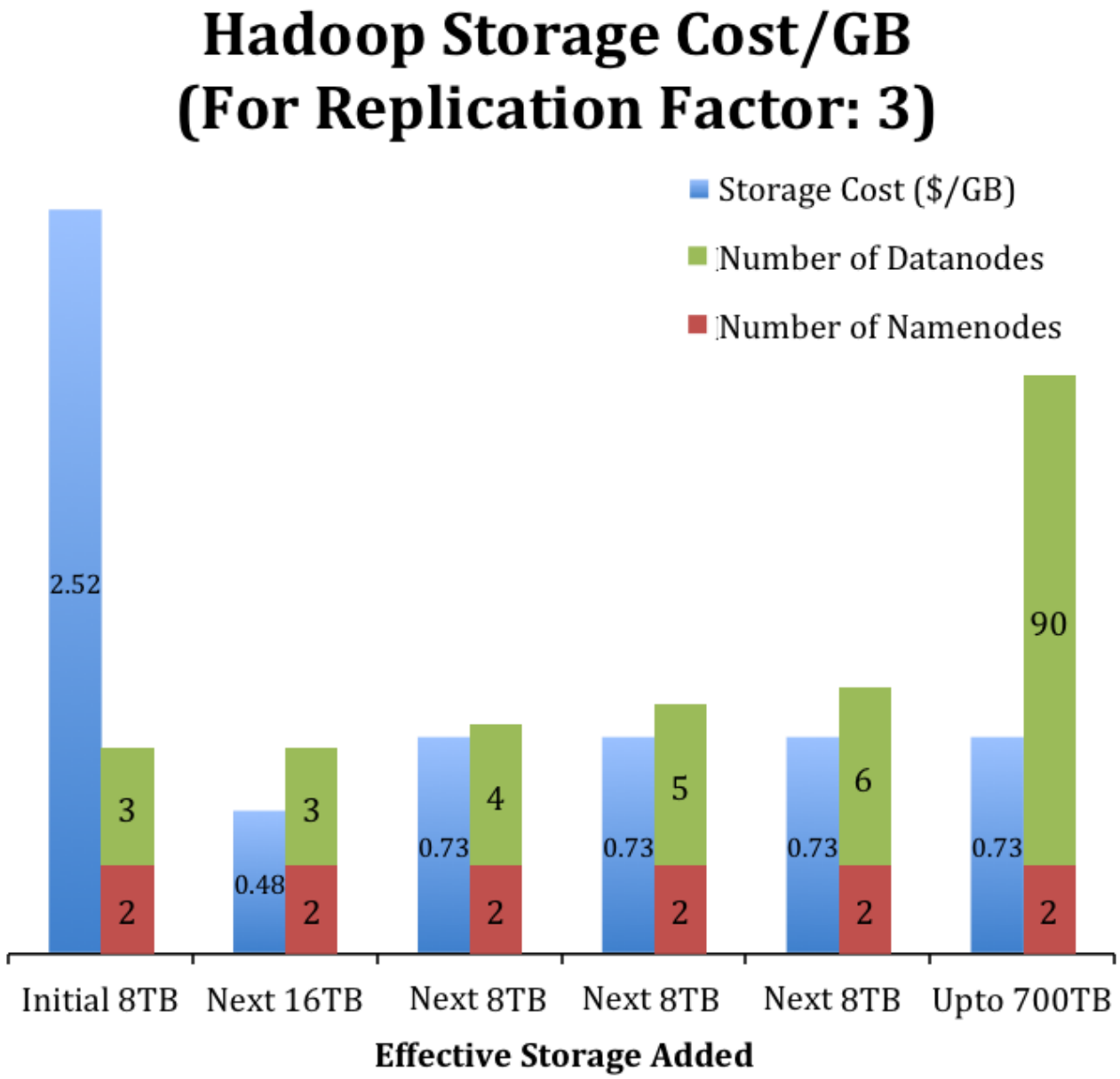


Figure 4. Hadoop storage cost per gigabyte for the University of Maryland libraries.

Next, we investigated the costs involved in setting up an Amazon AWS [23] cloud based Hadoop cluster. Amazon AWS has a complex pricing structure that depends on multiple factors including network bandwidth, storage, and CPU. This makes it difficult to estimate the overall expense. We calculated the aggregate costs of the services for different usage levels and system specifications. In almost every scenario, the costs were very high. The large amount of persistent storage required for the digital collection was one of the main factors that drove the cost up. Please see Appendix III for the AWS cost tables. Amazon provides Elastic MapReduce (EMR) [24] instances that are optimized for MapReduce

processing services, but they are not well suited for storage-centered clusters. Therefore, we used the standard AWS cloud instances to build and evaluate the Hadoop. Although it would require additional management overhead, running the cluster in-house is the cost-efficient and propitious option for this scenario.

Conclusion

This research study explains the real value Hadoop can bring to libraries. The first hand experience integrating and testing Fedora with Hadoop shows where Hadoop does well, and where it doesn't. Though Hadoop performed well in our experiments, there are a few major areas where it falls short. The lack of a straightforward disaster recovery option is the biggest drawback. There seems to be alternative means to implement disaster recovery using FUSE-HDFS, but it's not proven. Next, the performance tests conducted were on a very small scale – a few gigabytes. Hadoop scales well, but the performance of Akubra-HDFS for large-scale deployments cannot be extrapolated. Akubra-HDFS might need more tests and optimizations before it can be production ready.

Future Research

We are upgrading our test cluster with more nodes and additional resources. This will allow us to run more realistic tests on the cluster, and analyze other critical requirements for Hadoop backed digital libraries. The viability of tape backups with Hadoop needs to be examined. This will help identify the challenges and potential solutions in Hadoop disaster recovery.

Next, the potential for Hadoop HBase [25] to serve as the Fedora object store needs investigation [26]. It would be desirable to use a single storage for all the data so that the overall maintenance and backup processes are simplified. Currently, the Fedora objects are configured to use the local filesystem to mitigate the HDFS small files problem. HBase, the distributed database that runs on top of HDFS, handles small files very well. This will consolidate all data to be stored within Hadoop. HBase can also be tested with other data-driven applications that can profit from its strengths.

Hadoop Solr is another interesting tool to try. It is an implementation of Apache Solr, the popular enterprise search platform, built to work directly on HDFS. Hadoop Solr uses the cluster to provide a scalable and highly available search service. Lastly, the complementary benefits Hadoop can bring to digital libraries needs to be explored. The workflows in the digital object lifecycle that can potentially benefit from Hadoop's distributed processing should be identified and assessed. The workflows that involve time-consuming processes could be good candidates to start with.

Notes

[1] Akubra Project. Available from: https://wiki.duraspace.org/display/AKUBRA/Akubra+Project.

[2] Fasseg/Akubra-HDFS. Available from: https://github.com/fasseg/akubra-hdfs

[3] Mimtel/Android-Image-Cache. Available from: https://github.com/mitmel/Android-Image-Cache

[4] Umd-Lib/Filecache. Available from: https://github.com/umd-lib/filecache

[5] Umd-Lib/Akubra-HDFS. Available from: https://github.com/umd-lib/akubra-hdfs

[6] Cloudera Standard. Available from: http://www.cloudera.com/content/cloudera/en/products/cloudera-standard.html

[7] Cloudera, Inc. Available from: http://www.cloudera.com/content/cloudera/en/home.html

[8] Apache JMeter. Available from: http://jmeter.apache.org/

[9] JMeter Test Plans. Available from: https://github.com/umd-lib/jmeter-testplans

[10] The UI for Apache Hadoop. Available from: http://cloudera.github.io/hue/

[11] Demo: HDFS File Operations Made Easy with Hue. Available from: http://blog.cloudera.com/blog/2013/04/demo-

hdfs-file-operations-made-easy-with-hue/

[12] Filesystem in Userspace. Available from: http://fuse.sourceforge.net/

[13] Mountable HDFS. Available from: http://www.cloudera.com/content/cloudera-content/cloudera-docs/CDH4/latest/CDH4-Installation-Guide/cdh4ig_topic_28.html

[14] Support NFSv3 interface to HDFS. Available from: https://issues.apache.org/jira/browse/HDFS-4750

[15] Expose HDFS as a WebDAV store. Available from: https://issues.apache.org/jira/browse/HDFS-225

[16] Configuring Hadoop Security with Cloudera Manager. Available from: http://www.cloudera.com/content/cloudera-content/cloudera-docs/CM4Ent/latest/Configuring-Hadoop-Security-with-Cloudera-Manager/Configuring-Hadoop-Security-with-Cloudera-Manager.html

[17] Set up a Local KDC and Default Domain for the Hadoop Cluster. Available from: http://www.cloudera.com/content/cloudera-content/cloudera-docs/CM4Ent/4.5.3/Configuring-Hadoop-Security-with-Cloudera-Manager/cmeechs_topic_4_3.html

[18] Project Rhino. Available from: https://github.com/intel-hadoop/project-rhino/

[19] Sentry. Available from: http://www.cloudera.com/content/cloudera/en/products/cdh/sentry.html

[20] Apache Knox Gateway. Available from: http://hortonworks.com/hadoop/knox-gateway/

[21] Quotas Guide. Available from: http://hadoop.apache.org/docs/stable/hdfs_quota_admin_guide.html

[22] Support for RW/RO snapshots in HDFS. Available from: https://issues.apache.org/jira/browse/HDFS-2802

[23] Amazon Web Services. Available from: https://aws.amazon.com/

[24] Amazon Electronic MapReduce. Available from: http://aws.amazon.com/elasticmapreduce/

[25] Apache HBase. Available from: http://hbase.apache.org/

[26] Fasseg/Akubra-HBase. Available from: https://github.com/fasseg/akubra-hbase

References

Altman M, Bailey J, Cariani K, Gallinger M, Mandelbaum J, Owens T. 2013. NDSA Storage Report: Reflections on National Digital Stewardship Alliance Member Approaches to Preservation Storage Technologies. D-Lib Magazine [Internet]. [cited 2013 May]; 19:5/6. Available from: http://www.dlib.org/dlib/may13/altman/05altman.html

PoweredBy – Hadoop Wiki [Internet]. [updated 2013 Jun 19] Forest Hill (MD): Apache Software Foundation; [cited 2013 Aug 13]. Available from: http://wiki.apache.org/hadoop/PoweredBy

Asseg F, Razum M, Hahn M. 2012. Apache Hadoop as a Storage Backend for Fedora Commons. In: OR2012. The 7th International Conference on Open Repositories; 2012 Jul 9-12. Available from: http://www.scape-project.eu/publication/apache-hadoop

White T. 2009. The Small Files Problem [Internet]. Palo Alto (CA): Cloudera, Inc; [cited 2013 Aug 13]. Available from: http://blog.cloudera.com/blog/2009/02/the-small-files-problem/

Hadoop HDFS over HTTP [Internet]. [updated 2013 Jun 8] Forest Hill (MD): Apache Software Foundation; [cited 2013 Aug 13]. Available from: http://hadoop.apache.org/docs/current/hadoop-hdfs-httpfs/

Permissions Guide [Internet]. [updated 2013 Aug 4] Forest Hill (MD): Apache Software Foundation; [cited 2013 Aug 13]. Available from: http://hadoop.apache.org/docs/stable/hdfs_permissions_guide.html

Ranganathan J. 2013. Hadoop Backup and Disaster Recovery. Big Data TechCon Boston; 2013 April 8-10. Available from: http://www.slideshare.net/cloudera/hadoop-backup-and-disaster-recovery

Sze N. 2012. Community-driven Snapshot for HDFS – Part TWO [Internet]. Palo Alto (CA): Hortonworks; [cited 2013 Aug 13]. Available from: http://hortonworks.com/enterprise-ready-community-driven-apache-hadoop/

Loddengaard A. 2010. Cloudera's Support Team Shares Some Basic Hardware Recommendations [Internet]. Palo Alto (CA): Cloudera, Inc; [cited 2013 Aug 13]. Available from: http://blog.cloudera.com/blog/2010/03/clouderas-support-team-shares-some-basic-hardware-recommendations/

Lin J, Dyer C. 2010. Data-Intensive Text Processing with MapReduce [Internet]. San Rafael (CA): Morgan & Claypool Publishers. p. 26-28. Available from: http://lintool.github.io/MapReduceAlgorithms/

About the Author

Mohamed is a software developer at the University of Maryland (UMD) Libraries. He is a recent alumnus of UMD's iSchool, where he earned his masters in Information Management. Author's email: mohideen@umd.edu LinkedIn: http://www.linkedin.com/in/mohideenrasheed

Appendix

I. System Specifications

Servers

1. Fedora

   Fedora server runs on a linux virtual machine.
   CPU: Intel Xeon X5670 2.93GHz (Quad core)
   Memory: 4GB
   Storage: 80GB (SAN Storage)
   OS: CentOS 6.4

2. Test Hadoop Cluster

   Dell Optiplex 755 desktop machines are used for Hadoop..
   Name Nodes: 2 (1 Active, 1 Failover)
   Data Nodes: 3
   Dell Optiplex 755
   CPU: Intel E2200 2.20GHz (Dual core)
   Memory: 4GB
   Storage: 80GB HDD
   OS: CentOS 6.4

3. Kerberos

   Kerberos server runs on a linux virtual machine.
   CPU: Intel Xeon X5570 2.93GHz (Dual core)
   Memory: 4GB
   Storage: 26GB (SAN Storage)
   OS: CentOS 6.4

Software Versions

1. Fedora Version 3.4.2
2. Hadoop
3. Cloudera Standard 4.6.0. (Earlier known as: Cloudera Manager)
4. Cloudera Distributed Hadoop CDH 4.3.
5. Kerberos Version 5

II. Hadoop Cluster Costs

Cost of Dell PowerEdge Rack Servers:

1. Namenode Server – $5299USD

   CPU: 2 Intel Xeon 2.53GHz 6 core
   Memory: 64GB
   HDD: 500GB * 3

2. 8TB Datanode Server – $3368USD

   CPU: 2 AMD Opteron 3.1Ghz 8 core
   Memory: 16GB
   HDD: 2TB * 4

3. 24TB Datanode Server – $5987USD

   CPU: 2 AMD Opteron 3.1Ghz 8 core
   Memory: 16GB
   HDD: 2TB * 12

Cost of 2TB HDD – $325USD

The Dell website is used to calculate the prices (Calculated on April 2013).

III. AWS PRICING

The costs below were calculated on May, 2013 from Amazon AWS website and represented in US dollars.

Plan	Effective Storage (TB)	Nodes	Upfront Charge	Upfront Charge Monthly Breakdown	Usage Hours/Day	Usage Cost/Month (~21days)	Total Instance Cost/Month	Cost Type
1 Year	48TB	3	$9,024.00	$752.00	2	$223.43	$975.43	Base
3 Year	48TB	3	$13,587.00	$377.42	2	$180.44	$557.86	Base
1 Year	Every 24TB	+1	$3,968.00	$330.67	2	$97.07	$427.73	Adds to Base
3 Year	Every 24TB	+1	$5,997.00	$166.58	2	$78.43	$245.02	Adds to Base
1 Year	48TB	3	$9,024.00	$752.00	4	$446.85	$1,198.85	Base
3 Year	48TB	3	$13,587.00	$377.42	4	$360.88	$738.30	Base
1 Year	Every 24TB	+1	$3,968.00	$330.67	4	$194.13	$524.80	Adds to Base
3 Year	Every 24TB	+1	$5,997.00	$166.58	4	$156.87	$323.45	Adds to Base

Figure 5. Instance Cost (Replication factor of two is used. The effective storage is half the actual storage).

Transfer Out / Month	Cost
1GB	$0.00
1TB	$122.88
2TB	$245.76
5TB	$614.40
10TB	$1,228.80
20TB	$2,150.40
50TB	$4,915.20
100TB	$8,499.20

Figure 6. Instance Data Transfer Cost

Storage	Cost / Month
10TB	$834.56
50TB	$4,111.36
100TB	$7,695.36
200TB	$14,863.36

Figure 7. S3 Storage Cost

Transfer Out / Month	Cost / Month
1GB	$0.00
1TB	$122.88
2TB	$245.76
5TB	$614.40
10TB	$1,228.80
20TB	$2,150.40
50TB	$4,915.20
100TB	$8,499.20

Figure 8. S3 Data Transfer Cost

Request Type	Requests	Cost
PUT	100,000	$0.50
PUT	1,000,000	$5.00
PUT	10,000,000	$50.00
PUT	100,000,000	$500.00
GET	100,000	$0.04
GET	1,000,000	$0.40
GET	10,000,000	$4.00
GET	100,000,000	$40.00

Figure 9. S3 Request Cost