Mambo
Running Analytics on Enterprise Storage
Jingxin Feng, Xing Lin¹, Gokul Soundararajan
Advanced Technology Group
¹ University of Utah
Motivation

No easy way to analyze data stored in enterprise storage (NFS)
Motivation

No easy way to analyze data stored in enterprise storage (NFS)
Motivation

No easy way to analyze data stored in enterprise storage (NFS)
Motivation

No easy way to analyze data stored in enterprise storage (NFS)
Motivation
No easy way to analyze data stored in enterprise storage (NFS)

- Bank
- AutoSupport

Production System

Analytics System
Motivation

No easy way to analyze data stored in enterprise storage (NFS)
Motivation

No easy way to analyze data stored in enterprise storage (NFS)

- Separate infrastructures for production systems and analytics systems
Motivation

No easy way to analyze data stored in enterprise storage (NFS)

- Separate infrastructures for production systems and analytics systems
Motivation

No easy way to analyze data stored in enterprise storage (NFS)

- Separate infrastructures for production systems and analytics systems

Two isolated data silos
Motivation

No easy way to analyze data stored in enterprise storage (NFS)

- Separate infrastructures for production systems and analytics systems
- Problems
  - Copying PBs of data is time consuming
Motivation

No easy way to analyze data stored in enterprise storage (NFS)

- Separate infrastructures for production systems and analytics systems

- Problems
  - Copying PBs of data is time consuming
  - $3 \times$ storage overhead in HDFS

**Two isolated data silos**
Motivation
No easy way to analyze data stored in enterprise storage (NFS)

- Separate infrastructures for production systems and analytics systems
- Problems
  - Copying PBs of data is time consuming
  - $3 \times$ storage overhead in HDFS
  - Periodical re-synchronization later on

Two isolated data silos

Production System

Data Copying

Analytics System
Motivation
No easy way to analyze data stored in enterprise storage (NFS)

- Separate infrastructures for production systems and analytics systems
- Problems
  - Copying PBs of data is time consuming
  - 3 × storage overhead in HDFS
  - Periodical re-synchronization later on
  - Legal prevents data copying

Two isolated data silos

Production System

Analytics System

Data Copying
Mambo

An NFS connector, enabling direct analytics for data on Enterprise storage (NFS)
Mambo

An NFS connector, enabling direct analytics for data on Enterprise storage (NFS)
Mambo

An NFS connector, enabling direct analytics for data on Enterprise storage (NFS)

- Remove data copying
- Remove storage overhead (single copy)
- Remove data re-synchronization
- No legal issue
Mambo
An NFS connector, enabling direct analytics for data on Enterprise storage (NFS)

- Remove data copying
- Remove storage overhead (single copy)
- Remove data re-synchronization
- No legal issue

Copying is not required; you can do analytics in-place
## Project History

### From Research to Product

<table>
<thead>
<tr>
<th>Year</th>
<th>Activities</th>
</tr>
</thead>
</table>
| 2011   | • Talked with customers  
        | • Developed initial prototype                                               |
| 2012   | • Madalin Mihailescu refined prototype  
        | • Added a distributed cache  
        | • Obtained traces from UC Berkeley  
        | • Published in FAST’13                                                      |
| 2013 ~ now | • Xing Lin refactored code for Hadoop 2.0  
        | • Optimized for 10 Gb networks  
        | • Obtained legal approval for open-source  
        | • Posted to GitHub  
        | • Customer Proof-of-Concepts (PoCs)  
        | • Pushing to merge into Hadoop |
Use Cases

How many ways can you use Mambo?
Analyze Enterprise Data In-place

User jobs

Job

Compute layer
MapReduce

Resource management layer
Yarn

Generic file system layer
File System

Storage layer
HDFS
Analyze Enterprise Data In-place

- User jobs
  - Job
- Compute layer
  - MapReduce
- Resource management layer
  - Yarn
- Generic file system layer
  - File System
- Storage layer
  - HDFS

*HDFS gets swapped out with NFS*
Analyze Enterprise Data In-place

- HDFS gets swapped out with NFS
- Apache Hadoop does not get modified.
- User programs (jobs) are not modified.

User jobs

Compute layer

Resource management layer

Generic file system layer

Storage layer

• HDFS gets swapped out with NFS*
• Apache Hadoop does not get modified.
• User programs (jobs) are not modified.
Use NetApp FlexClones for creating test environments quickly
- Use a copy of production data for realistic Test/QA environments (e.g., AutoSupport)
Use NetApp FlexClones for creating test environments quickly
- Use a copy of production data for realistic Test/QA environments (e.g., AutoSupport)
Use cloud to Analyze Data

Secondary private storage at a colocation facility (e.g., Equinix), for backup and fast restoration with cloud
Use cloud to Analyze Data

Secondary private storage at a colocation facility (e.g., Equinix), for backup and fast restoration with cloud

1. On-premise (onsite)
2. Colocated facility
3. Public cloud (offsite)

Express interconnection
Use cloud to Analyze Data

Secondary private storage at a colocation facility (e.g., Equinix), for backup and fast restoration with cloud

1. On-premise (onsite)
2. Colocated facility
3. Public cloud (offsite)

Express interconnection

SnapMirror
Use cloud to Analyze Data

Secondary private storage at a colocation facility (e.g., Equinix), for backup and fast restoration with cloud

Launch Hadoop in the cloud and use data on private storage
Design and Implementation
Architecture Overview

Mambo: an NFS client in Java, implementing the Hadoop generic file system API
Architecture Overview

Mambo: an NFS client in Java, implementing the Hadoop generic file system API
Architecture Overview

Mambo: an NFS client in Java, implementing the Hadoop generic file system API
Architecture Overview

Mambo: an NFS client in Java, implementing the Hadoop generic file system API

Copying is not required

Filled the missing piece
Architecture Overview

Mambo: an NFS client in Java, implementing the Hadoop generic file system API

- No changes to Hadoop framework
- No changes to user programs

- Copying is not required

---

**YARN**
*(Cluster Resource Management)*

**FileSystem**
*(Interfaces to interact with storage systems)*

- HDFS
- Amazon S3
- GlusterFS
- Azure
- NFS

*Filled the missing piece*
Tight integration with Hadoop/MapReduce

- Optimized for large sequential I/O (e.g., 1MB IO)
- Commit data to disk only when a task succeeds
- Intelligent prefetching for streaming reads; aware of task sizes
Implementation

- MapReduce
- YARN
- File System
- Computation frameworks
- Resource management layer
- Hadoop generic filesystem API
Implementation

- MapReduce
- YARN
- File System
  - NFS File System
    - FS metadata OPs
    - NFS client protocol
    - File reads
    - File writes

Computation frameworks
Resource management layer
Hadoop generic filesystem API
Implementation

MapReduce ... Computation frameworks
YARN Resource management layer
File System Hadoop generic filesystem API
NFS File System
✓ FS metadata OPs
✓ NFS client protocol
✓ File reads
✓ File writes

NFSv3FileSystem.java
NFSv3FileSystemStore.java
NFSv3InputStream.java
NFSv3OutputStream.java
Implementation

- MapReduce
- YARN
- File System
- NFS File System
  - FS metadata OPs
  - NFS client protocol
  - File reads
  - File writes

Computation frameworks

Resource management layer

Hadoop generic filesystem API

Standard NFSv3 protocol

- NFSv3FileSystem.java
- NFSv3FileSystemStore.java
- NFSv3InputStream.java
- NFSv3OutputStream.java
How to Use it?

- Source code (jar library file)
  - Get code from GitHub
  - Compile the code
  - Install the jar file
    - Copy the jar file to the library directory for Hadoop installation
- Only need to modify two configuration files
  - core-site.xml (hadoop core configuration file)
  - nfs-mapping.json (nfs configuration file)
How to Use it?

- **Source code (jar library file)**
  - Get code from GitHub
  - Compile the code
  - Install the jar file
    - Copy the jar file to the library directory for Hadoop installation

- **Only need to modify two configuration files**
  - core-site.xml (hadoop core configuration file)
  - nfs-mapping.json (nfs configuration file)

- Or just try the Amazon Cloud Formation template with everything configured
## Configure core-site.xml

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fs.defaultFS</td>
<td>hdfs://namenode:54310/</td>
</tr>
</tbody>
</table>

### HDFS

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>fs.defaultFS</td>
<td>nfs://nfsserver:2049/</td>
</tr>
<tr>
<td>fs.nfs.configuration</td>
<td>&lt;path-to-configuration-file&gt;</td>
</tr>
<tr>
<td>fs.nfs.impl</td>
<td>org.apache.hadoop.fs.nfs.NFSv3FileSystem</td>
</tr>
</tbody>
</table>
Configure nfs-mapping.json

- Configurable properties
  - Export path
  - Read/write sizes
  - Split size (Hadoop task granularity)
  - Authentication method (supporting AUTH_NONE or AUTH_UNIX)
  - ...
- Supports multiple controllers (for NetApp clustered ONTAP)
  - Aggregated bandwidth
Performance Evaluation
Highlights from MixApart

- MixApart: NFS connector + data prefetcher + local disk as cache

- Better performance with NFS connector than Hadoop with ingest (18%~26% reduction in job duration)
  - Overlaps data ingest with task computation

- Matches ideal Hadoop (data ingested into HDFS beforehand), with moderate/high data reuse across jobs

---

1 MixApart: De-coupled Analytics for Shared Storage Servers. Madalin Mihai  
Mihai  
Mihai  
Mihaiescu, Gokul Soundararajan, and Cristiana Amza. In FAST ‘13
Scaling experiments
How does the NFS Connector scale with more storage and compute?

28 Nodes (UCS B230M2) with 20 CPU cores and 256 GB RAM

8 Nodes (FAS 8080) with 48 HDDs each and 8 10Gb links each
Scaling TeraGen

![Graph showing the relationship between data size and running time for 1 HDD and 8 HDD scenarios.](image)
Scaling TeraGen

<table>
<thead>
<tr>
<th>Data Size (in GB)</th>
<th>Running Time (Normalized)</th>
</tr>
</thead>
<tbody>
<tr>
<td>279</td>
<td>0.00</td>
</tr>
<tr>
<td>373</td>
<td>0.50</td>
</tr>
<tr>
<td>466</td>
<td>1.00</td>
</tr>
<tr>
<td>559</td>
<td>1.50</td>
</tr>
<tr>
<td>652</td>
<td>2.00</td>
</tr>
<tr>
<td>745</td>
<td>2.50</td>
</tr>
<tr>
<td>838</td>
<td>3.00</td>
</tr>
<tr>
<td>931</td>
<td>3.50</td>
</tr>
<tr>
<td>1024</td>
<td>4.00</td>
</tr>
</tbody>
</table>

© 2015 NetApp, Inc. All rights reserved.
Scaling TeraGen

NFS connector scales well for large datasets.
Scaling TeraSort

Running Time (Normalized) vs Data Size (in GB)

- Blue square line: 1 HDD
- Red diamond line: 8 HDD

Data Size:
- 279 GB
- 373 GB
- 466 GB
- 559 GB
- 652 GB
- 745 GB
- 838 GB
- 931 GB
- 1024 GB

© 2015 NetApp, Inc. All rights reserved.
Overcome NFS server bottleneck

Optimize with Caching
Overcome NFS server bottleneck

Optimize with Caching

- Real workloads are cacheable

Overcome NFS server bottleneck

Optimize with Caching

- Real workloads are cacheable\(^1\)

\(^1\)MixApart: De-coupled Analytics for Shared Storage Servers. Madalin Mihailescu, Gokul Soundararajan, and Cristiana Amza. In FAST ‘13
Overcome NFS server bottleneck

Optimize with Caching

- Real workloads are cacheable\(^1\)

---

\(^1\)MixApart: De-coupled Analytics for Shared Storage Servers. Madalin Mihailescu, Gokul Soundararajan, and Cristiana Amza. In FAST ‘13

Optimize with Caching

Use local disk as cache

Flash cache tier

### Compute nodes

- NFS servers

---

© 2015 NetApp, Inc. All rights reserved.
Overcome NFS server bottleneck

Optimize with Caching

- Real workloads are cacheable

Distributed in-memory cache tier

Compute nodes

Use local disk as cache

Flash cache tier

NFS servers

Next steps

We need your help.
Future Work

• Productization within NetApp
  • Support pNFS protocol
  • Security (Kerberos authentication)
Future Work

- Productization within NetApp
  - Support pNFS protocol
  - Security (Kerberos authentication)

- Integration tests with other frameworks
  - Tachyon, HBase, Spark, and etc.
Future Work

- Productization within NetApp
  - Support pNFS protocol
  - Security (Kerberos authentication)

- Integration tests with other frameworks
  - Tachyon, HBase, Spark, and etc.

- Production System Integration
  - NetApp Auto Support (ASUP) Team
  - Customer systems
We Need Your Help

- Anyone interested
  - Try it out and tell us how it works
  - Filing bugs

- Hadoop committers
  - Help to push NFS connector into Hadoop mainstream

- Help integration tests with other frameworks (Tachyon, HBase, etc)

- Help to improve the code at GitHub!
References

- Connector Information

- Public on GitHub:

- Technical Report:

- Paper at FAST’13
  - MixApart: De-Coupled Analytics for Shared Storage Servers

- If you have any question, please contact
  - Xing.Lin@netapp.com, Gokul.Soundararajan@netapp.com, Jingxin.Feng@netapp.com
Summary

- NetApp NFS connector for Hadoop
  - Allows analytics to use any NFS
  - An open implementation (no proprietary code) – contribute back to Hadoop
  - Works with Apache Hadoop, Apache Spark, Tachyon, and Apache HBase
  - In many cases, only configuration file change is needed (no source code changes)
Summary

- NetApp NFS connector for Hadoop
  - Allows analytics to use any NFS
  - An open implementation (no proprietary code) – contribute back to Hadoop
  - Works with Apache Hadoop, Apache Spark, Tachyon, and Apache HBase
  - In many cases, only configuration file change is needed (no source code changes)

- NetApp NFS connector for Hadoop is being deployed
  - Internal testing with other teams
  - Testing with select customers
Acknowledgements

- Madalin Mihaiescu for starting down this path
- Kaladhar Voruganti, Scott Dawkins, Jeff Heller, AJ Mahajan, and Siva Jayasenan for supporting this effort
- Karthikeyan Nagalingam for validation and customer PoCs
- NetApp AutoSupport team for testing it in production
- NetApp NFS team for continuing the effort
Thank you

Mambo: analyze enterprise data in-place