Technical Product (301)
Horizon Workspace 1.5 - Files

Version 1.0

Rasmus Jensen
rjensen@vmware.com

Overview, Architecturing & Scaling, Sharing, Backup & Restore

Marcello Golfieri
mgolfieri@vmware.com

Components & Inner Workings, Admin Operations, Performance Tuning

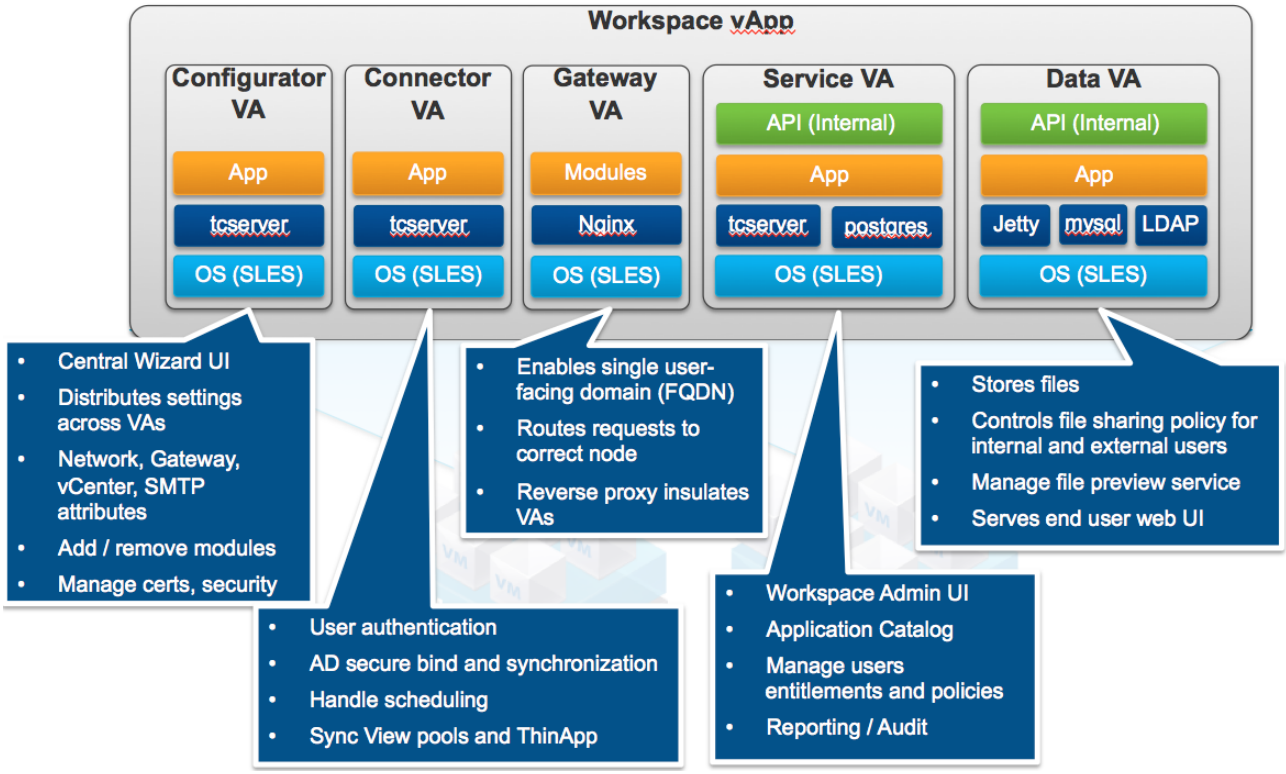
Contents

Overview

VMware Horizon Workspace allows companies to allow secure remote access and collaboration features to its workforce as well as empowering users with easy access to SaaS and virtual applications across multiple devices.
Horizon Workspace Files is the collaboration part of Horizon Workspace that provides an easy and secure on-premise solution for sharing data with colleagues as well as external collaborators.
This document - with a focus on Horizon Workspace Files - is aimed at explaining the technical details of the product as well as how to architect, scale, monitor, operate and tune a Horizon Workspace Files implementation.
This guide is aimed at Specialist SEs, TAMs, and PSO consultants looking to gain deep knowledge of Horizon Workspace Files, how it operates, perform troubleshooting and maintenance operations.

The Horizon Workspace solution

Horizon Workspace is made up of multiple Virtual Appliances (VAs) based on Suse Linux Enterprise Server 11 (SLES) SP2 and is delivered in a vApp OVA container format for easy deployment and initial configuration.
The vApp is initially deployed with all the required VAs to get the solution up and running. The vApp deployment happens against a vCenter Server.
An explanation of the VAs and their purpose is shown below.

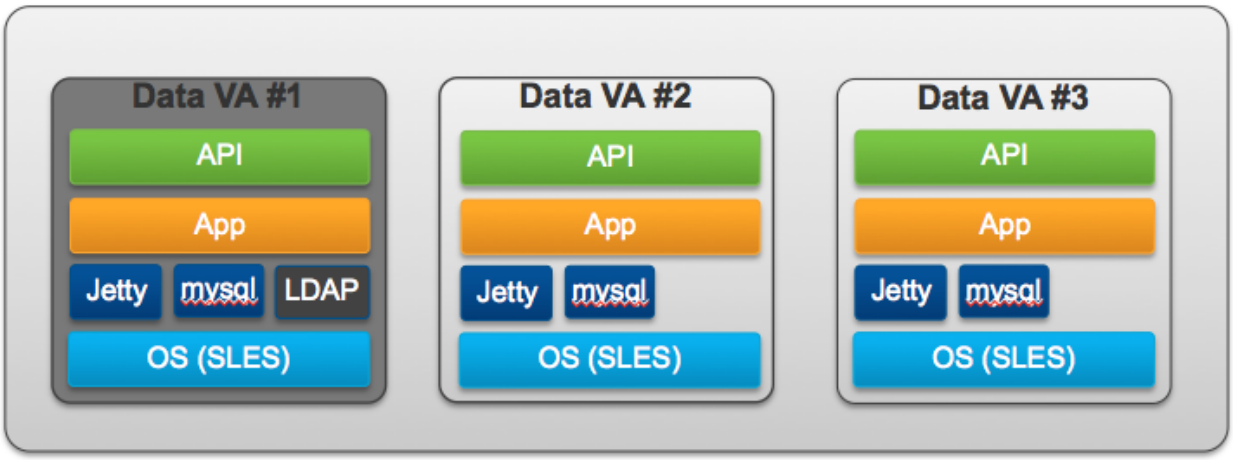


For detailed information on installation and configuration refer to the official [VMware Horizon Workspace product documentation].

Horizon Files is a spin-off from the Zimbra source code, in particular the store component hosting the user data, and good part of the nginx layout and configuration. The code was forked on 2012/5/21 at 19:01:30 PDT.
The Files VA holds different services and components that makes up the capabilities to serve a web based user interface to the users as well as handling files, sync operations, revisions and potentially high fidelity preview features of documents.
Web UI components are powered by Jetty

Some of the components - specifically OpenLDAP - is unique to the first Data VA which is deployed as part of the initial vApp. The LDAP service is not running on subsequently added Data VAs meaning that all user related queries goes through the first - primary - Data VA.
A subset of the information synchronized via Connector VAs from Active Directory (AD) kept at the Service VAs database is kept in the OpenLDAP store and then referenced by the

different Data VAs added to the solution afterwards.
From an architectural point of view this means that the first Data VA is a potential Single Point of Failure (SPF) and should be treated as a critical component from a backup/restore perspective.
It is recommended that no user data is kept on the first Data VA by excluding it from the Class Of Service (COS) to avoid making potential restore operations complex and having direct impact on user stored data. Additionally no external stores - NFS - should be mapped against the first Data VA.



As seen on the picture above the first Data VA is the only node carrying the OpenLDAP service.

As of version 1.5 of Horizon Workspace adding additional Data VAs is no longer relying on snapshots on the primary Data VA.
A data-va-template is introduced which is a full clone of the initially deployed Data VA from where subsequently added Data VAs are based.
This removes any reliance on snapshots and improves disk performance by not having to traverse a snapshot chain.
The process is the same for adding additional VAs and is outlined in the official VMware Horizon Workspace product documentation where usage of the hznAdminTool is shown.
NOTE:
If upgrading from Workspace 1.0 to 1.5 the data-vatemplate needs to be added manually.
See *Creating a New datava-template Virtual Machine* in the *Installing and Configuring Horizon Workspace Guide.*
If using Libre as preview-server option this also needs to be updated. This process is detailed in the official Horizon Workspace guidance.

Technical product purpose

The purpose of this guide is to:

- Explain the architecture of Horizon Workspace Files
- Explain the inner workings of the Files components
- To offer best practices to support design decisions
- Show typical operation and maintenance tasks

This guide is not intended for guiding in the operational aspects of Horizon Workspace and does not cover other aspects then Files in details.

Writing conventions

LDAP attributes

Every attribute starting with `hzndata...` , then each following unspaced word is capitalized in the initial (e.g. `hzndataDocumentMaxRevisions`) is an LDAP-hosted attribute, and all attributes we'll refer to can be found in `/opt/zimbra/conf/attrs/zimbra-attrs.xml` with the full syntax and a brief description. See the user-data structure for further information and reference on the whole attributes list. For managing them, `zmprov` is used and you should refer to our official CLI Reference guide for more information.

Localconfig attributes

If the attribute name is all lowercases and uses underscores, it's a localconfig value and it's stored in `/opt/zimbra/conf/localconfig.xml`. e.g.

```
oo_working_dir = ${zimbra_tmp_directory}
```

Localconfig values have meaning and scope just for the local data node it's set on. They are changed via the `zmlocalconfig` command, refer to our official CLI Reference guide for more information.

Components and inner workings

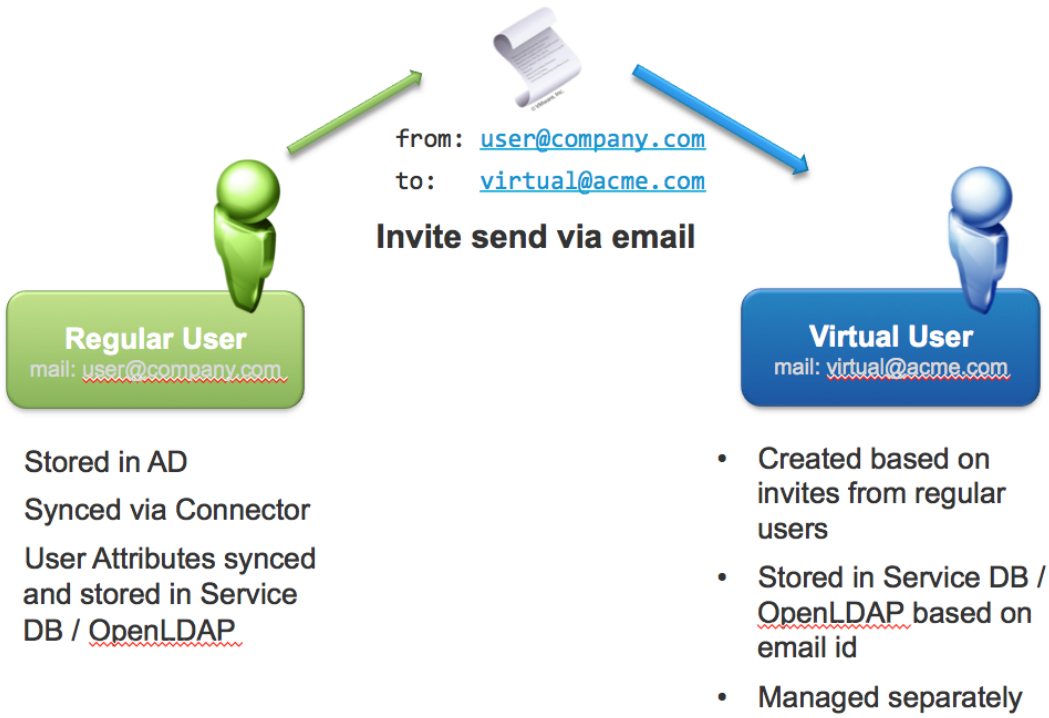
This section detailes the major components of Horizon Files as well as available user types, clients, sharing and collaboration features.

User account types

In Horizon Workspace 2 kind of user accounts exists:

- Regular User
- Virtual User

A regular user is an account that originates from Active Directory (AD) initially synced via connector-VAs and populated with selected attributes from AD.
A virtual user is an account external from the corporate AD and is created based on a regular user requesting sharing and collaboration with such a user.



The picture above details the account types, their creation process and attributes.
Virtual users undergoes the same logging level and audit as a regular user would and audit reports and logging contains information on virtual users as they would regular users activity.

Class Of Service - COS

A neat way of grouping certain classes of users under the same configuration settings in one shot is done through a concept called Class Of Service. It's a concept similar to groups in AD, or tagging in others, and it's aimed to avoid redundant configuration steps for similar users. Horizon Files greatly relies on this feature, as we can rapidly configure our user base

with the minimum effort. Examples of COSes might be:

- Gold, Silver, Bronze
- CXO, Sales, Marketing

And so on. Further on will be made an example based on the above classes to cover every combination we might face.

User accounts definition and configuration

User accounts in Horizon Workspace are defined in multiple locations:

- Active Directory (authentication, identity management, roles)
- service-va DB (policies, entitlements, etc.)
- data-va OpenLDAP (Files-related settings)

We will focus on the data-va portion in this section. Every Horizon Files entitled account has an entry in OpenLDAP, as it's the core location where Horizon Files-related settings are. As every data-va will point to the LDAP service found on the first data-va, set apart some caching issues if not planned in advance -see next-, any change will be then seen from any data-va, regardless of where the setting was modified via CLI or GUI.

LDAP data structure and layout

Here is a LDIF extract of a typical user:

```
dn: uid=user1,ou=people,dc=example,dc=com
sn: user1
givenName: user1
displayName: user1 user1
objectClass: inetOrgPerson
objectClass: hzndataAccount
hzndataId: 9
hzndataCreateTimestamp: 20130711105033Z
hzndataAccountStatus: active
hzndataTenantId: 4eeb7494-9b9d-492d-b1fc-ed8026227b49
hzndataCOSId: b700d7b9-339c-47af-9128-c456c1c45c91
hzndataHost: hdata.cork.zimbralab.com
hzndataPrimaryName: user1@example.com
mail: user1@example.com
cn: user1 user1
uid: user1
```

```
structuralObjectClass: inetOrgPerson
entryUUID: 74d2bca6-7e63-1032-8d14-79f1a35e820c
creatorsName: uid=hzndata,cn=admins,cn=hzndata
createTimestamp: 20130711105033Z
entryCSN: 20130711105033.735986Z#000000#000#000000
modifiersName: uid=hzndata,cn=admins,cn=hzndata
modifyTimestamp: 20130711105033Z
```

This is just the dump for a single user. Every account has a *hzndataId* attribute, unique throughout the Horizon installation, and stored in the service-va DB as well. The email address will be used as a unique identifier as well, this is mostly due to a source legacy reason, as this comes from Zimbra, where email addresses were prominent in its usage. That is why -when provisioning from AD- the email address attribute is mandatory, and users can't have the same email address.

To get an idea on what's contained in OpenLDAP, as the *zimbra* user on the first data-va, the following will produce the dump for the whole directory:

```
/opt/zimbra/libexec/zmslapcat /tmp; less /tmp/ldap.bak
```

Attributes inheritance

There are many more settings associated with an account, potentially. As mentioned, all of them are defined in a file on any data-va, that is `/opt/zimbra/conf/attrs/zimbra-attrs.xml`. Most of them are not shown in the user LDIF dump, since they are inherited from the COS pointed by `hzndataCOSId` if missing, when inquired. In fact, the inheritance relationship here is that account-level setting will override the COS settings. By default, the COS shipped as factory default and assigned to every new user is called default. Any user can have only one COS assigned at any one time, and COSes cannot be inheriting from other COSes.

This means that -reusing the example COSes mentioned earlier-, we'll need 9 COSes as a result of a matrix multiplication to cover every combination of classes available to be assigned by admins to the users.:

- Gold-CXO, Gold-Sales, Gold-Marketing
- Silver-CXO, Silver-Sales, Silver-Marketing
- Bronze-CXO, Bronze-Sales, Bronze-Marketing

Should we set something specifically for a single account only as an exception:

```
zimbra@hdata:~> zmprov ga user1@example.com hzndataDocumentMaxRevisions
# name user1@example.com
```

```
hzndataDocumentMaxRevisions: 20

zimbra@hdata:~> zmprov ma user1@example.com hzndataDocumentMaxRevisions 40
zimbra@hdata:~> zmprov ga user1@example.com hzndataDocumentMaxRevisions
# name user1@example.com
hzndataDocumentMaxRevisions: 40
```

We would now see that the account entity in LDAP has a further setting stored there, that is:

```
dn: uid=user1,ou=people,dc=example,dc=com
sn: user1
[...truncated...]
createTimestamp: 20130711105033Z
hzndataDocumentMaxRevisions: 40
entryCSN: 20130722104305.604450Z#000000#000#000000
modifiersName: uid=hzndata,cn=admins,cn=hzndata
modifyTimestamp: 20130722104305Z
```

Attributes definition

All the attributes that can be defined not only on accounts, but also on other scopes, such as COS, servers, etc. are to be found in `/opt/zimbra/conf/attrs/zimbra-attrs.xml`.Its contents are not just for documentation purposes, but they are parsed at application boot, therefore it's mandatory that this file is accessed without committing any change, at all. Every attribute has a precise definition, scoping, cardinality, inheriting definition and a short description as well.

For example, speaking of `hzndataAccountStatus`:

```
<attr id="481" name="hzndataDocumentMaxRevisions" type="integer" min="0"
cardinality="single" optionalIn="account,cos" flags="accountInherited"
since="1.0.0">
  <defaultCOSValue>20</defaultCOSValue>
  <desc>maximum number of revisions to keep for documents. 0 means
unlimited.</desc>
</attr>
```

A thorough description of such attributes can be found at the beginning of `/opt/zimbra/conf/attrs/zimbra-attrs.xml`. But just to give a quick overview as we speak, in our specific case we can see that:

- the id reference for this attribute is `481` (internal only)
- the type of value that can be assigned is an integer (`type="integer"`)
- the minimum value assignable is 0 (`type="integer"`)
- this setting can have only one value set at a time (`cardinality="single"`)
- that it's not mandatory and can be set on both at the COS and account level (`optionalIn="account,cos"`)
- this setting was introduced since the first GA version (`since="1.0.0"`)
- that it gets inherited if missing from the account level definition (`flags="accountInherited"`)
- if not set, the factory default is set, 20 in our case (`defaultCOSValue` tag)
- last but not least, a brief description of the setting (`desc` tag)

The schema (on the first data-va, in */opt/zimbra/openldap/etc/openldap/schema/*) defines constraints for any setting to be stored in Data, in any scope.
At the moment we don't have Multi-Master or Master-Slave Replication of LDAP. The original data-va (that is, the one installed when initially deploying the OVA) will be hosting such service. Refer to the Architecture&Scale section for further information on our current best practices about this.

User data structure breakdown

Every Data node is responsible for its own users hosted locally which means that a user is tied to a specific Data VA node.
Each time we add a file in a client, the server takes care of the transfer of the file by:

- updating the metadata in MySQL
- indexing to allow full-text searches
- and then placing the file on the FS where the current active zmvolume is pointing to.

This whole process is transactional. Any fail within the above three steps will revert back and return an error, this to avoid any inconsistency whatsoever.

MySQL

MySQL is in charge of storing the metadata. But what exactly do we mean with "Metadata"? Metadata is everything that is either "added value" for the file we store (permissions, folder

organizations, system flags... anything that is not directly found on the file itself) and/or information regarding files that have to be accessed as quickly as possible, avoiding to hit the disk subsystem. For instance, when opening a folder, we are actually inquiring MySQL only, that includes retrieving the folder listing (names, sizes, dates, etc) and the folder structure itself. Only when we click to preview or open a file we actually hit the disk subsystem.

Among things that we store in MySQL:

- Filenames
- Folder structures
- Flags
- Timestamp
- File Size
- Volumes
- Shares definitions
- File digest
- File path details

The most important MySQL InnoDB tables are:

- `zimbra.mailbox`: Main table listing the users hosted on the node.
- `mboxgroup`*`XX`*`.mail_item`: Main table containing any files metadata the user owns. We spread user DB creation when provisioning users for performance/recovery reasons, algorithm is Round Robin, and `1<N<100` by factory default
- `mboxgroup`*`XX`*`.revision`: Revision table, keeps track of the file versioning.
- `mboxgroup`*`XX`*`.*_dumpster`: Recovery tables for retrieving files when they have deleted by the user (default recovery time is one month).
- `zimbra.volume`: Definition of each Data volume configured on the node
- `zimbra.current_volumes`: The currently pointed and active volumes on the system.

Getting access to the mysql client is as easy as connecting via SSH to a data-va node, then switching to the root user and finally the zimbra user, which then can just run the `mysql` command:

```
root@zcs7-multi-ldap:~# ssh sshuser@hdata

Welcome to SUSE Linux Enterprise Server 11 SP2 for VMware  (x86_64) - Kernel \r (\l).


sshuser@hdata's password:
Last login: Tue Aug  6 16:15:59 2013 from zcs7-multi-ldap.cork.zimbralab.com
sshuser@hdata:~> su -
Password:
hdata:~ # su - zimbra
zimbra@hdata:~> mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
```

```
Your MySQL connection id is 35203
Server version: 5.5.27-log Source distribution

Copyright (c) 2000, 2011, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

(Refer to the MySQL official documentation to see how to inspect the databases found on each data-va, as well as their respective tables and columns)

The main rule of thumb here is that, if it doesn't exist in MySQL, then the file doesn't exist at all within the application flow. Not even if the file is present on the FS would change the fact that Horizon Workspace will not possibly be able to find it and/or retrieve it. In fact, FS access to read the file is done by inquiring its full path, that is made up of different column values within the same row, all coming from the `mail_item` table. See the File Storage chapter for further information.

With MySQL, we can get hold of any information on any file handled by Horizon Files. Although a good deal of reports and statistics are provided within the admin dashboard, not every possible inquiry can be answered by the most common reports grouped in our GUI. Therefore inquirying MySQL is the ultimate way of having control on every possible bit of information regarding our users' data layout.

The MySQL configuration file my.cnf can be found in `/opt/zimbra/conf/my.cnf` and has pretty much everything defined there. For further information, the tuning chapter will have some more detail on this.

MySQL examples

The following is just the tip of the iceberg, but it's meant to be a proof of concept. As MySQL already has quite a wealth of online documentation and active forums, don't expect the following examples to be the state of the art of SQL queries, rather a starting point to get the hang of it and creating new ones.

Warning: MySQL databases should be never modified manually. This said, even if the following examples are just SELECTs, they should be run at night time only and tested first when it comes to performance hits (as they are obviously custom queries not being validated by QA).

Note: as it's safer to not rely on DB temporary tables via chunky queries, not to mention keeping the overall SQL query complexity, the outer loop is usually done in bash.

- To list the biggest folders in a data-va

```
export min=500;
for i in `seq 1 100`; do
echo "mboxgroup$i";
mysql -s -e "select count(mboxgroup$i.mail_item.id),folder_id,comment from
mboxgroup$i.mail_item left join zimbra.mailbox on
zimbra.mailbox.id=mboxgroup$i.mail_item.mailbox_id group by
folder_id,mailbox_id having count(mboxgroup$i.mail_item.id)> $min ;";
[[ $? -eq 1 ]] && break;
done |egrep '^[-0-9].*' | sort -n
```

- To find the most revisioned files (useful to monitor HW-3446):

```
export min=5;
for i in `seq 1 100`; do
echo "mboxgroup$i";
mysql -s -e "select count(mboxgroup$i.revision.version),mailbox_id,comment from
mboxgroup$i.revision left join zimbra.mailbox on mailbox_id having
count(mboxgroup$i.revision.version)>$min order by
count(mboxgroup$i.revision.version) desc;";
[[ $? -eq 1 ]] && break;
done |egrep '^[-0-9].*' | sort -n
```

The tables we'll usually need to play with are the following

Indexing

Indexing allows us to find virtually everything in a matter of seconds, thus greatly enhancing the usability of our data. An indexing engine breaks down documents and files into a number of terms. With this info, it builds and increments the user index file as an Inverted Index data structure, which might be loosely resembling -in its scope and usage scenario- to the better known *hashtable* construct. *Spotlight* for Mac OS X is an equivalent in terms of functionalities for a desktop computer.

Indexes in Horizon data-vas are produced by Lucene, a 3rd party component integrated by the *mailboxd* component as an API library. Every file being handled by data-va will be eventually prepared and then served for indexing by *convertd*, a service handled on top of Apache that runs on port 7047. Every time we change (modify) the files, mailboxd will summon Lucene to process the file and update the index. Every index lives on the same node the user is, as well as the rest of its data (metadata and files). This is for performance reasons. The path to the user index files follows pretty much the same reasoning shown in

the slide about the mailbox structure.

What's being returned by searches against the indexes is actually just metadata column values, that then get listed as the search results set. When then the user wants to inspect such entries, by clicking on them the real fetching phase of the file gets triggered, thus enabling IO access to the filesystem.
When we have shares in place, when we search in our accounts where we have folders mapped from other users' accounts, the search inquiries other nodes where the sharer's index resides, and return the information back to the searcher through the home data-va node of the inquirer, then back to us through the gateway-va.

Index might get corrupted, or being outdated due to operational issues. If so, a reindexing might be triggered via command line as well. (type `zmprov rim` as the *zimbra* user for the full usage). This is quite resource intensive, and it'll start from scratch by parsing every file the user has. User reindexing is recommended off business hours, and even more so if this involves multiple users in need of re-indexing.

The search syntax is very powerful, supporting the most common operators plus some others, and it resembles a lot what we can already do in a Google Search, e.g.

```
mbox user1@example.com> s 'in:/Briefcase/myFolder'
num: 1, more: false

      Id  Type   From                 Subject
Date
   -----  ----   --------------------
------------------------------------------------  --------------
1. 9:267   doc   user1@example.com     test123 - Copy.txt
06/10/13 12:01

mbox user1@example.com> s 'in:/Briefcase/myFolder -Copy'
num: 0, more: false
```

(Note: If you are wondering why the Id column is formatted so that you have 2 numbers separated by a colon, that is because the format is: "<serviceDBuserId>:<objectId>")

```
mbox user1@example.com> s 'get*'
num: 2, more: false

      Id  Type   From                 Subject
Date
   -----  ----   --------------------
------------------------------------------------  --------------
1. 9:257   doc   user1@example.com     974081.htm
07/12/13 14:58
2. 9:260   doc   user1@example.com     Getting Started.pdf
```

```
01/11/13 16:13
```

If we then try to confront the above with the MySQL contents:

```
mysql> select * from mail_item where id=257 and mailbox_id=1\G
*************************** 1. row ***************************
  mailbox_id: 1
          id: 257
        type: 8
   parent_id: NULL
   folder_id: 16
    index_id: 0
        date: 1357895217
        size: 87803
     locator: 1
 blob_digest: fKvTXmbL4mejnHhsy3YVLHcgGplMf0j,Y3j,MK8s8PI=
      unread: NULL
       flags: 0
        tags: 0
   tag_names: NULL
      sender: user1@example.com
  recipients: NULL
     subject: 974081.htm
        name: 974081.htm
    metadata:
d2:cr17:user1@example.com2:ct9:text/html3:dee5:false5:mdveri3e1:vi10ee
mod_metadata: 3
 change_date: 1373540705
 mod_content: 3
        uuid: 453807dd-7e8c-4e04-b90f-d74b49bbce2c
1 row in set (0.00 sec)


zimbra@hdata:~> ls -l /opt/zimbra/store/0/1/msg/0/257-*
-rw-r----- 1 zimbra zimbra 87803 Jul 11 12:05
/opt/zimbra/store/0/1/msg/0/257-3.msg

mysql> select * from mail_item where id=257 and mailbox_id=1\G
*************************** 1. row ***************************
  mailbox_id: 1
          id: 257
        type: 8
   parent_id: NULL
   folder_id: 16
    index_id: 0
        date: 1373637525
        size: 87804
     locator: 1
 blob_digest: WC++GtyO+1gcQZi8sc7RHdGexPfUqu2PR5XYRwZJa64=
      unread: NULL
```

```
       flags: 4096
        tags: 0
   tag_names: NULL
      sender: user1@example.com
  recipients: NULL
     subject: 974081.htm
        name: 974081.htm
    metadata:
d2:cr17:user1@example.com2:ct9:text/html3:dee5:false5:mdveri6e1:vi10e3:veri2ee
mod_metadata: 809
 change_date: 1373637495
 mod_content: 809
        uuid: 453807dd-7e8c-4e04-b90f-d74b49bbce2c
1 row in set (0.00 sec)

zimbra@hdata:~> ls -l /opt/zimbra/store/0/1/msg/0/257-*
-rw-r----- 1 zimbra zimbra 87803 Jul 11 12:05
/opt/zimbra/store/0/1/msg/0/257-3.msg
-rw-r----- 1 zimbra zimbra 87804 Jul 12 14:58
/opt/zimbra/store/0/1/msg/0/257-809.msg
```

A very good read on how Lucene indexing works can be found here:
http://www.ibm.com/developerworks/library/wa-lucene/

File Storage

Users files are finally stored as they are into the data-va where the user resides. This user store can be internal or external to the data-va, and currently we support either VMDKs (internal) or NFS v3+ (external) for such. NFS is recommended for production use as it allows to delegate backing up of the actual files (the biggest part of the data to be saved, as metadata takes up a fraction of the overall space an average file takes up) to typical SAN/NAS features such as volume snapshots.

Files are stored unencrypted and unmodified -that is, without modifications of any sort if not for the naming and permissions-, thus delegating deduplication to the storage layer, if such feature is provided by the storage vendor.

The internal storage location is initially mounted at `/opt/zimbra/store` which is based on the local VMDK shipping with the data-va and it's the only `primaryMessage` volume, therefore it's also active (`current:true` in zmvolume -l output). All files being added are added here in R/W. When another `primaryMessage` volume is added (NFS), there will be another volume created and listed, which will also become the new current/active one by default. The files present on the other volume will stay and will not move over though, and this old volume will become inactive (`current:false`) and accessed in R mode, just when a file needs to be retrieved/previewed/etc.

Currently, the only supported option to consolidate files onto a same volume is to move the accounts to another data-va with the zmvolume we want as destination to be set as current. The typical scenario is a customer leveraging initially on the internal storage only, but then NFS migration has been planned as it's better suited for production use and backup purposes. Refer to the *Moving Accounts* chapter for further information.

Speaking of naming of the files and their full path on the filesystem, it is determined by an algorithm requiring sourcing of various column values found in the metadata stored within the local MySQL instance. Using a colored overview of a typical file full path that can be found on a data-va node:

/opt/zimbra/store/0/1/msg/0/257-3.msg

Where:

/opt/zimbra/store found via `current_volumes` and `volume` in `zimbra` DB
0 right bitshift by 12 of 1
1 `mboxgroupXX.mail_item.mailbox_id`
0 right bitshift by 12 of 257
257 `mboxgroupXX.mail_item.id`
3 `mboxgroupXX.mail_item.mod_content`

In SQL terms, the query that might allow us to retrieve is the following:

```
zimbra@hdata:~> mysql -s -e 'connect mboxgroup1; select concat((select path from
zimbra.volume where id=(select message_volume_id from zimbra.current_volumes)),"/",
(mailbox_id >> 12), "/", mailbox_id, "/msg/", (id >> 12), "/", id, "-", mod_content,
".msg") as path from mail_item where mailbox_id="1" and id="259" limit 1;'
path
/opt/zimbra/store/0/1/msg/0/259-5.msg
```

Which then can be verified by:

```
zimbra@hdata:~> ls -l /opt/zimbra/store/0/1/msg/0/259-5.msg
-rw-r----- 1 zimbra zimbra 0 Jul 26 10:30 /opt/zimbra/store/0/1/msg/0/259-5.msg
zimbra@hdata:~> file /opt/zimbra/store/0/1/msg/0/259-5.msg
/opt/zimbra/store/0/1/msg/0/259-5.msg: PDF document, version 1.6
```

NOTE: The above SQL query requires notion of the following:

- The mailbox ID of the user (`1` in our example). This can be retrieved by doing the following on the data-va node hosting the user:

  ```
  zimbra@hdata:~> zmprov gmi user1@example.com
  mailboxId: 1
  quotaUsed: 30121319
  ```

- Which `mboxgroupXX DB` the user belongs to. This is actually determined by a simple algorithm, which requires a modulo operation between the mailbox ID and number `100`. In our case:
  ```
  zimbra@hdata:~> expr 1 % 100
  1
  ```

- The object ID (`259` in our case), which can be found in many ways, one of which by means of the `zmmailbox` subcommands such as `search (s)` for example. This subcommand is not completely documented in the CLI guide and in the following examples, the object IDs `260, 331, 309` and `330` can be found:

  ```
  mbox user1@example.com> s Getting
  num: 1, more: false

         Id  Type   From                 Subject
  Date
  ------------------------------------------------  --------------
  1. 10:260   doc   user1@example.com     Getting Started.pdf
  01/11/13 16:13
  ```

  Or, to see a whole folder:

  ```
  mbox user1@example.com> s 'in:"/Briefcase/Stuff/Microsoft Office/Office14"'
  num: 25, more: false

          Id  Type   From                 Subject
  ```

```
Date
-------------------------------------------------  --------------
 1. 10:331   doc   user1@example.com     VVIEWER.DLL
12/07/12 12:46
 2. 10:309   doc   user1@example.com     GROOVEEX.DLL
08/16/12 06:43
 3. 10:330   doc   user1@example.com     VVIEWDWG.DLL
04/26/12 08:17
[...]
```

Refer to the indexing section for an explanation on this output.

We store a full file copy on each change made by the users, and by factory default every installation has a limit on the number of revisions of 20 for each file. Every revision committed over the limit will actually trigger the deletion of the oldest one stored.

Main services layout

To recap, a list of the services and components that is running in the Data VAs:

- **mailboxd**: It's the main java code handling threads and listeners for the admin and user interface. It runs on top of Jetty, and handles the indexing actions when new messages come in.
- **zmconfigd**: a watch-dog process running every minute to parse any changes in the node configuration that require certain services reload/restart.
- **convertd**: an apache-backed attachment conversion/expansion service
- **mysql**: in charge of storing the metadata for each single file owned by users, as well as its revision history and folder structure.
- **ldap**: an OpenLDAP instance in charge of storing the configuration for:
    - Data global Configuration
    - Each specific Data node config
    - Accounts
    - Class Of Service (COS)
    - ...
- **Scripts/Utilities**: most of these are still named after the Zimbra ones (zmprov, zmdiaglog, zmmailbox, zmboxmove, zmcontrol, ...), providing glue to many of the workflows and administration tasks. Renaming/rebranding efforts are on their way to make the naming more consistent with the Horizon trademark.

Admin Operations

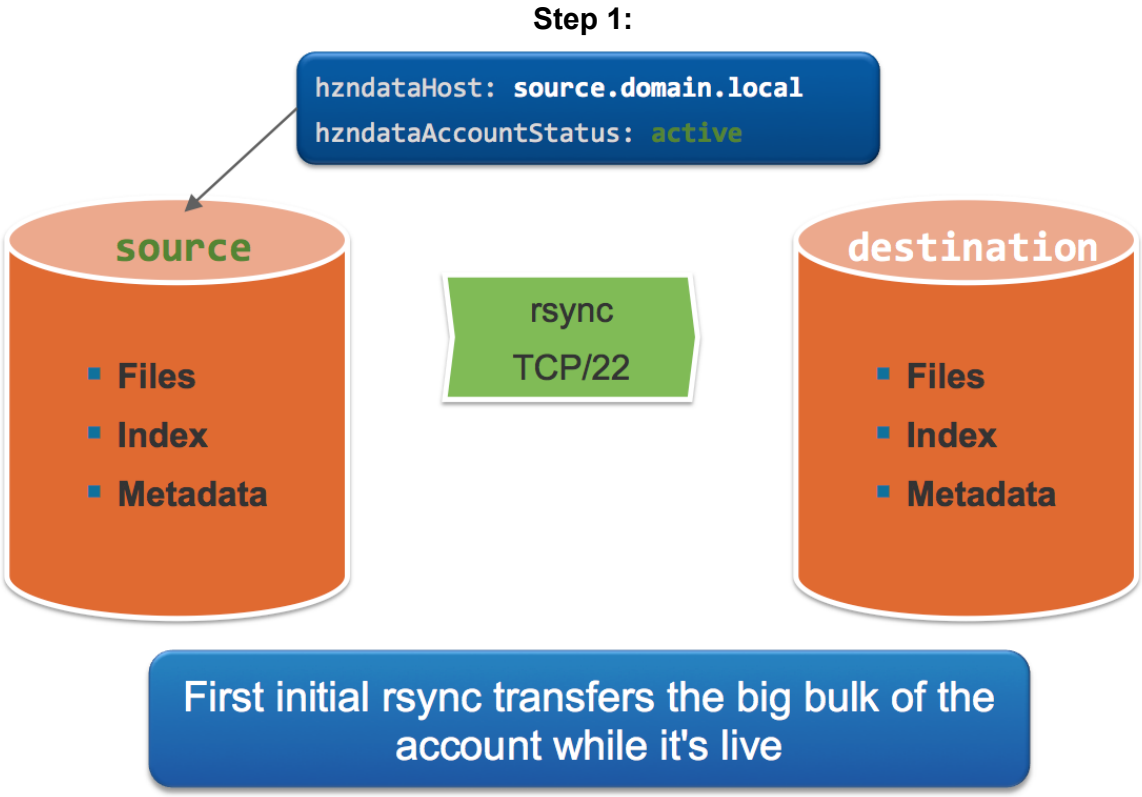
Moving accounts

When a user is moved, everything (files, index, mysql) gets transferred via rsync through SSH, and on success ldap gets updated in the `hzndataHost` value. The communication happens by means of passwordless authentication through the `/opt/zimbra/.ssh/zimbra_identity.pub` files being installed on each data-va node. The move requires very little downtime (statistically never over 60s at worst).

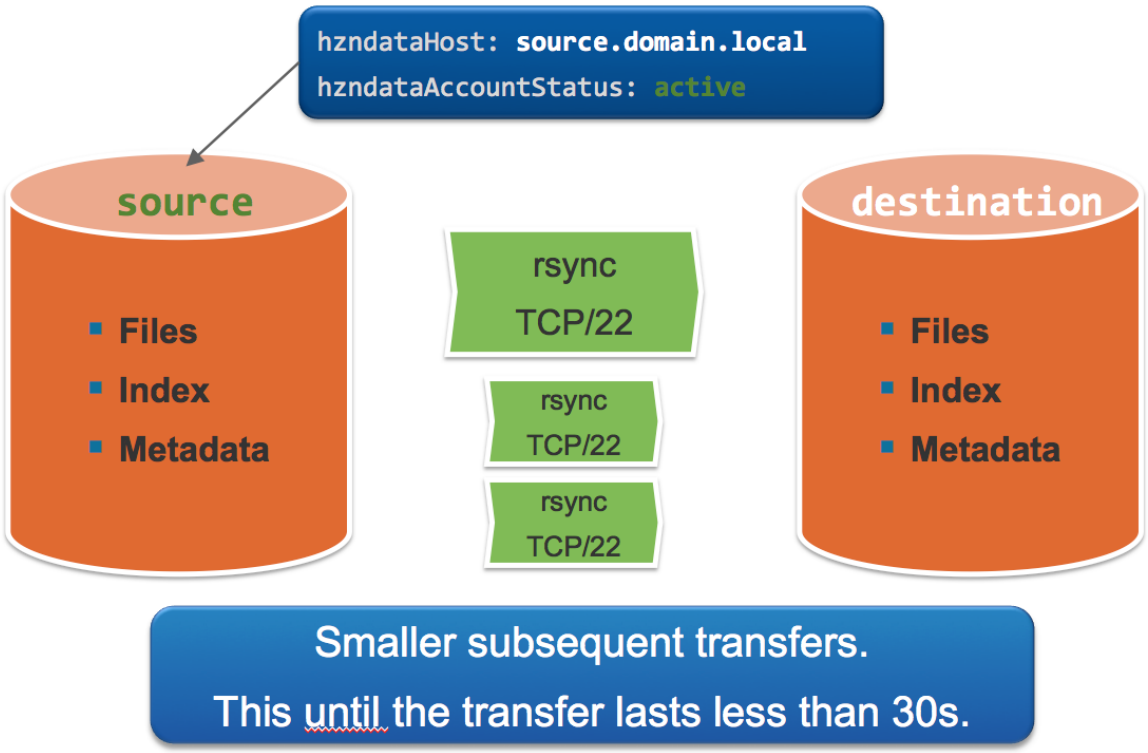
Moving accounts is usually needed for three reasons:

- Node deletions
- Horizontal scaling
- Files consolidation

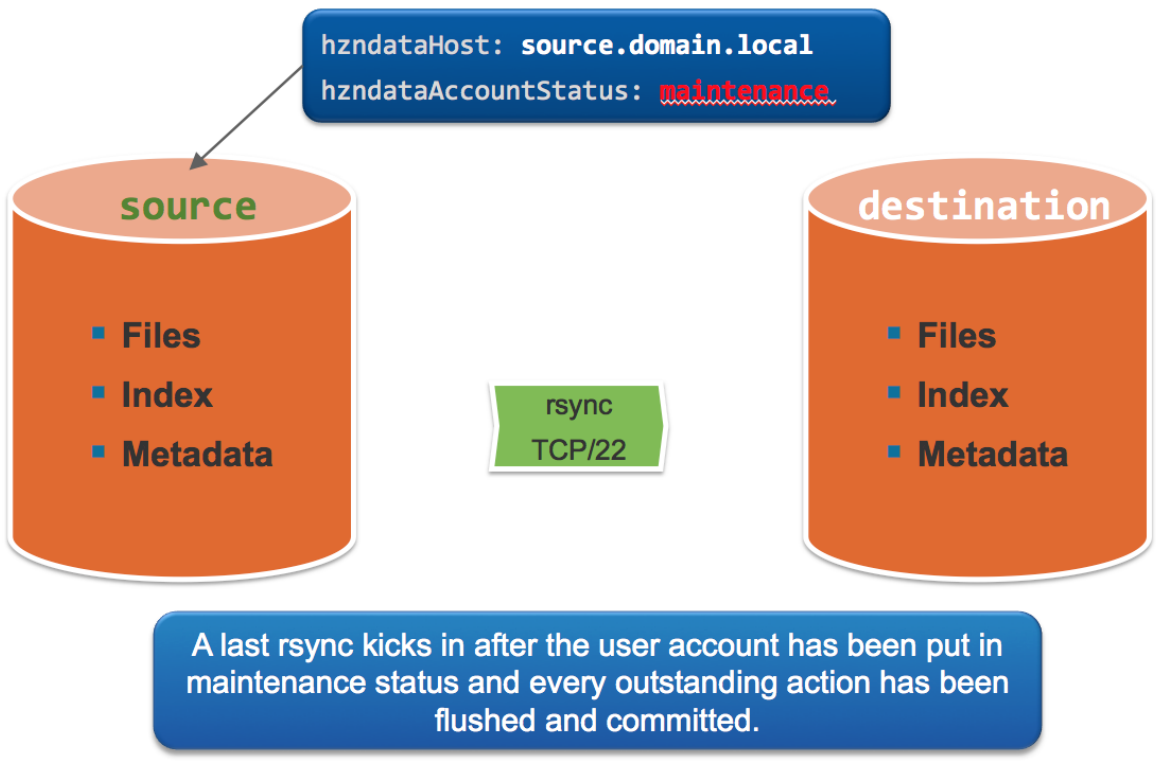
Graphically it might be summarized as follows:



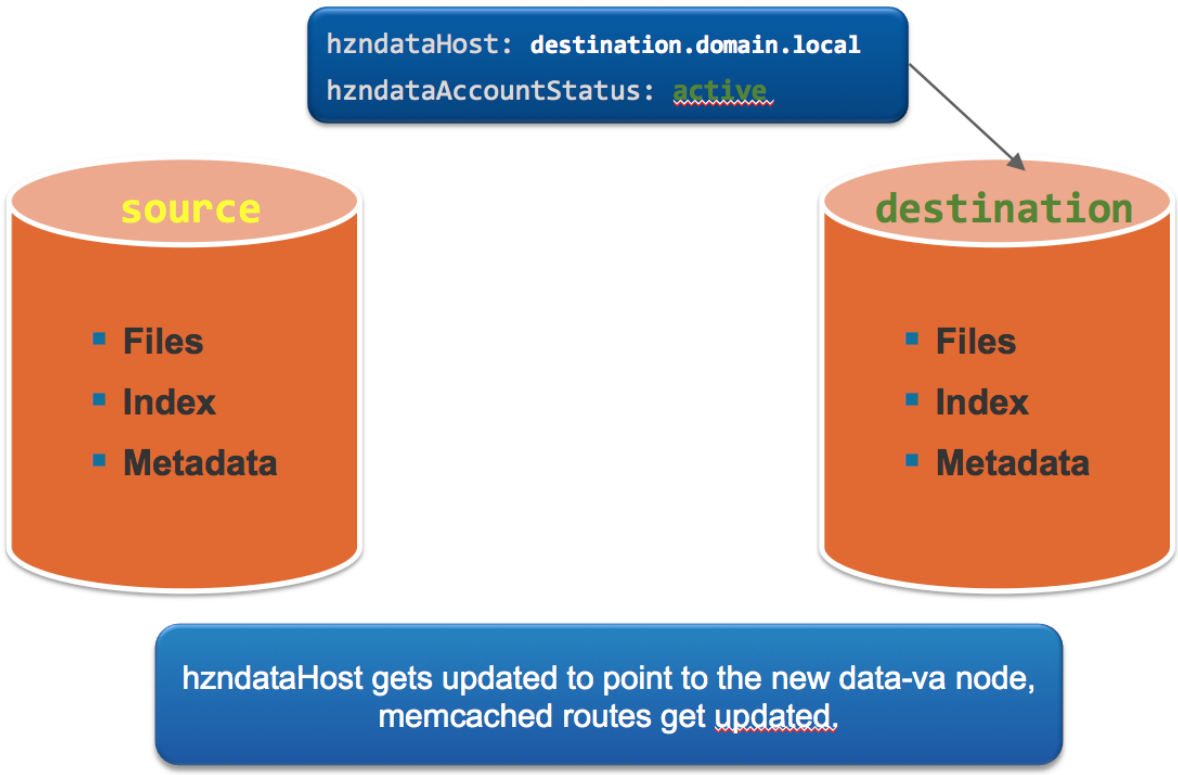
Step 2:


hzndataHost: source.domain.local
hzndataAccountStatus: active
source
Files
Index
Metadata
rsync
TCP/22
rsync
TCP/22
rsync
TCP/22
destination
Files
Index
Metadata
Smaller subsequent transfers.
This until the transfer lasts less than 30s.

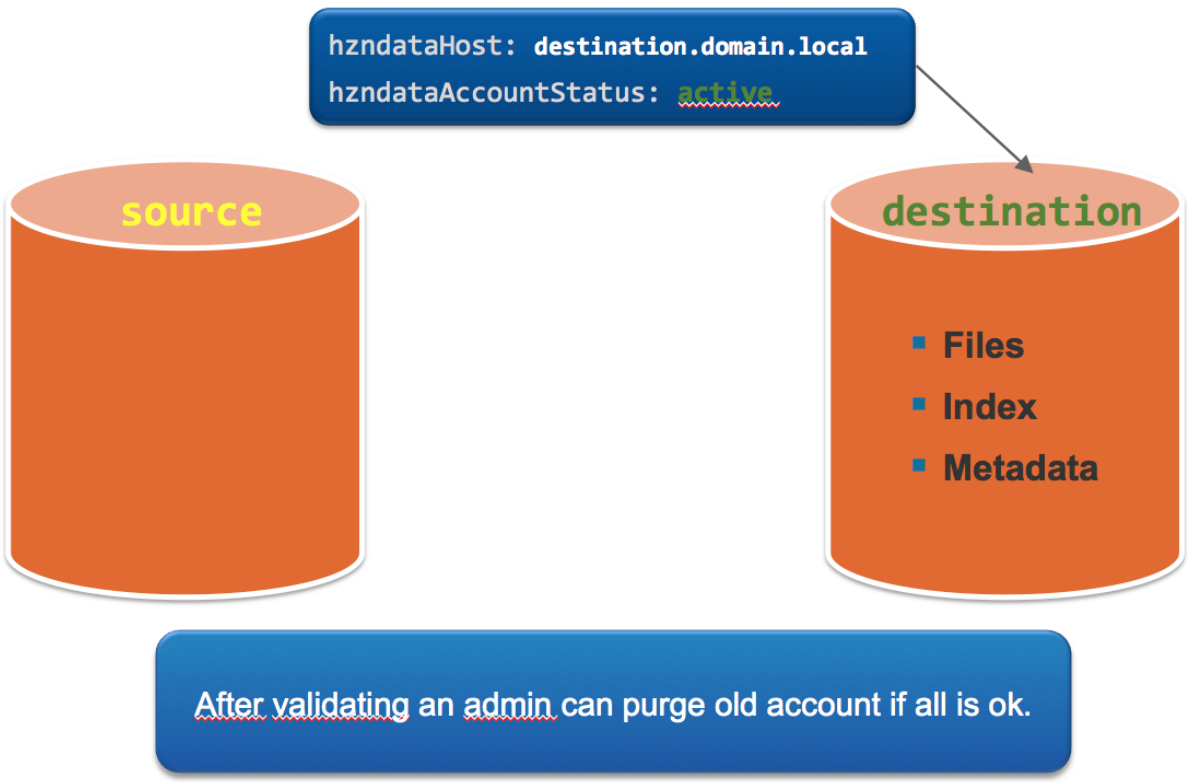
Step 3:


hzndataHost: source.domain.local
hzndataAccountStatus: maintenance
source
Files
Index
Metadata
rsync
TCP/22
destination
Files
Index
Metadata
A last rsync kicks in after the user account has been put in maintenance status and every outstanding action has been flushed and committed.

Step 4:


hzndataHost: destination.domain.local
hzndataAccountStatus: active
source
Files
Index
Metadata
destination
Files
Index
Metadata
hzndataHost gets updated to point to the new data-va node, memcached routes get updated.

Step 5:


hzndataHost: destination.domain.local
hzndataAccountStatus: active
source
destination
Files
Index
Metadata
After validating an admin can purge old account if all is ok.

As an example, an user being moved from data-va-1.example.com to data-va-2.example.com will go through this sequence:

1. User data is being copied without interruptions for the first time over rsync: index, files, metadata. This will be the first initial bigger transfer.
2. Subsequently other rsyncs will happen to catch up with any other changes that came during step 1 due to user activities. During these transfers, user is unaware of the move (`hzndataAccountStatus: active`) and the active data-va node will still be the source server we are moving from (`hzndataHost: data-va-1.example.com`)
3. As soon as an rsync transfer in step 2 lasts less than 30s, user is then put into maintenance status (`hzndataAccountStatus: maintenance`) so that the user will not be able to synchronize further files, and its caches flushed on the source server to guarantee consistency.
4. The last rsync now kicks in to have all the data transferred consistently for the final import.
5. The import phase starts now, and the biggest chunk of time spent now is actually the MySQL import of the metadata in the DB on the destination server. Files and indexes don't require any application-level action, if not to be present and stored in the right pathes when the mailbox is reactivated.
6. The destination host now becomes the current one (`hzndataHost: data-va-2.example.com`) and the account is reenabled (`hzndataAccountStatus: active`)
7. Cached routes should be now purged by flushing the caches, by SSHing on any data-va and then:
   `wget -O - --quiet http://localhost:6035/flush`
8. When the move is successful, the admin can now purge the account on the source server, thus deleting the data residing there.

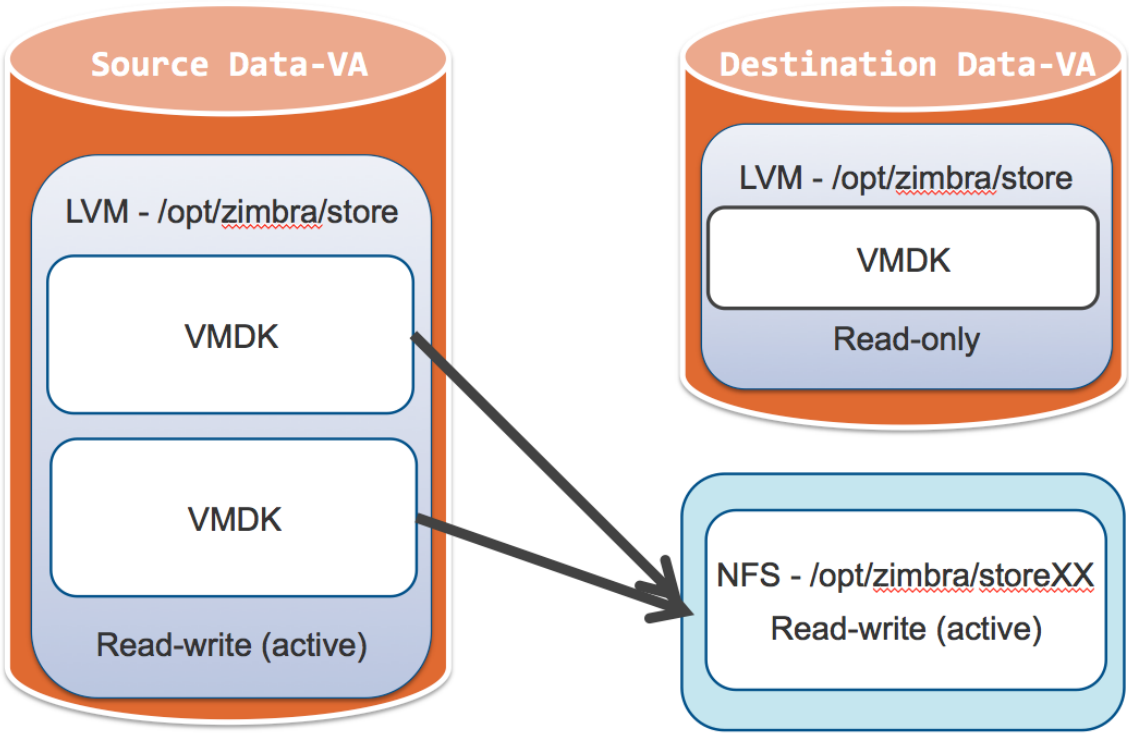
Moves can be either done in the foreground (--sync) or launched then backgrounded, but the above steps are the same, regardless of the method chosen. Multiple moves can be made in parallel, but we suggest to avoid going over 4 concurrent transfers at a time, as the transfers require some load due to the rsync activities in the background.

Files consolidation

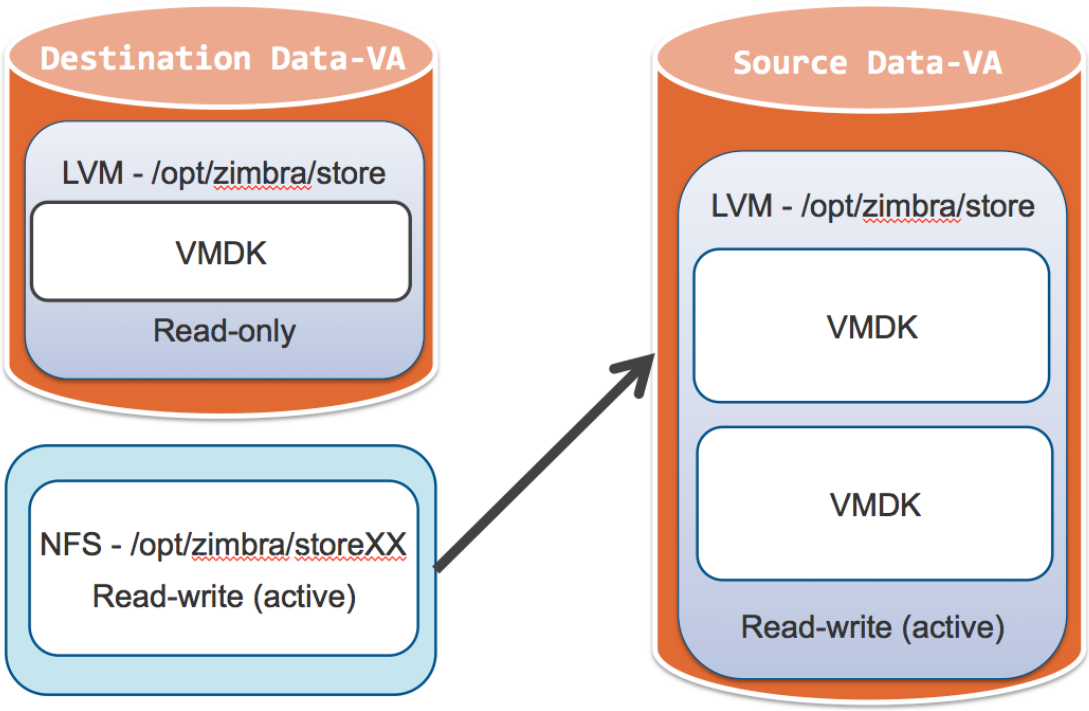
There are multiple reasons for wanting to move or consolidate user accounts across multiple Data VAs but the primary reasons tends to be adding NFS datastores, horizontal scaling or the fact that an existing store is running out of space.

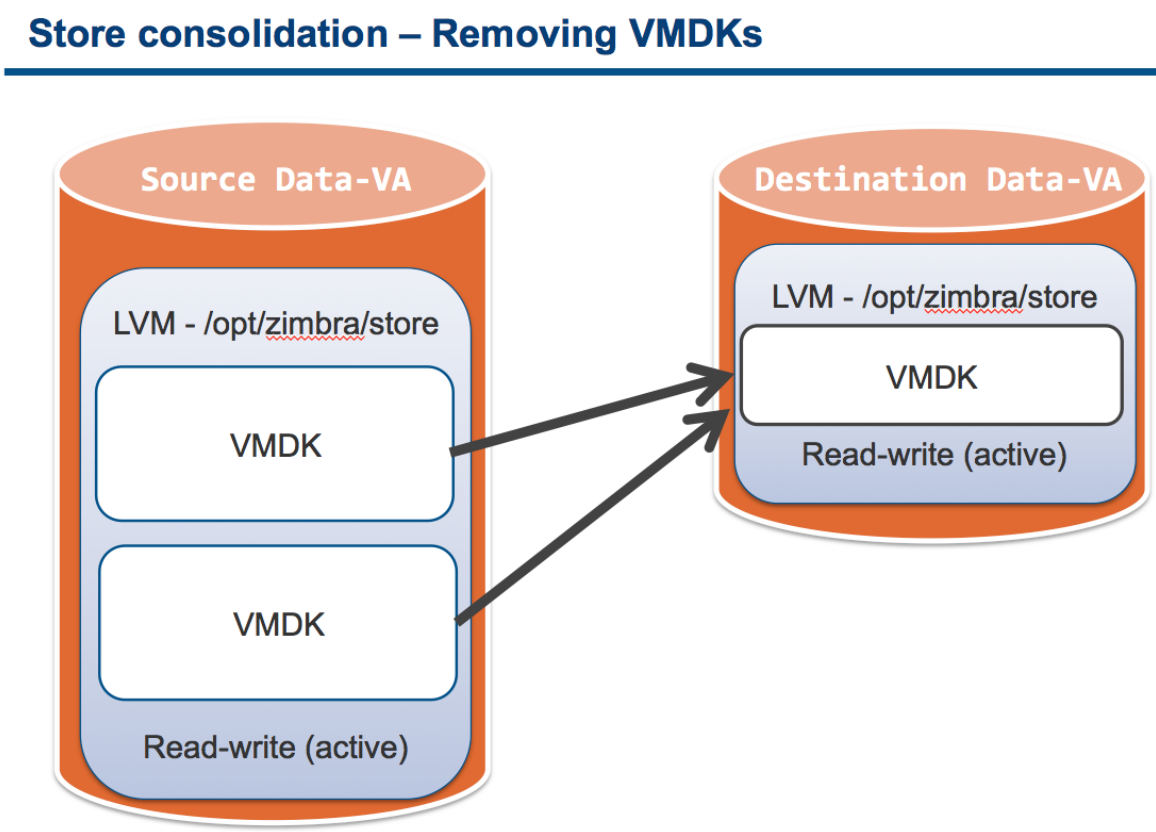
As mentioned earlier, to have files compacted from multiple volumes to just one, or transitioning from VMDKs to NFS or viceversa, the following diagrams will summarize how, based on the simple account migration described above.

Store consolidation – VMDK to NFS



Store consolidation – NFS to VMDK





Sharing

Horizon Workspace Files is a collaboration platform and therefore allows sharing both internally with users and groups and with people external to the corporate network as well. External users - known as *Virtual Users* in Horizon Workspace - is created when an internal user shares content with external users by entering that users email address.
External users have to be tracked by Horizon Workspace, hence OpenLDAP on the first Data VA node will be updated to contain such user accounts which means that - to be consistent - Horizon Workspace will create a user object for each Virtual User based on the email address. This information is stored in Horizon Workspace and is *not* created in the corporate Active Directory domain.

As part of the sharing features there are settings that the administrator control as part of the COS. These settings include but are not limited to:

- With whom data is shared (eg. “deny: @gmail.com” – black/white listing)
- What file types can be stored (file extension based)
- Changes can be made on a per user-basis (Quota changes eg.)

Any LDAP attribute in the format *hzndataVirtual** is describing such users when it comes to account/COS level configuration, and a brief description of them all can be found in `/opt/zimbra/conf/attrs/zimbra-attrs.xml` as usual.

To get the folder shares definitions from CLI, you can find the following by means of the *zmmailbox* command. An example is as follow:

```
zimbra@hdata:~> zmmailbox -z -m user1@example.com
mailbox: user1@example.com, size: 0 B, messages: 10, unread: 0
authenticated as OAuthClient_Data__OAuthClient__HGATEWAY@noreply.com
mbox user1@example.com> gaf
                                  Id  View      Unread   Msg Count  Path
----------------------------------  ----  ----------  ----------  ----------
                                10:1  unkn           0           0  /
                               10:16  docu           0          10  /Briefcase
                              10:300  docu           0           0  /Briefcase/TEST
                                10:2  mess           0           0  /Inbox
mbox user1@example.com> gf 10:300
{
     "color": "defaultColor",
     "contentSequence": 10245,
     "defaultView": "document",
     "effectivePermissions": "rwidxapfc",
     "flags": "i&",
     "grants": [
          {
               "id": "10",
               "name": "user1@example.com",
               "permissions": "rwidxapf",
               "type": "usr"
          },
          {
               "id": "8",
               "name": "test1@cork.zimbralab.com",
               "permissions": "r",
               "type": "usr"
          },
          {
               "id": "7",
               "name": "test2@cork.zimbralab.com",
               "permissions": "rwidx",
               "type": "usr"
          },
          {
               "id": "9",
               "name": "user2@example.com",
               "permissions": "rwidxa",
               "type": "usr"
```

```
            }
        ],
        "id": "10:300",
        "imapMODSEQ": 0,
        "isCheckedInUI": false,
        "isExcludedFromFreeBusy": false,
        "isSyncEnabled": false,
        "isSyncFolder": false,
        "isSystemFolder": false,
        "itemCount": 0,
        "name": "TEST",
        "parentId": "10:16",
        "path": "/Briefcase/TEST",
        "pathURLEncoded": "/Briefcase/TEST",
        "size": 0,
        "subFolders": [],
        "unreadCount": 0,
        "uuid": "e55ae239-587c-41c1-b03c-457725dcc1d3"
}
mbox user1@example.com>
```


The above shows the inner details of a folder named *TEST*, shared as:

- View only: test1@cork.zimbralab.com
- View/Edit: test2@cork.zimbralab.com
- View/Edit/Share: user2@example.com

Internal sharing between users works in a server-client model. Grossly, sharees access a share by means of a mountpoint, whereas all the sharer's data (index, files, metadata) stays on the sharer's node. Searches are delegated to the sharer's data-va node, as he's going to be the owner of the index files and metadata required.

Regarding quota, the owner of the share is charged for the space being used by other sharees, counting against his own quota.

Architecture and scale

Horizon Workspace is a platform designed for large scale implementations and all VAs that makes up Horizon Workspace can either scale vertically or horizontally.
VMware has published a Reference Architecture (RA) for Horizon Workspace 1.0 that uses a pre-defined workload profile to simulate user activity.

For more information on the RA see the VMware Horizon Workspace Reference Architecture PDF. The RA also includes information on network port requirements, external access and VA placement.

Horizon Workspace Files scaling

The approach for scaling Horizon Workspace Files is based around each Data VA serving up to 1.000 users. This allows for horizontal scaling where additional Data VAs can be added to Horizon Workspace as the solution grows.

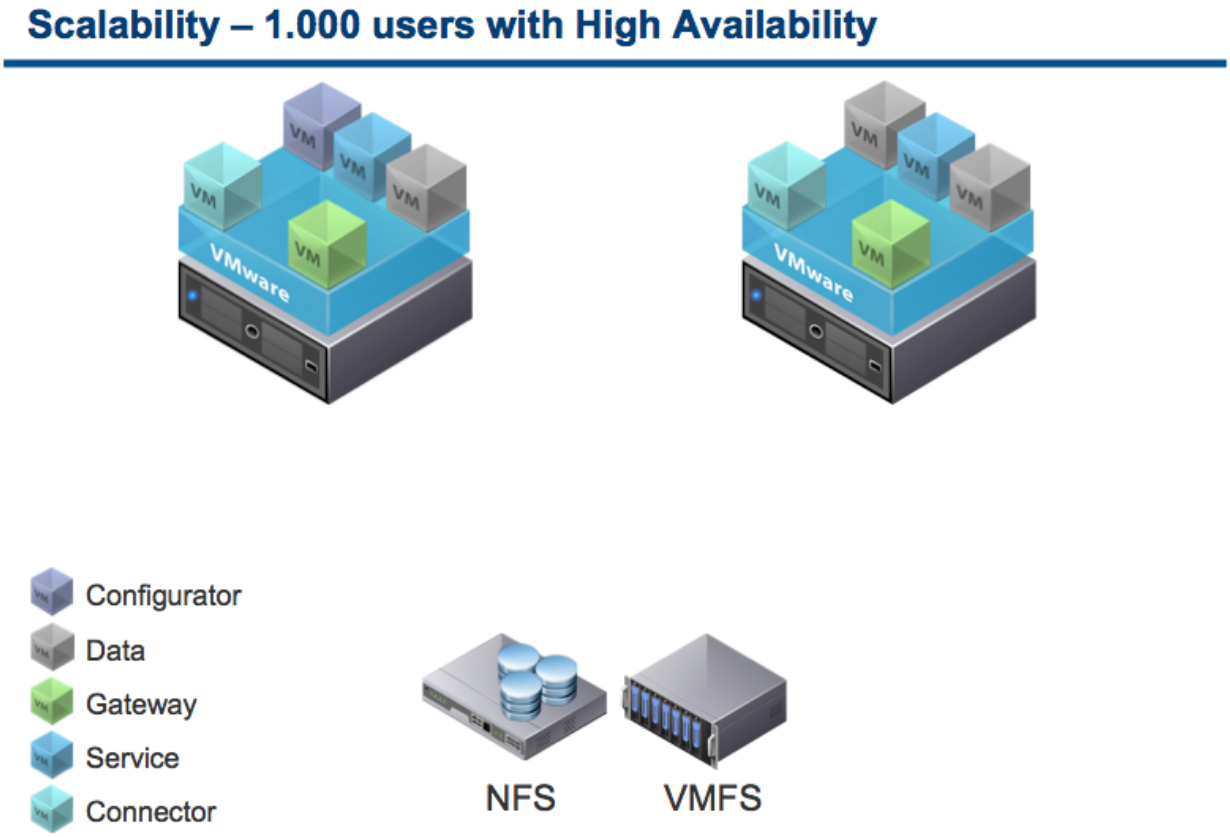
The best practice recommendation for scaling Horizon Workspace 1.x is outlined in the table below.

NOTE: The information outlined in this chapter is focused on Horizon Files for 1.000 users and does not go into sizing details for other VA types than Data VA.

	Number of vCPU's	**RAM (GB)**
Configurator VA	1	1
Gateway VA	6	16
Service VA	6	8
Connector VA	2	4
Data VA	6	16

NOTE: In the above sizing the Gateway VA can handle 2.000 users

An example of how the logical design for 1.000 Horizon Workspace Files entitled users could look with service high availability in mind is shown in the figure below.



The design above takes an N+1 approach for each VA so everything service part of Horizon Workspace is redundant, except for the configurator-va, which is not required to be constantly up for services to continue working. This approach ensures service high availability for all the components but takes up additional resources.

The only component requiring load balancer/reverse proxy is the Gateway VAs.
For more information on how to configure a reverse proxy in front of Gateway VAs please refer to the following document: https://communities.vmware.com/docs/DOC-24577

Best practices

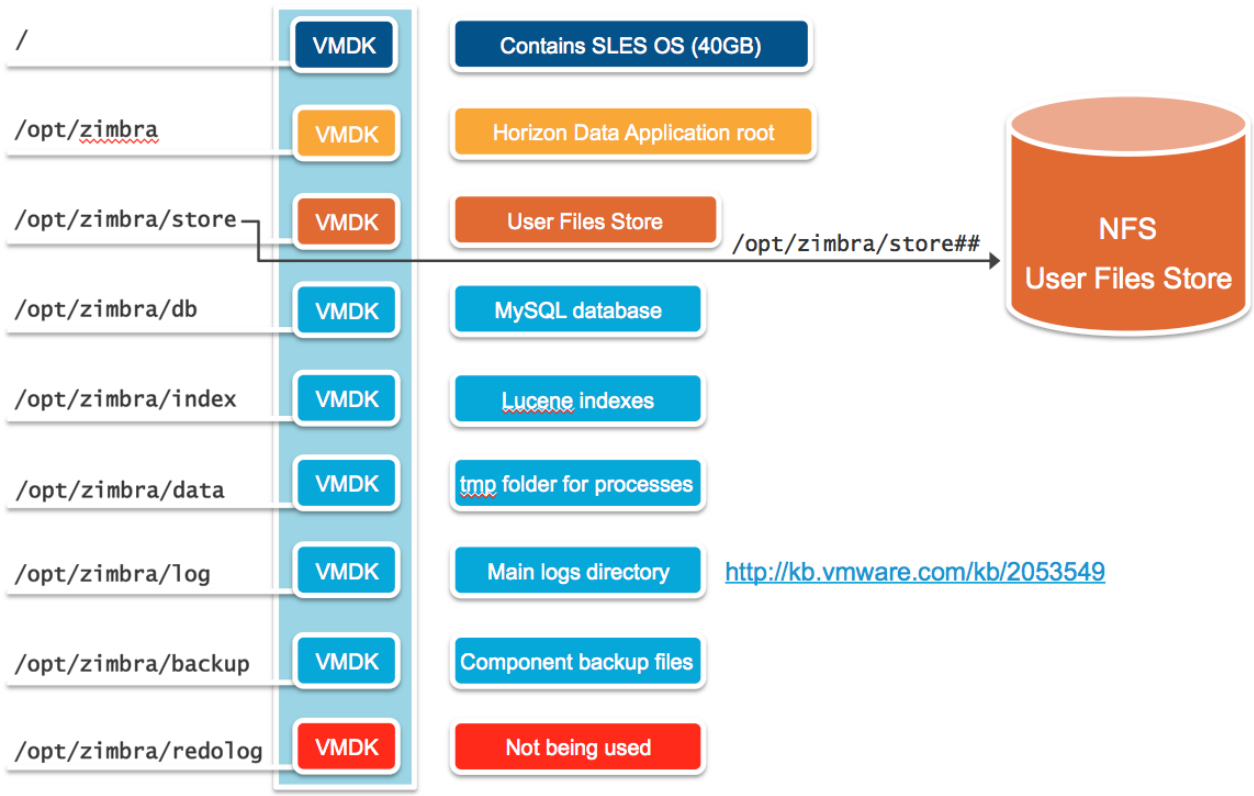
The following lists the best practices and recommendations when architecting and scaling Horizon Workspace.

- Do not keep users on the first Data VA node (Exclude from COS)
- Use NFS mounts to store user files instead of local VMDK
- Do not go above 1.000 active Horizon Files users for each Data VA
- Backup each individual VA thats part of the vApp construct
- Do not place Gateway VAs in DMZ
    - Keep all VAs in the same portgroup (VLAN) for a supported configuration

Disks layout and main folders

The Data VA comes with 9 VMDKs which is mounted to different mount points inside the SLES operating system. A description of the disks and where they are mounted are outlined below.

/: root
/opt/zimbra: Horizon Data application root folder
/opt/zimbra/store*/: where the files are stored (native NFS or VMDK)
/opt/zimbra/index/: where the index files for searches are produced by Lucene
/opt/zimbra/db/: MySQL databases storing user metadatas
/opt/zimbra/data/: OpenLDAP data (first Data VA only)
/opt/zimbra/backup/: Components backup folder
/opt/zimbra/log/: Main location of log files
/opt/zimbra/docs: Different documentation files(beware, not everything is updated for Data)



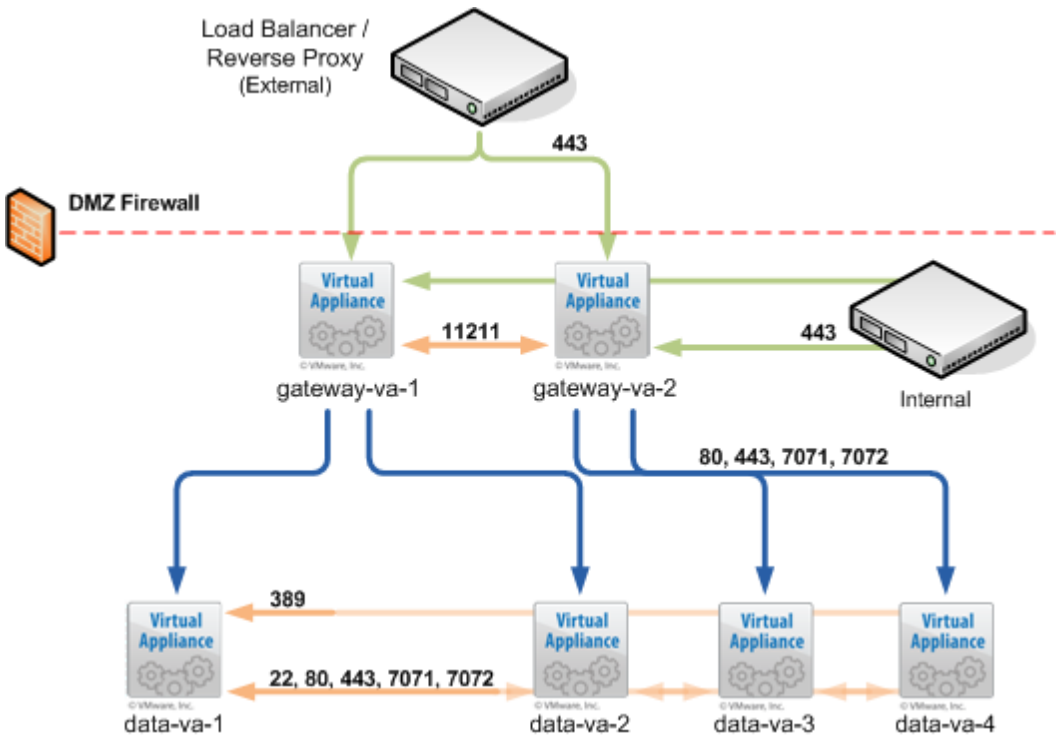
At the moment the only documented action for extending disk space on VMDKs is for the User Files Store. The reason why we don't document the others is that we allocated enough space on the other mountpoints to sustain conservatively the 1000 users we were talking about. Should one of these mount points become full, the recommendation would be to horizontally scale, and therefore accounts moves should be put in place. In exceptional situations and under certain usage scenarios, extension of the other partitions can be done, but GSS should be contacted first for support and recommendations.

Load balancers, gateway-va and backends relationship

When a user logs into Horizon Workspace parts of the WebUI presented is served from the Data VA where the user resides if said user is entitled to Horizon Files. If the user is not entitled to Horizon Files the WebUI will be served from any available data-VA. Availability is determined by the nginx running on gateway-VAs and its upstream configuration monitoring data-VAs. Round-Robin (RR) is used to choose between available data-VAs.

Depending on the connector type being used, on login, the authentication page shows up being loaded by any data-VA backend in a RR fashion. In the case of using kerberos enabed connectors for Windows SSO the authentication page is being loaded from the connector and not a backend data-VA. When authentication is granted, a token is being produced, memcached tables updated on the gateways-VAs with the correct data-va node to be contacted from now on.
Memcached data is shared between gateway-VAs to keep session states should a gateway-VA fail. The ports and their flow is detailed in the picture below.

NOTE: The diagram below is only showing gateway-VA and data-VA traffic



Nginx configuration files in /opt/vmware/nginx/conf on gateway-VAs have the upstream servers (backends) defined and the SSL certificate to be used. Typically in a load balanced setup the SSL traffic terminates at the load balancer and not directly at the gateway-VAs. The configuration file for all upstream servers is available at:

/opt/vmware/nginx/conf/gen/upstream-443.conf
This file is automatically updated whenever additional data-VAs, service-VAs or connector-VAs with *useGatewayAsIDP=n* are added and should not be edited manually in any case.

Communication to the backends is on HTTPS for added security.
If a Data VA goes down, predefined timeouts will go off, so that the backend will be excluded from the RR pool at login time. Users with files stored on such nodes won't be available until restored from backups.
To see the cached route tables, on a gateway-VA node:

```
memcached-tool localhost:11211 dump

Dumping memcache contents
  Number of buckets: 1
  Number of items  : 5
Dumping bucket 2 - 5 total items
add 90761a27-b15d-44f2-b82f-88a4ba76af98 0 1374142155 1
0
add dcfe4749-4c12-40df-bdbd-4689cc8e169f 0 1374142155 1
0
add hzn-route:user1@domain.com:33 0 1374141908 19
192.168.112.173:443
```

The output shows the hzn-route for user1 and the upstream server (data-VA) where the user session and files resides by IP and port.
User initiated traffic will never reach any upstream servers; HTTPS at the gateway, load balancer or kerberos enabled connector level is the only point where user initialized traffic will terminate. Data-VAs, service-VAs will never be reached directly by the user.

Preview Server - Microsoft Office

To provide the users with document previews a preview server is required either in the form of a Libre Office installed on each Data VA or with one or more dedicated Microsoft Windows Preview servers running dedicated on Windows 7 or Windows Server 2008 R2.
Please refer to the official product documentation on how to setup and configure Microsoft Windows Preview.

The following table outlines recommendations for sizing Microsoft Preview Server.

	Number of vCPUs	**RAM (GB)**	**Users**
MS Preview Server	8	4	1.000

When using the above recommendation around MS Preview Server sizing each server can handle up to 1.000 users.

It is possible to add additional Microsoft Preview Servers and placing them behind a load-balancer or simple Round-Robin.

Backup & restore

Each Data VA in Horizon Workspace 1.5 is a full VM and does not rely on any snapshots to function.
Each Data VA should be treated as an individual VM and backup as a single entity including any User Data store mounted be it internal (VMDK) or external (NFS) mounted stores.

When backing up a Data VA with an external NFS based store both the VM and the NFS store should be backed up as close as possible to each other timewise as to not have the store - the files - and the MySQL and OpenLDAP get out of sync as to lose or corrupt users data.

The first Data VA contains the OpenLDAP service which is unique to this VA. The OpenLDAP implementation in Horizon Workspace Files is not a multi-master implementation which means that special precautions should be taken when doing backup and restore operations. As user data is stored in multiple different locations (OpenLDAP, MySQL, Postgres/Oracle) it is important that all VAs in Horizon Workspace is backed up as close as possible to each other timewise as to not have anything be out of sync upon restore operations.

The official recommendations around backup and restore within Horizon Workspace can be found via : http://www.vmware.com/pdf/hw10-backup-best-practices.pdf

Performance tuning

NOTE: Performance analysis is a really complex topic, therefore the following is just an overview of the most relevant aspects of an investigation when poor performance is experienced, or suspected. Regardless of what's being described here, tuning and its evaluation should be carried over exclusively by VMware GSS via a SR.

Mainly, tuning is usually done by acting on the following:

- Storage IO performance
- JVM memory tuning (GC, heap, etc)
- Threads allocation limits
- …

Horizontal scaling should be evaluated when vertical is not viable anymore, that is adding new data-va nodes. What's being described here is vertical tuning only.

zmdiaglog overview

This tool is currently under readaptation, at the moment it still has some issues, but it is though currently able to provide enough data and performance details to provide enough elements for the majority of any investigation:

- OS/HW overview
- Various stats CSV files for perfomance charting
- Main configuration files
- Log files
- 10 samples each 10s apart of the following outputs:
    - top
    - ps
    - netstat
    - stats from /proc/
    - mailboxd java thread dumps

For performance investigations, it's *imperative* that it is launched during the issue going on, before triggering any restart or other disruptive actions. Although this might mean a slightly longer recovery period (ranging from a couple of minutes, to even tens according to the vRAM amount if the `-a` option is used for dumping the heap), this allows us to take a snapshot of the system trend in the very moment they are showing signs of problem. This will also result in a slightly higher server load, and -with heap dump in particular- quite a few GBs

Usage

At the moment, zmdiaglog requires some workarounds to make it work. As root on a data-va node:

```
$ /opt/zimbra/libexec/zmdiaglog
Unable to determine mailboxd pid
No '-a' argument, skipping heap and coredump collection.
ZCS mailboxd pid:
[... output cropped ...]
 *** Diagnostics collection done.  Data stored in
/opt/zimbra/data/tmp/zmdiaglog-zcs7-multi-ldap.cork.zimbralab.com.201
30723-141747.13056/.
Skipping zip creation.
```

The resulting directory will now be stored:

```
$ ZMDIAG='/opt/zimbra/data/tmp/zmdiaglog-<FQDN>.<DATE>'
$ chown -R zimbra: $ZMDIAG
$ su - zimbra
$ zmstat-chart-config > $ZMDIAG/stats/zmstat-chart.xml
$ cd $ZMDIAG;  mkdir charts;
$ /opt/zimbra/bin/zmstat-chart -s stats/ -d charts/ -c
./zmstat-chart.xml
$ cd ..
$ tar cvzf $ZMDIAG.tar.gz $ZMDIAG
```

Now you can copy the archive away in a personal computer, then open `$ZMDIAG/charts/index.html` to check the charts just produced. Here is an example:

Charts

Speaking of the charts themselves, some of them are blank, and that's due to some adaption still to be done to adapt zmdiaglog to work properly on data-va nodes. This will be addressed in our next releases, but for the time being, there's more than enough to get started with Horizon Workspace 1.5.

The main charts deserving attention are:

Total CPU

Good for checking whether CPU (shares, # of cores, etc) might be a bottleneck or not. IOwait is charted as well.

Individual CPU sys time

If high, this usually indicates IO slow and/or lagging. If running purely on VMDKs, you should check the datastore performance charts in vSphere, if on NFS, check first the other charts,

then investigate on the Storage side for performance issues and/or network lags of any kind.

Individual CPU iowait time

Another measurement to verify IO performance. The more, the worse, as it means we wait too much for writes/reads to commit/happen.

Disk Utilization (Top 10 avg)

Allows to see the absolute usage of each mounted device in a given moment.

Process CPU Time (Top 10 avg)

Since we have so many services in a data-va node, we can get a bird's eye view of which is impacting the most at a given moment.

SOAP: Invocation Count: Summary (Top 10 max)

This is quite useful, as we see what's been summoned the most. Please note, everyt action in data is mapped and invoked through SOAP calls.

SOAP: Average Call Duration: Summary (Top 10 avg)

Together with the Invocation Count summary , it gives the whole picture on what's being done and how impacting this was.

MySQL: InnoDB Buffer Pool Hit Rate

Ideally this should be 1000, but 990+ without showing reboots during the charts time window is a good number. `innodb_buffer_pool_size` is automatically sized on RAM increases, so there should be no need to change it though.

MySQL: Database Connections In Use

Usually database connections are reused, so you hardly see more than 2-3 at the same time. But due to possible loops, lags, or delays, this will actually spike if the connection is not released, and new connections will stack up.

MySQL: Total Slow Queries Count

It's a monotonically increasing graph, and it's just useful to see how fast this builds up, as slow queries (either uncached, or complex) will always be happening from time to time. This graph after a reboot is expected to ramp up fast. If steep long ramps happen during a normal production day, it might be worth checking MySQL as a whole.

Mailboxd: Minor Garbage Collection Time | Count

If too many happen, or they take too long, it might be that we need to increase the vRAM, as this is possibly due to a small newgen area. Increasing the vRAM and rebooting will increase `mailboxd_java_heap_size`, which will therefore increase the new gen repartition

(`mailboxd_java_heap_new_size_percent` -25%- of the total heap)

Mailboxd: Major Garbage Collection Time | Count

Major GCs are necessaries from time to time, but the less the better. When they happen, the whole JVM freezes for a small time (***zmmailboxd.out*** contains the info). If this happens too often, it might be that either there's a memory leak, or a bug involving memory allocation, or simply the vRAM is not enough and has to be increased.

convertd: CPU time used

This is tightly coupled with the preview and indexing activities, it might be a good sign for triggering an investigation towards these two directions.

System command outputs

top

The *`top`* output contains a lot of useful informations. Being 10 top outputs each captured 10s apart, it allows to see and quantify trends. Given that we already have good information on the most vCPU-demanding processes, the most noteworthy details are the swap usage and the load average.

threaddumps

Threaddumps are literally the detailed report of any thread going on within the `mailboxd` process and JVM-related actions. This enables us to understand what actions might have gotten stuck, which is taking the most time, which is waiting for what, and so on. Threaddump analysis is falls outside the scope of this whitepaper though, but plenty of information can be found on the net.

lsof

`lsof` is a standard linux command that outputs every file descriptor, socket, pipe, FIFO on the system, and it's often useful to see what files where pointed by `mailboxd` at the time of the issue, among other things.

ps

The processes here are listed in full with all their invocation parameters. This is quite useful to see whether processes are being called with the wrong settings, or if there's something that shouldn't be listed and viceversa.

Log files

There are quite a few here, and summarizing all of what you can find in which log file would

be close to impossible. The general overview is as follow:

mailbox.log

This is the main resource for finding information on everything logged by mailboxd, our main java application handling most of the activities. Therefore we can find log entries for:

- Purging actions
- Auditing events
- Previewing errors
- Indexing issues
- Files inconsistencies/IO issues
- Main service restarts
- …

Really, if in doubt on where something related to data might be, this is definitely the #1 spot where to look into.

zmmailboxd.out

The JVM bit of logging goes here, in particular we can find:

- Crashes information
- GC activities details and times the JVM froze
- Useful entries when `mailboxd` doesn't start and mailbox.log shows nothing.
- Threaddumps when either one of the following is used:
    - `zmthrdump` (has quite a few options, see `--help`)
    - `jstack <mailboxd pid>`
    - `kill -QUIT <mailboxd pid>`

zimbra.log

The following are recorded here:

- `zmmailboxdmgr` status healtcheck (mailboxd wrapper)
- `zmconfigd` activities (zmconfigd is our watchdog process for triggering services restarts when required)
- `slapd` logging (that is OpenLDAP main process, in case we are on the original data-va hosting it)

mysql_error.log

As the name suggests, here is where MySQL logs its reboots, crashes, and so on.

myslow.log

Here every “slow” query gets logged, that is any query that is a miss in the innodb_buffer_pool_size and that gets to be fetched from the disks.

YYYY_MM_DD.trace.log

This contains extracts from the headers of HTTP calls and the method is listed as well, useful under certain troubleshooting scenarios.

access_log.YYYY-MM-DD

Very similar in scoping to what `trace.log` logs already, but more readable and meant to be more thorough and verbose.

messages

It really is the standard log file we are all well acquainted with. It’s the underlying SLES 11sp2 logging here.

NOTE: *Once again, please note that performance tuning is a really complex topic, and this chapter should be seen just as an FYI, rather than a manual. In case doubts arise regarding performances, the best and only thing to do is to contact GSS while detailing as much as possible the context and symptoms of the issue, as seen by end users and administators.*