Alfresco Enterprise

Alfresco 4.2

Day Zero Configuration Guide

Contents

Alfresco Day Zero Configuration Guide

By default, Alfresco configuration is optimized for single user evaluation of Alfresco. This configuration minimizes resource usage at the expense of scalability (particularly scalability in the presence of large concurrent traffic volumes). Therefore, for any other use of Alfresco (including but not limited to QA, performance/scalability testing, production, production mirror, and disaster recovery), Alfresco strongly recommends that additional configuration be performed.

The Day Zero Configuration Guide describes the generally valid configuration steps that should be taken to achieve this, regardless of the specific Alfresco use case. It describes the steps to take before Alfresco is started for the first time, together with optional configuration and tuning to reach optimal Alfresco performance.

This document does not describe the full breadth of Alfresco configuration options that can be leveraged to scale Alfresco in use case specific ways, but aggregates a general set of recommendations from the official documentation in a one-stop-shop document. For additional documentation, see the rest of the product documentation, access the Knowledge Base through the Support Portal, or the Scalability Blueprint document.

This is a live document generated out of Alfresco product documentation, so make sure that you check this page often for updates. Check the PDF publication date to make sure you are using the latest available version.

Day Zero architecture validation

This section describes the steps required to validate the architecture to ensure that it meets the prerequisites for an Alfresco installation.

Check the following steps to validate the architecture:

1. Check the supported stacks list for Alfresco.
2. Validate the architecture.
3. Optimize the hardware settings.
4. Software requirements settings.
5. Validate the database.
6. Validate the Operating System.
7. Validate and tune the JVM.

Supported platforms

The supported platforms are the combinations of operating systems, databases, and application servers that are tested and certified for Alfresco.

For the latest list, refer to the **Supported Platforms** page at http://www.alfresco.com/services/subscription/supported-platforms/.

Validating the architecture

Use these steps to validate the architecture of an Alfresco installation against the recommended prerequisites.

1. Validate the disk performance. See Hardware settings for more information.
2. Validate network performance.

In each case, the goal is to minimize the *latency* (response time) between Alfresco and the storage system, while also maximizing bandwidth. Low latency is particularly important for database I/O, and one rudimentary test of this is to `ping` the database server from the Alfresco server - round trip times greater than *1ms* indicate a sub-optimal network topology or configuration that will adversely impact Alfresco performance. *Jitter* (highly variable round trip times) is also of concern, as that will increase the variability of Alfresco's performance. The standard deviation for round trip times should be less than 0.1ms.

An example ping is:

```
ping -c 20 dbserver.com
    ...
20 packets transmitted, 20 received, 0% packet loss, time 19029ms
rtt min/avg/max/mdev = 0.286/0.750/1.818/0.391 ms
```

3. Ensure that your system has a clock speed of greater than 2.5Ghz. See Hardware settings for more information.
4. Ensure that you allocate extra virtual memory on Linux systems.

   This extra space is required for processes within the Alfresco server that use the fork operation (for example, ImageMagick). Allocating this extra space ensures that Alfresco has sufficient memory to complete fork operations without reserving extra RAM.
5. Validate the database.

   Alfresco does not provide technical support for maintaining or tuning your relational database. Ensure that your project has access to a certified database administrator (DBA) to support your Alfresco installation.

   Regular maintenance and tuning of the Alfresco database is necessary. Specifically, all of the database servers that Alfresco supports require at the very least that some form of index statistics maintenance be performed at frequent, regular intervals to maintain optimal Alfresco performance.

   Index maintenance can have a severe impact on Alfresco performance while in progress, hence it needs to be discussed with your project team and scheduled appropriately.
6. Validate the operating system.
   a. Ensure that your chosen OS has been officially certified for use with Alfresco (refer to the Supported Stacks list for details).
   b. Alfresco recommends that a 64-bit OS is used. See the Supported Stacks list for information on the exceptions.
   c. If your system is running Windows Server 2008 R2 or Windows 7, you need to install Fix373886. This is to avoid the "no buffer space available" exception on your system. For details, see the Microsoft Support website.
7. Validate and tune the JVM.

   Ensure that your chosen JDK-enabled Java Virtual Machine has been officially certified for use with Alfresco (refer to the Supported Stacks list for details).

   For information on configuring and tuning the JVM, refer to Tuning the JVM.

Hardware settings

This section describes how to validate your I/O subsystems and CPU.

- **I/O**: One of the primary determinants of Alfresco's performance is I/O. Optimize the following, in priority order:
  1. I/O to the relational database Alfresco is configured to use.

2. I/O to the disk subsystem on which the Lucene or Solr indexes are stored.
3. I/O to the disk subsystem on which the content is stored.

In each case, the goal is to minimize the latency (response time) between Alfresco and the storage system, while also maximizing bandwidth.

Alfresco recommends that the disk throughput is greater than 200MB/sec. On Linux, use the `hdparm` tool to measure disk throughput. The following sample output is on an SATA disk:

```
hdparm -tT /dev/sda1
/dev/sda1:
Timing cached reads:   27998 MB in  2.00 seconds = 14018.28 MB/sec
Timing buffered disk reads: 536 MB in  3.01 seconds = 178.05 MB/sec
```

Other useful tools for detecting disk I/O issues include `dd`, `seeker`, and `iozone`.

- **CPU**: Alfresco will function correctly on virtually all modern 32bit and 64bit CPUs, however, for production use, Alfresco recommends a clock speed greater than 2.5Ghz to ensure reasonable response times to the end user. Although it is not strictly necessary, a 64bit architecture is also recommended, primarily because it allows the JVM to utilize more memory (RAM) than a 32bit architecture.

  CPU clock speed is of particular concern for the Sun UltraSPARC architecture, as some current UltraSPARC based servers ship with CPUs that have clock speeds as low as 900Mhz, well below what is required for adequate Alfresco performance! If you intend to use Sun servers for hosting Alfresco, please ensure that all CPUs have a clock speed of at least 2.5Ghz.

  At time of writing, this implies that:

  - an X or M class Sun server is required, with careful CPU selection to ensure 2.5Ghz (or better) clock speed
  - T class servers should not be used, as they do not support CPUs faster than approximately 2Ghz

  Understandably, Alfresco is unable to provide specific guidance on Sun server classes, models or configurations, so you should talk with your Sun reseller to confirm that minimum CPU clock speed recommendations will be met.

Software requirements

The following table lists the required software that must be on your system for manually installing Alfresco.

Component	Installer Version	Recommendation
Java SE Development Kit (JDK)	JDK 7 U25 X64	The Sun Microsystems JDK 7 is required. The `JAVA_HOME` environment variable must be set to the location of the JDK installation.
Application server	Tomcat 7.0.42	Alfresco runs within an application server. Alfresco Enterprise runs within Tomcat but can be installed on other application servers.
Database	PostgreSQL 9.2.4	Alfresco comes preconfigured with the PostgreSQL 9.2.4 database. If you intend to use Alfresco in a production environment, you can use one of the supported databases. For the latest information on supported databases, refer to the Alfresco website. For information on configuring the database settings, refer to Configuring databases.

Component	Installer Version	Recommendation
LibreOffice	LibreOffice 4.0	Alfresco uses LibreOffice for transforming documents from one format to another, for example, a text file to a PDF file.
ImageMagick	ImageMagick 6.8.6-6	Alfresco uses ImageMagick to manipulate images for previewing.
GhostScript	Supported Ghostscripts for different operating systems: Ubuntu – Ghostscript 8.71 Rhel 5.5 - Ghostscript 8.15.2 SLES 11 - Ghostscript 8.62 RHEL 6 - Ghostscript 8.70 RHEL 6.4 X64 - Ghostscript 8.70 Windows/ Solaris - latest stable version or Ghostscript 9.0.7.	Alfresco uses GhostScript in conjunction with ImageMagick to manipulate images for previewing.
Flash Player	Flash Player Version 10.x	Alfresco Share requires Flash Player Version 10.x to upload multiple files and view Flash previews. If you do not install Flash, you see the upload screen for single files. Use the latest (stable) version of Flash Player for your platform.
SWF Tools	SWFTools 0.9.2	Alfresco Share uses the pdf2swf utility for previewing PDF files. If you do not install SWF Tools, you will not see PDF previews, but image previews will still be available.

Database validation

This section describes how to validate your database to ensure that it meets the prerequisites for an Alfresco installation.

Disclaimer: Alfresco is unable to provide specialized support for maintaining or tuning your relational database. You MUST have an experienced, certified DBA on staff to support your Alfresco installation(s). Typically this will not be a full time role once the database is configured and tuned and automated maintenance processes are in place. However an experienced, certified DBA is required to get to this point.

Maintenance and Tuning:

As with any application that uses a relational database, regular maintenance and tuning of the Alfresco database and schema is necessary. Specifically, all of the database servers that Alfresco supports require a minimum level of index statistics maintenance at frequent, regular intervals. Unless your DBA suggests otherwise, Alfresco recommends daily maintenance.

Relying on your database's automated statistics gathering mechanism may not be optimal – consult an experienced, certified DBA for your database to confirm this.

Index maintenance on most databases is an expensive, and in some cases blocking, operation that can severely impact Alfresco performance while in progress. Consult your experienced, certified DBA regarding best practices for scheduling these operations in your database.

The following table describes example commands for specific databases. These commands are for illustration only. You must validate the commands required for your environment with your DBA.

Database	Example maintenance commands
MySQL	ANALYZE - consult with an experienced, certified MySQL DBA who has InnoDB experience (Alfresco cannot use a MyISAM database and hence an InnoDB-experienced MySQL DBA is required). Refer to the following link: http://dev.mysql.com/doc/refman/5.6/en/analyze-table.html.
PostgreSQL	VACUUM and ANALYZE – consult with an experienced, certified PostgreSQL DBA. Refer to the following link: http://www.postgresql.org/docs/8.4/static/maintenance.html.
Oracle	Depends on version – consult with an experienced, certified Oracle DBA. Refer to the following link: http://download.oracle.com/docs/cd/B19306_01/server.102/b14211/stats.htm#PFGRF003.
Microsoft SQL Server	ALTER INDEX REBUILD (http://technet.microsoft.com/en--#us/library/ms188388.aspx), UPDATE STATISTICS (http://technet.microsoft.com/en--#us/library/ms187348.aspx) – consult with an experienced, certified MS SQL Server DBA
DB2	REORGCHK () http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/index.jsp?topic=/com.ibm.db2.luw.admin.cmd.doc/doc/r0001971.html RUNSTATS () http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/index.jsp?topic=/com.ibm.db2.luw.admin.cmd.doc/doc/r0001980.html

Operating System validation

You should ensure that your chosen OS has been officially certified for use with Alfresco.

Alfresco is not sensitive to changes to the OS configuration, beyond the impact on I/O performance (for details, please see the I/O section).

Java Virtual Machine validation

You should ensure that your chosen JDK-enabled Java Virtual Machine has been officially certified for use with Alfresco.

For information on configuring and tuning the JVM, see Tuning the JVM.

Day Zero environment validation

It is important to validate certain environment-specific items prior to installing Alfresco.

You can validate a setup either by using the Alfresco Environment Validation Tool (EVT) or without using the EVT.

Environment validation using EVT

This topic describes how to validate a setup by using the Alfresco EVT.

An Environment Validation Tool (EVT) is also available that can validate most of the following requirements. The tool and documentation are available at https://artifacts.alfresco.com/nexus/content/repositories/alfresco-docs/alfresco-environment-validation/latest/index.html. All versions of the EVT are also available on the Alfresco Nexus server at https://artifacts.alfresco.com/nexus/index.html#nexus-search;quick~alfresco-environment-validation.

To get a list of the tests that currently run in the latest version of the tool, see the sample report at https://artifacts.alfresco.com/nexus/content/repositories/alfresco-docs/alfresco-environment-validation/latest/usage.html.

Environment validation without EVT

This topic lists some tests you can do to validate a setup without using the EVT.

1. Validate that the host name of the server can be resolved in DNS.

   This is required if Alfresco is going to be configured in a cluster.

   Using an incorrect host name or a host name that no longer resolves to its own IP address can give an internal error, such as `ObjID already in use`. You can get more information about this error by adding the following line into the `log4j.properties` file:

   ```
   log4j.logger.org.springframework.remoting.rmi.RmiServiceExporter=debug
   ```

   To resolve this error, you can either:

   - Validate that the IP address and the host name of the server are correctly set in the `/etc/hosts` file. For example, if you set the IP address as `10.20.30.40` and the host name as `ip-10-20-30-40`, the content of the `/etc/hosts` file should contain the following entry:

     ```
     10.20.30.40 ip-10-20-30-40
     ```

   - Specify the correct IP address in the `alfresco-global.properties` file as shown below:

     ```
     alfresco.rmi.services.host=10.20.30.40
     ```

2. Validate that the user Alfresco will run as can open sufficient file descriptors (4096 or more). See http://stackoverflow.com/questions/34588/how-do-i-change-the-number-of-open-files-limit-in-linux for more information.

3. Validate that the ports on which Alfresco listens are available.

   To check port availability, use the `netstat -lnpv` command on Linux, or use the `netstat -anl` command on OSX.

   The ports listed in the following table are the defaults. If you are planning to reconfigure Alfresco to use different ports, or wish to enable additional protocols (such as HTTPS, SMTP, IMAP or NFS), update this list with those port numbers.

Protocol	Port number	Notes
FTP	TCP 21	On Unix-like operating systems that offer so-called “privileged ports”, Alfresco will normally be unable to bind to this port, unless it is run as the root user (which is not recommended). In this case, even if this port is available, Alfresco will still fail to bind to it, however for FTP services, this is a non-fatal error. The Alfresco FTP functionality will be disabled in the repository.
SMTP	TCP 25	SMTP is not enabled by default.
SMB/NetBT:	UDP 137,138	
SMB/NetBT:	TCP 139,445	On Unix-like operating systems that offer so-called “privileged ports”, Alfresco will normally be unable to bind to this port, unless it is run as the root user (which is not recommended). In this case, even if this port is available, Alfresco will still fail to bind to it, however for CIFS services, this is a non-fatal error. The Alfresco CIFS functionality will be disabled in the repository.
IMAP	TCP 143	IMAP is not enabled by default.
SharePoint Protocol	TCP 7070	This port is only required if you install support for the SharePoint Protocol.
Tomcat Administration	TCP 8005	
HTTP	TCP 8080	
RMI	TCP 50500	

4. Refer to the **Supported Platforms** page at http://www.alfresco.com/services/subscription/supported-platforms/ to validate the installed JVM version.
5. Validate that the directory in which the JVM is installed does not contain spaces.
6. Validate that the directory in which Alfresco is installed does not contain spaces.
7. Validate that the directory Alfresco will use for the repository (typically called `alf_data`) is both readable and writeable by the operating system user that the Alfresco process will run as.
8. Validate that you can connect to the database as the Alfresco database user, from the Alfresco server.

   Ensure that you install the database vendor's client tools on the Alfresco server.
9. Validate that the character encoding for the Alfresco database is UTF-8.
10. (MySQL only) Validate that the storage engine for the Alfresco database is InnoDB.

11. Validate that the relevant third-party softwares are installed. See Software requirements for more information.
12. (RHEL and Solaris only) Validate that LibreOffice is able to run in headless mode.

Day Zero configuration

This section describes the configuration changes that will improve Alfresco reliability, stability and performance when used for anything other than single user evaluation purposes.

Disabling unneeded Alfresco Features

You can disable common product components, if you do not require them for your Alfresco instance. This summary gives the example property settings for disabling the main components.

Avoid disabling features if you are unsure of what you are doing. Refer to Alfresco Support for recommendations.

Add the following property settings to the `alfresco-global.properties` file:

Property	Description
`system.usages.enabled=false`	Disables quotas or user usages. Avoid setting this property when using Hybrid Sync, as the service might be impacted.
`replication.enabled=false`	Disables content replication.
`audit.enabled=false`	Specifies a way to globally enable or disable the auditing framework.
`cifs.enabled=false`	Specifies whether to enable or disable the CIFS server.
`ftp.enabled=false`	Specifies whether to enable or disable the FTP server.
`system.workflow.engine.jbpm.enabled=false`	Specifies whether to enable or disable the jBPM workflow engine. The jBPM workflow engine is disabled by default.
`system.workflow.engine.activiti.enabled=false`	Specifies whether to enable or disable the Activiti workflow engine. The Activiti workflow engine is enabled by default.
`transferservice.receiver.enabled=false`	Disables the Transfer or Replication Service receiver.
`sync.mode=OFF`	Use this property to disable synchronization permanently.
`lucene.indexer.cacheEnabled=false`	Specifies whether to enable or disable index level caching.
`lucene.indexer.contentIndexingEnabled=false`	Specifies whether or not the content of the document is indexed. If false, content is not indexed.

Property	Description
`index.tracking.cronExpression=* * * * ? 2099`	Specifies the scheduled Lucene index tracking for the future. Do not use this property if you are using `index.subsystem.name=lucene`
`audit.alfresco-access.enabled=false`	To enable generation of audit data that you can view in Explorer or Share, you will need to enable the `audit.alfresco-access.enabled` property.
`audit.filter.alfresco-access.default.enabled=false`	Disables auditing of Alfresco accesses. Do not use this property if you require auditing.
`home.folder.creation.eager=false`	Disables the automatic creation of home folder for new users.
`db.schema.update=false`	Specifies whether the system bootstrap should create or upgrade the database schema automatically.
`syncService.mode=OFF`	Disables Cloud sync.
`activities.feed.notifier.enabled=false`	Disables the Share Activities email notification.
`sync.pullJob.enabled=false`	Use this Alfresco cloud sync property to disable synchronization temporarily.
`sync.pushJob.enabled=false`	Use this Alfresco cloud sync property to disable synchronization temporarily.

Repository system configuration files

The path for `<configRoot>` is different depending on your application server. For example:

- Tomcat: `<TOMCAT_HOME>\webapps\alfresco\WEB-INF`
- JBoss: `<JBOSS_HOME>\server\default\tmp\deploy\tmp*alfresco-exp.war\WEB-INF`

The system configuration files are maintained by Alfresco and contained in `<configRoot>` and `<configRoot>\classes\alfresco`.

The preferred method of configuring Alfresco is to extend the default files using the global properties file (`alfresco-global.properties`).

The following files represent the core of the application configuration:

1. `<configRoot>\classes\alfresco\application-context.xml`

   This file is the starting point of the Spring configurations. This file only performs imports, including a wild card import of all `classpath*:alfresco/extension/*-context.xml` files.
2. `<configRoot>\classes\alfresco\core-services-context.xml`

   Core Alfresco beans are defined here, including the importing of properties using the `repository-properties` bean.
3. `<configRoot>\classes\alfresco\repository.properties`

This file is imported by the `repository-properties` bean. The file defines the core system properties, including:

- `dir.root`

  This folder is where the binary content and indexes are stored. The `alf_data` folder is where they are stored by default, but you should change this to your own location. The path is relative by default, but it must point to a permanent, **absolute**, backed-up location for data storage.
- `dir.auditcontentstore`

  This folder is where the audit's content store is stored.
- `dir.indexes`

  This folder contains all Lucene indexes and deltas against those indexes.

  Alfresco recommends that you **do not store Lucene indexes on an NFS volume**. The indexes must be on a local disk. For best performance, use a separate hardware chain (for example, controller, disk, and so on) to avoid I/O contention with other operations, like storing content and other applications.

  When using the Lucene index subsystem, make sure the disk is local to the web application server. Alfresco recommends the use of an SSD drive to store the indexes. When using Solr, the indexes are local to the Solr server. If a disk full error occurs while the system is running, this can lead to indexes corruption. Avoid disk full errors by leaving sufficient free disk on the indexes partition. Index subsystems merge the indexes, so if you use the current amount of disk space used to store your indexes to evaluate future needs in disk space, be sure you multiply this value by 2. If you want to evaluate the disk space needed to store Solr indexes when you switch from Lucene to Solr, also multiply the Lucene index size by 2. In Solr, most of the content is indexed twice: once using the locale of the document, and once using the standard analyzer.
- `db.*`

  These are the default database connection properties.
- `db.schema.update`

  This property controls whether the system bootstrap should create or upgrade the database schema automatically.

Tuning the JVM

The hardware requirements for the Alfresco repository, Explorer, and Share are variable and depend on the number of concurrent users that access the system. You can tune the memory and garbage collection parameters for the JVM to be appropriate for your situation. This section suggests metrics and estimates, but your system may vary.

Hardware

Alfresco degrades gracefully on low-powered hardware, and small installations can run well on any modern server. However, for optimum performance, we recommend the following:

- Use 64 bit systems only. Benchmarks show a significant performance gain when using 64 bit hardware and a 64 bit JRE.
- Use a system with a clock speed above 2.5 GHz.
- Reserve enough RAM for your operating system beyond the memory required for your JVM.

- Keep search indexes on your local disk instead of on network storage.

Disk space usage

The size of your Alfresco repository defines how much disk space you will need; it is a very simple calculation. Content in Alfresco is, by default, stored directly on the disk. Therefore, to hold 1000 documents of 1 MB will require 1000 MB of disk space. You should also make sure there is sufficient space overhead for temporary files and versions. Each version of a file (whether in DM or WCM) is stored on disk as a separate copy of that file, so make allowances for that in your disk size calculations (for DM, use versioning judiciously).

The disk space usage calculation above is only for content storing. It does not take into account any indexes (Lucene or Solr).

Use a server class machine with SCSI Raid disk array. The performance of reading/writing content is almost solely dependent on the speed of your network and the speed of your disk array. The overhead of the Alfresco server itself for reading content is very low as content is streamed directly from the disks to the output stream. The overhead of writing content is also low but depending on the indexing options (for example, atomic or background indexing), there may be some additional overhead as the content is indexed or metadata is extracted from the content in each file.

Virtualization

Alfresco runs well when virtualized, but you should expect a reduction in performance. When using the rough sizing requirements below, it may be necessary to allocate twice as many resources for a given number of users when those resources are virtual. Para-virtualization, or virtualized accesses to native host volumes do not require as many resources. Benchmarking your environment is necessary to get a precise understanding of what resources are required.

JVM memory and CPU hardware for multiple users

The repository L2 Cache, plus initial VM overhead, plus basic Alfresco system memory, is setup with a default installation to require a maximum of approximately 1024MB.

This means that you can run the Alfresco repository and web client with many users accessing the system with a basic single CPU server and only 1024MB of memory assigned to the Alfresco JVM. However, you must add additional memory as your user base grows, and add CPUs depending on the complexity of the tasks you expect your users to perform, and how many concurrent users are accessing the client.

Note that for these metrics, **N** concurrent users is considered equivalent to **10xN** casual users that the server could support.

Number of users	Recommended memory / CPU settings per server
For 50 concurrent or up to 500 casual users	2 GB JVM RAM 2x server CPU (or 1xDual-core)
For 100 concurrent users or up to 1000 casual users	4 GB JVM RAM 4x server CPU (or 2xDual-core)
For 200 concurrent users or up to 2000 casual users	8 GB JVM RAM 8x server CPU (or 4xDual-core)

Concurrent users are users who are constantly accessing the system through Alfresco Explorer with only a small pause between requests (3-10 seconds maximum) with continuous access 24/7. Casual users are users occasionally accessing the system through Alfresco Explorer or WebDAV/CIFS interfaces with a large gap between requests (for example, occasional document access during the working day).

For full performance tuning, contact Alfresco Support or Alfresco Consulting.

Calculate the memory needed for Solr nodes

Solr can have high memory requirements. You can use a formula to calculate the memory needed for the Alfresco internal data structures used in Solr for PATH queries and read permission enforcement.

By default, there are two cores in Solr: `WorkspaceSpacesStore` and `ArchiveSpacesStore`. Normally, each core has one searcher but can have a maximum of two searchers.

In the calculation below:

- N = refers to the number of nodes in the store. Each core's value is calculated separately. If there are more than two cores, you will need to add additional queries to calculate the value for that core (as shown in the example code block below).
- T = refers to the number of transactions in the repository and this is same for each core
- A = refers to the number of ACLs in the repository and this is same for each core
- X = refers to the number of ACL transactions in the repository and this is same for each core

The values for N, T, A and X come from the database. Use the following commands to derive these values:

```
select * from
(select count( * ) N_Alfresco from alf_node where store_id = (select id from
 alf_store where protocol = 'workspace' and identifier = 'SpacesStore')) as
 N1 ,
(select count( * ) N_Archive from alf_node where store_id = (select id from
 alf_store where protocol = 'archive' and identifier = 'SpacesStore')) as N2 ,
(select count( * ) T from alf_transaction ) as T,
(select count( * ) A from alf_access_control_list ) as A,
(select count( * ) X from alf_acl_change_set) as X;
```

For example, if there are three cores, include additional queries to calculate the value for that core, as shown below:

```
select * from
(select count( * ) N_Alfresco from alf_node where store_id = (select id from
 alf_store where protocol = 'workspace' and identifier = 'SpacesStore')) as
 N1 ,
(select count( * ) N_Archive from alf_node where store_id = (select id from
 alf_store where protocol = 'archive' and identifier = 'SpacesStore')) as N2 ,
(select count( * ) N_Version2 from alf_node where store_id = (select id from
 alf_store where protocol = 'workspace' and identifier = 'version2Store'))as
 N3 ,
(select count( * ) T from alf_transaction ) as T,
(select count( * ) A from alf_access_control_list ) as A,
(select count( * ) X from alf_acl_change_set) as X;
```

Memory calculation for the Alfresco data structures associated with one searcher

For a store containing 100M nodes, 100M transactions, 100M ACLs and 100M ACL transactions, 21.6 G of memory is needed. Assuming there are not many ACLs or ACL changes, for 100M nodes, you will need 12G -16G of memory depending on the number of transactions. This calculation is based on the following formula: `120N + 32(T + A + X)` bytes.

Memory calculation for the Solr caches associated with one searcher

The Solr cache will use up to (2N + T + A + X)/8 bytes for an entry in any cache.

The formula to calculate the total memory needed for the caches for a single core is: `(solr.filterCache.size + solr.queryResultCache.size + solr.authorityCache.size + solr.pathCache.size) * (2N + T + A + X)/8` bytes

So, for 100M documents and 100M transactions, 96G of memory is needed using the out of box configuration.

`(512 + 1024 + 512 + 512)(300M)/8 = 96G`

The default cache values needs to change to accommodate a large repository. So, for 100M documents, 100M transactions and reduced cache size, 12G of memory is needed.

`(64 + 128 + 64 + 64)(300M)/8 = 12G`

Solr memory planning

For the Alfresco JVM, the most important parameter is `-Xmx`, which controls the heap. The above formula helps to evaluate the memory required by Solr and for capacity planning. Solr memory requirements increase with the size of the repository but also with the amount of memory you allocate to the Solr caches. Decreasing the Solr cache parameters can dramatically lower the memory requirements, with the drawback of hitting the disk more often. You can set these parameters to different values for the each of the stores.

```
solr.filterCache.size
solr.queryResultCache.size
solr.authorityCache.size
solr.pathCache.size
```

Overall Solr memory use

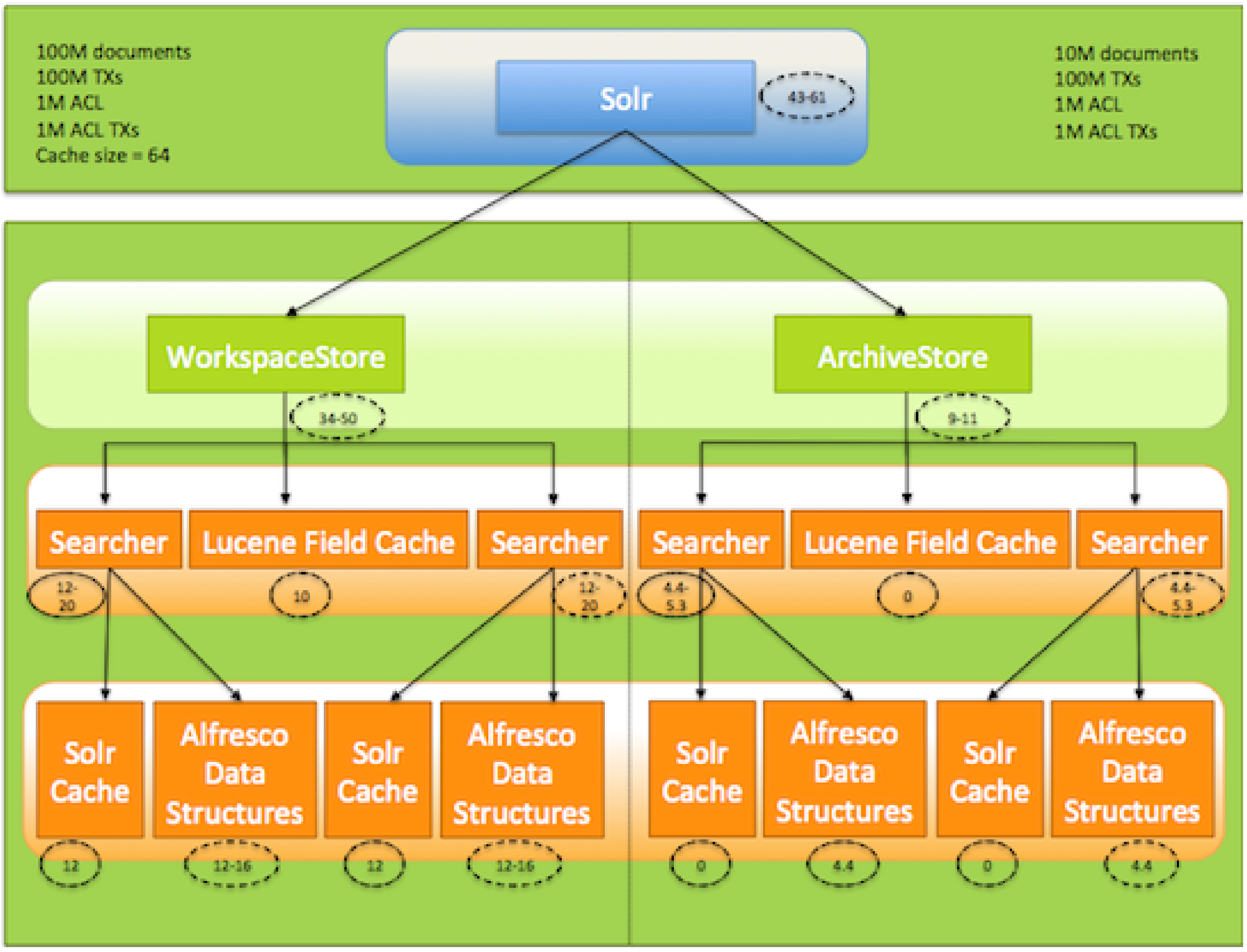
This example is based on the data above.

For WorkspaceStore: Assuming that there are 100M docs, 100M TXs, 1M ACLs and ACL TXs, cache size of 64 entries each for FilterCache, AuthorityCache and QCache, and 128 entries for PathCache, between 12 to 20G of memory is needed per searcher. Normally, there is one searcher live but around commit time there can be two searchers. So, approximately 34 to 50G of memory will be needed in total.

For Archivestore: Assuming that there are 100M transactions, 10M docs and all caches are tuned down, between 4.4G to 5.3G of memory is needed per searcher. Total memory needed for both the searchers will be between 9G to 11G.

So, the total memory requirement for both the cores is between 43G to 61G.

The following diagram shows the overall memory use for a Solr node as explained above:



Minimize the memory requirements for Solr nodes

- Reduce the cache sizes and check the cache hit rate.
- Disable ACL checks using `alfresco.doPermissionChecks=false`
- Disable archive indexing for Solr, if you are not using it. See Solr configuration files for details on how to configure store indexation in Solr.
- Find the exact number of nodes in the store (N), exact number of transactions in the repository (T), number of ACLs (A) and related ACL transactions in the repository (X).
- Since everything scales to the number of documents in the index, add the Index control aspect to the documents you do not want in the index. See Controlling indexes for more information.

Advanced database configuration properties

As an administrator, you need to edit some advanced properties to customize your database configuration. Many properties, however, do not need to be edited.

Alfresco Enterprise supports Oracle, Microsoft SQL Server, DB2, as well as MySQL and PostgreSQL.

The advanced database configuration properties are categorized into two groups based on their relevance:

- properties that you **SHOULD** edit
- properties that you **COULD** edit

The following table describes the properties that you **SHOULD** edit:

Property name	Description	Default value
`db.txn.isolation`	The JDBC code number for the transaction isolation level, corresponding to those in the `java.sql.Connection` class. The value of -1 indicates that the database's default transaction isolation level should be used. For the Microsoft SQL Server JDBC driver, the special value of 4096 should be used to enable snapshot isolation.	`-1`
`db.pool.initial`	The number of connections opened when the pool is initialized.	`10`
`db.pool.validate.query`	The SQL query that will be used to ensure that your connections are still alive. This is useful if your database closes long-running connections after periods of inactivity.	For Oracle database, use `select 1 from dual` For MySQL database, use `select 1` For SQL Server database, use `select 1`

The following table describes the properties that you **COULD** edit:

Property name	Description	Default value
`db.pool.statements.enable`	A Boolean property. When set to `true` it indicates that all pre-compiled statements used on a connection will be kept open and cached for reuse.	`true`
`db.pool.statements.max`	The maximum number of pre-compiled statements to cache for each connection. The Alfresco default is 40. Note that Oracle does not allow more that 50 by default.	`40`
`db.pool.idle`	The maximum number of connections that are not in use kept open.	`-1`
`db.pool.max`	The maximum number of connections in the pool. See the Note below for more information on this property.	`40`
`db.pool.min`	The minimum number of connections in the pool.	`0`
`db.pool.wait.max`	Time (in milliseconds) to wait for a connection to be returned before generating an exception when connections are unavailable. A value of 0 or -1 indicates that the exception should not be generated.	`-1`
`db.pool.validate.borrow`	A Boolean property. When set to `true` it indicates that connections will be validated before being borrowed from the pool.	`true`
`db.pool.validate.return`	A Boolean property. When set to `true` it indicates that connections will be validated before being returned to the pool.	`false`
`db.pool.evict.interval`	Indicates the interval (in milliseconds) between eviction runs. If the value of this property is zero or less, idle objects will not be evicted in the background.	`-1`

Property name	Description	Default value
`db.pool.evict.idle.min`	The minimum number of milliseconds that a connection may remain idle before it is eligible for eviction.	`1800000`
`db.pool.evict.validate`	A Boolean property. When set to `true` it indicates that the idle connections will be validated during eviction runs.	`false`
`db.pool.abandoned.detect`	A Boolean property. When set to `true` it indicates that a connection is considered abandoned and eligible for removal if it has been idle longer than the `db.pool.abandoned.time`.	`false`
`db.pool.abandoned.time`	The time in seconds before an abandoned connection can be removed.	`300`

The `db.pool.max` property is the most important. By default, each Alfresco instance is configured to use up to a maximum of 40. All operations in Alfresco require a database connection, which places a hard upper limit on the amount of concurrent requests a single Alfresco instance can service (that is, 40), from all protocols, by default.

Most Java application servers have higher default settings for concurrent access (Tomcat allows up to 200 concurrent HTTP requests by default). Coupled with other threads in Alfresco (non-HTTP protocol threads, background jobs, and so on) this can quickly result in excessive contention for database connections within Alfresco, manifesting as poor performance for users.

If you are using Alfresco in anything other than a single-user evaluation mode, increase the maximum size of the database connection pool to at least the following setting.

```
[number of application server worker threads] + 75.
```

For a Tomcat default HTTP worker thread configuration, and with all other Alfresco thread pools left at the defaults, this means this property should be set to at least 275.

To increase the database connection pool, add the `db.pool.max` property to the `alfresco.global.properties` file and set it to the recommended value of 275, for example:

```
db.pool.max=275
```

For clarity, add this property immediately after the other database properties.

After increasing the size of the Alfresco database connection pools, you must also increase the number of concurrent connections your database can handle to at least the size of the cumulative Alfresco connection pools. In a cluster, each node has its own independent database connection pool. You must configure sufficient database connections for all of the Alfresco cluster nodes to be able to connect simultaneously. Alfresco recommends that you configure at least 10 more connections to the database than are configured cumulatively across all of the Alfresco connection pools to ensure that you can still connect to the database even if Alfresco saturates its own connection pools. Remember to factor in cluster nodes (which can each use up to 275 database connections) as well as connections required by other applications that are using the same database server as Alfresco.

The precise mechanism for reconfiguring your database's connection limit depends on the relational database product you are using; contact your DBA for configuration details.

Modifying the global properties file

For edits to the `alfresco-global.properties` file, when specifying paths for Windows systems, you must replace the Windows path separator characters with either the `\\` separator or the forward slash `/` Unix path separator. Also, when using folder names like `User Homes`, you must manually escape the space. For example, change the value to `User_x0020_Homes`.

1. Browse to the `<classpathRoot>` directory.

   For example, for Tomcat, browse to the `$TOMCAT_HOME/shared/classes/` directory.
2. Open the `alfresco-global.properties.sample` file.

   This file contains sample configuration settings for Alfresco. To enable or modify a setting, ensure that you remove the comment (#) character.
3. Add a root location for the storage of content binaries and index files in the `dir.root=` property.

   For example, `dir.root=C:/Alfresco/alf_data`.
4. Set the database connection properties.

Property	Description
`db.username=alfresco`	Specifies the name of the main Alfresco database user. This name is used to authenticate with the database.
`db.password=alfresco`	Specifies the password for the Alfresco database user. This password is used to authenticate with the database.

   Additional database properties may be set for further configuration. Refer to the Configuring databases for more information.
5. Specify the locations of the following external software:

Property	Description
`ooo.exe=`	Specifies the location of the LibreOffice installation.
`ooo.enabled=`	Specifies whether to use the Direct LibreOffice subsystem.
`jodconverter.officeHome=`	Specifies the location of the LibreOffice installation for JODConverter transformations. To use the JODConverter, uncomment the `ooo.enabled=false` and `jodconverter.enabled=true` properties.
`jodconverter.portNumbers=`	Specifies the port numbers used by each JODConverter processing thread. The number of process will match the number of ports.
`jodconverter.enabled=`	Specifies whether to use the JODConverter. Set the property to `jodconverter.enabled=true`.
`img.root=`	Specifies the location of the ImageMagick installation.
`swf.exe=`	Specifies the location of the SWF tools installation.

6. Configure your supported database for use with Alfresco. See Configuring databases.
7. Select a JDBC driver used with each connection type.
8. Add your global custom configurations.
9. Save your file without the `.sample` extension.

You need to restart the Alfresco server for the configuration changes to take effect.

Modifying Alfresco applications