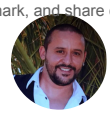


Solr Tuning - Maximizing your Solr Performance

Blog Post created by **lcabaceira** on Jun 20, 2014

Like • 2 Comment • 10

Solr Tuning Tips

Hi folks, another useful post for your alfresco related knowledge. This time dedicated to Solr tuning tips, using this information wisely can heavily contribute to increased performance on your Solr related topics such as searching and indexing content.

Solr, when properly tuned It's extremely fast, easy to use, and easy to scale. I wanted to share some lessons learned from my field experience while using Alfresco with

Solr It's a search indexer built on top of Lucene , there are two main disciplines to consider for Solr :

* **Indexing data (writing/committing)**

* **Search for data (reading/querying)**

Each of those disciplines has different characteristics and considerations while addressing performance.

It's important to mention that there is no rule of thumb that enables you to maximize your Solr performance for every project. Solr tuning is an exercise that highly depen

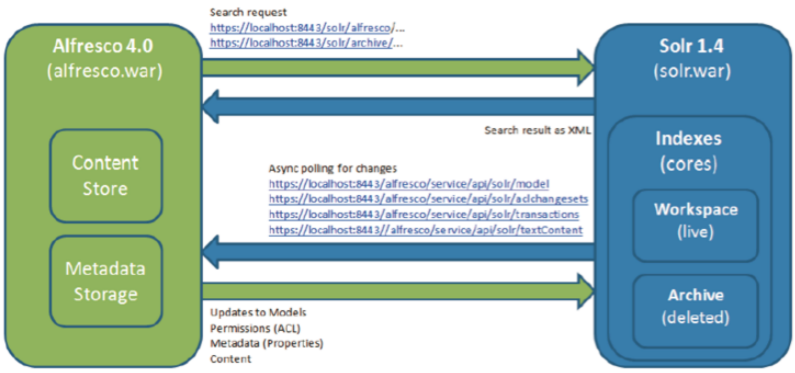
on the specific project use cases, architecture and business scenarios. Depending on the particularities of your project, the actions to perform may vary in terms of wha

needs to be done to achieve best Solr performance. This post includes procedures and methodologies that will help you to understand how Solr performance is driven.

Let's first Analise how does alfresco search and indexing works together with Alfresco

2 main Solr Cores (Live content and archived content)

By default the Solr that comes with Alfresco contains 2 cores, the live content core (**workspaces:spacesStore**) and the archived content (**archive:spacesStore**). Each cores contains the indexes for their particular set of content.



Alfresco Search (After a user searches for content what happens behind the scenes ?)

Alfresco sends a secure GET request (https) to the Solr cores and Solr responds with a streaming formatted in JSON or XML with the response to the search request.

This is then interpreted by alfresco and the results are presented in a user-friendly format.

Solr Indexing new items (tracking Requests)

This tracking occurs by default on every 15 seconds (can be configured), Solr asks alfresco for changes in content and newly created documents in order to index those on its cores. It also asks for changes on the content models and for changes on the ACLs for documents.

In summary, Solr updates its indexes by looking at the number of transactions that have been committed since it last talked to Alfresco, a bit like index tracking in a clus diagram above you see several http requests going from Solr to Alfresco, those requests are explained below:

1. **New models and model changes** https://localhost:8443/alfresco/service/api/solr/model
    1. Solr keeps track of new custom content models that have been deployed and download them to be able to index the properties in these models.
1. **ACLs changes** https://localhost:8443/alfresco/service/api/solr/aclchangesets
    1. Any changes on permission settings will also be downloaded by Solr so it can do query time permission filtering.
1. **Document Content changes**
    1. https://localhost:8443/alfresco/service/api/solr/textContent
1. **New transactions** (create, delete, update or any other action that triggers a transaction)
    1. https://localhost:8443/alfresco/service/api/solr/transactions

Brief analysis to New document indexing scenario

Let's check what happens in Solr when we create a new document and Solr executes it's tracking detecting that a new document has been created.

1. First Solr requests a list of ids of all new transactions on that document (create, update, delete, …) https://localhost:8443/alfresco/service/api/solr/transactions
1. Transactions and acl changesets are indexed in parallel, and for each transactionId, Solr requests, on this order:
    1. Document metadata
    1. Document Content

Solr Architecture variations method

There are 3 different architecture variations than can be considered while using Solr with Alfresco on a cluster. For the scope of this post i will only be addressing cluste configurations that include the following advantages:

Alfresco - > Solr search load balancing

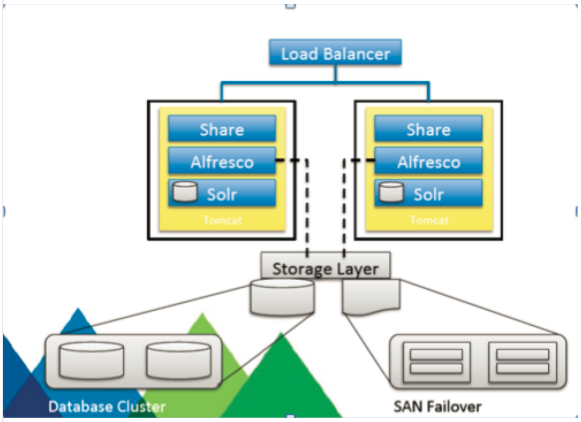
This is the most obvious use case for scalability purposes. Search requests are directed from Alfresco to a pool of Solr instances, each of which contains a full copy of t and is able to service requests in a purely stateless fashion.

Solr -> Alfresco index tracking balancing

In the other direction, Solr nodes use a load balancer to redirect their index tracking requests to one or multiple dedicated/shared Alfresco nodes. This is useful in case

indexing load, due to a heavy concurrent write/update scenario.

Option 1 - Solr in the same machine as alfresco, non dedicated trackin



On this architecture we have a solr instance deployed on the same application server of both alfresco and share web-applications.

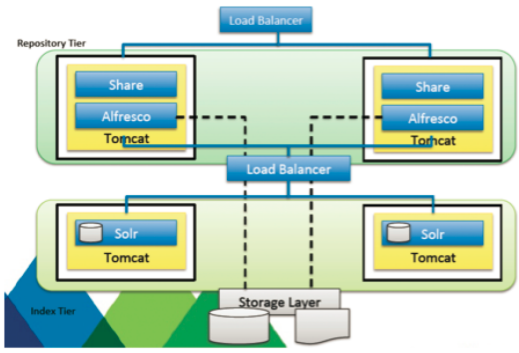
Advantages

- Easy to maintain / backup.

Disadvantages

- Shared JVM, if Solr crashes both Alfresco and Share become unavailable.
- Shared hardware, Memory is shared between all layers of the application
- When Solr downloads content, there is transformation load to generate the indexing text file (CPU/Memory intensive) on the Alfresco side, having everything o box impacts both search and indexing as all the applications are on the same application server sharing its resources like the connection pools, threads, etc.
- Only possible to scale vertically (Only possible to add more CPU and Memory)

Option 2 - Solr separated from alfresco - Non-Dedicated tracking



On this architecture variation we have the Solr instances deployed on separated machines and application servers from alfresco and share.

Advantages

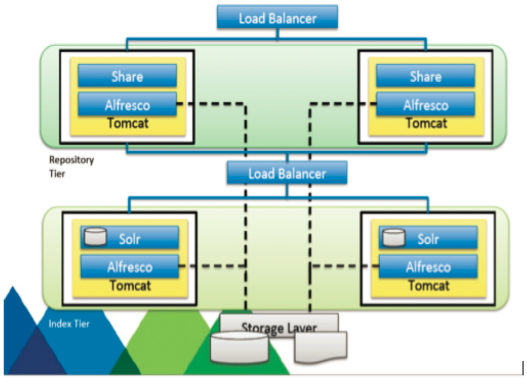
- Simple upgrade, administration and maintenance of Solr server's farm

- Allows for vertical and horizontal scalability
- Introduces the ability to load balance the queries
- Ready for Future Solr sharding feature
  - It's expected that alfresco will support, on a near future the ability to slit the index on different solr instances that will lead to increased performance, this ar ready to implement that feature.

Disadvantages

Remote Tracking can stress the network, if network problems occur solr performance gets affected.

Option 3 Solr server with a dedicated tracking Alfresco instance



On this architecture variation we use dedicated alfresco instances on the solr servers that are only used for indexing and do not receive or process any user's requests. local alfresco instances take care of important CPU/Memory intensive operations such as the transformations and the overall tracking and indexing actions. With this sc repository Tier is released from those operations resulting on a overall performance gain. This Alfresco instances are not part of the cluster and do not require ehcache configuration.

Note: When Solr requests Alfresco for the content to be indexed, it's the Alfresco server that is responsible for perform the content transformation onto a plain text file, the content is sent to Solr for indexing. This is an IO/CPU/Memory intensive operation that can decrease the overall alfresco performance.

Advantages

- Indexing operations offloaded from repository and client tier
- Dedicated Alfresco for transformation operations
  - Allow for specific transformations tuning on the index tier and on the repository tier considering the use cases. Transformation for previews and thumbnails related) and transformations for Solr indexing.
- Allows for Vertical and horizontal scalability
- General performance increase

Disadvantages

- None, in my opinion this is the best option :)

Solr Indexing Best practices

Now lets discuss the Solr indexing best practices, if your problem is regarding the indexing performance this is the juice part of the post for you.

General Indexing Golden Rules (Solr Indexing Tuning)

- Have local indexes (don't use shared folders, NFS, use Fast hardware (RAID, SSD,..)
- When using an alfresco version previous to 4.1.4 you should reduce your caches as the default caches configuration may lead to OOM when solr in under big
- Manage wisely your Ram buffer size (ramBufferSizeMB) on solrconfig.xml, this is set to 32 MB by default, but generally increasing this to 64 or even 128 has p increased performance. But this depends on the amount of free memory you might have available.
  - ramBufferSizeMB sets the amount of RAM that may be used by Solr indexing for buffering added documents and deletions before they are flushed to disk
- Tune the **mergeFactor**, **25 is ideal for indexing**, while **2 is ideal for search**. To maximize indexing performance use a mergeFactor of 25.
- During the indexing, plug in a monitoring tool (YourKit) to check the repository health during the indexing. Sometimes, during the indexing process, the repositc executes heavy and IO/CPU/Memory intensive operations like transformation of content to text in order to send it to Solr for indexing. This can become a bottle for example the transformations are not working properly or the GC cycles are taking a lot of time.
- Monitor closely the JVM health of both Solr and Alfresco (GC, Heap usage)
- Solr operations are memory intensive so tuning the Garbage collector is an important step to achieve good performance.
- Consider if you really need tracking to happen every 15 seconds (default). This can be configured in Solr configuration files on the Cron frequency property. **alfresco.cron=0/15 * * * * ? ***
- This property can heavily affect performance, for example during bulk injection of documents or during a lucene to solr migration. You can change this to 30 se more when you are re-indexing. This will allow more time for the indexing threads to perform their actions before they get more work on their queue.
- Increase your index batch counts to get more results on your indexing webscript on the repository side. In each core **solrcore.properties**, raise the batch cour more **alfresco.batch.count=2000**
- In **solrconfig.xml** of each core configure the ramBufferSize to be at least 64 Mb , you can even use 128 if you have enough memory **. <ramBufferSizeMB>64</ramBufferSizeMB>**
- In **solrconfig.xml** of each core configure the mergeFactor to 25, this is the ideal value for indexing. **<mergeFactor>25</mergeFactor>**
- Disable Full Text Indexing on archive:SpacesStore Solr, this is done by adding the property alfresco.index.transformContent=false. Alfresco never searches for inside files that are deleted/archived. This saves on disk space, memory on Solr, Cpu during Indexing and overall resources.
- Tune the transformations that occur on the repository side, set a transformation timeout.
- Important must reply project questions:
  - SSL really needed? If inside the intranet, you should disabled to reduce complexity.
  - Full Text indexing really necessary, some customers do full text index but they don't actually use it.

- Is an archive core really necessary for indexing, if you are not making use of this indexing core, it would be beneficial to disable it.

For index updates, Solr relies on fast bulk reads and writes. One way to satisfy these requirements is to ensure that a **large disk cache** is available. Use local indexes fastest disks possible. In a nutshell, you want to have enough memory available in the OS disk cache so that the important parts of your index, or ideally your entire inde into the cache. Let's say that you have a Solr index size of 8GB. If your OS, Solr's Java heap, and all other running programs require 4GB of memory, then an **ideal** mer for that server is at least 12GB. You *might* be able to make it work with 8GB total memory (leaving 4GB for disk cache), but that also might NOT be enough.

Solr Indexing Troubleshooting techniques

Troubleshooting Solr indexing performance means finding the bottleneck that is delaying the overall indexing process. Since this is a process that involves at least 2 lay application architecture, the best way to troubleshoot is trough a dedicated series of tests measuring performance and comparing results.

First thing to discover is where is the bottleneck occurring, it can be on:

- **Repository layer**
  - **Database** - If it's a database performance issue, normally adding more connections to the connection pool will increase performance.
  - **I/O** - If it's a IO problem, it can normally occur when using virtualized environments, you should use hdparam to check read/write disk speed performance i running on a linux based system, there are also some variations for windows. Find the example below:

    sudo hdparm -Tt /dev/sda

```
Timing cached reads: 12540 MB in 2.00 seconds = 6277.67 MB/sec

Timing buffered disk reads: 234 MB in 3.00 seconds = 77.98 MB/sec
```

- **JVM** – Jvm configuration can impact the performance on the repository layer indexing activities.
- **Cpu and Memory usage** – monitor the usage of the CPU and Memory on this layer and check for unusual usage of this two components.
- **Tranformations** – Set a timeout for the transformations that occur on the repository layer. There is no timeout set by default and sometimes, when there's a transformation issue the threads are frozen waiting for the transformations to occur.
- **SOLR Indexing layer**
  - **Number of threads for indexing** – You can add more threads to the indexing processes if you detect that indexing is slow on the Solr side.
  - **Solr caches -** There are several caches that you can configure to increase indexing performance.
  - **JVM** – Jvm configuration can impact the performance on the Solr layer indexing activities. Focus your efforts on analyzing and tuning the Garbage collecto big GC pauses by analyzing the gc logs.
  - **Hardware scalability** – If none of the above actions improve your performance you may need to increase memory and CPU power on the Solr layer. Also horizontal scaling when appropriate.

The rule for troubleshooting involves testing and measuring initial performance, apply some tuning and parameter changes and retest and measure again until we reach necessary performance. I strongly advice you to plugin a profiling tool such as Yourkit to both the repository and Solr servers to help with the troubleshooting.

Solr Search Best practices

This section is about tuning the search performance while using Solr, in general it will be sufficient to follow the golden rules below, if applying those does not solve your

you might need to scale your architecture.

General Search Golden Rules

- Use local folders for the indexes (don't use shared folders, NFS)
- Use Fast hardware (RAID, SSD,..)
- Tune the mergeFactor, a mergeFactor of 2 is ideal for search.
- Decrease the Solr caches, specially when running an Alfresco version prior to 4.1.4.
- Increase your query caches and the RAMBuffer.
- Avoid path search queries, those are know to be slow.
- Avoid using sort, you can sort your results on the client side using js or any client side framework of your choice.
- Avoid * search, avoid ALL search
- Tune your Garbage collector policies and JVM memory settings.
- Consider lowering your memory on the JVM if the total heap that you are assigning is not being used. Big JVM heap sizes lead to bigger GC pauses.
- Get the fastest CPU you can, search is CPU intensive rather then RAM intensive.
- Separate search and indexing tiers. If you can have 2 separate solr server farms, you can dedicate one to the indexing and the other to search. This will increa global performance (Only available since alfresco 4.2.X)
- Optimize your ACL policy, re-use your permissions, use inherit and use groups. Don't setup specific permissions for users or groups at a folder level. Try to re-u Acls.
- Upgrade your Alfresco release with the latest service packs and hotfixes. Those contain the latest Solr improvements and bug fixes that can have great impact overall search performance.
- Make sure you are using only one transformation subsystem. Check the alfresco-global.properties and see if you are using either OooDirect or JodConverter, r enable both sub-systems.

Typical issues with Searching

It can happen that you are searching and indexing on the same time, this causes concurrent accesses to the indexes and that is known to cause performance issues. T some workarounds for this situation. To start you should Plugin a profiler and search for commit Issues (I/O locks), this will allow you to check if you are facing this probl

Solr Search Troubleshooting techniques

To troubleshoot your Solr search problems you should start by choosing a load testing tool such as SolrMeter or Jmeter and design your testing scenario with one of thc You can also choose to use the default alfresco benchmark scenario. The second step is to attach a profiler like Yourkit or other java profiler of your choice and records performances snapshots for analysis.

Apply the tunings suggested on this document (specially on the golden rule section) and retest until you reach the necessary performance.

Solr usage in Share

If your project relies on the share client offered by Alfresco, you should know that tuning your Solr indexing and search performance will affect positively the overall shar performance.

Share relies on Solr in the following situations:

- Full Text Search (search field in top right corner)
- Advanced Search
- Filters
- Tags
- Categories (implemented as facets)
- Dashlets such as the Recently Modified Documents
- Wildcard searches for People, Groups, Sites (uses database search if not wildcard)

Overall Best Practices Technical Details

This section contain important technical details that will allow you to implement the various best practices mentioned previously on this post.

1 - Turn on Logging During Search

If you want to have a look at the queries that Alfresco is running against Solr when you click around in Alfresco Share. You can enable debug logging as follows in **log4j.properties**.

log4j.logger.org.alfresco.repo.search.impl.solr.SolrQueryHTTPClient=debug

A log for a full text search on “Alfresco” looks like this:

2014-01-17 08:21:15,696 DEBUG [impl.solr.SolrQueryHTTPClient] [http-8080-26] Sent :/solr/alfresco/afts?
q=%28%28PATH%3A%22%2Fapp%3Acompany_home%2Fst%3Asites%2Fcm%3Atest2%2F*%2F%2F*%22+AND+%28Alfresco++AND+%28%2BTYPE%3A%22cm%
%22+%2BTYPE%3A%22cm%3Afolder%22%29%29+%29+AND+-TYPE%3A%22cm%3Athumbnail%22+AND+-TYPE%3A%22cm%3AfailedThumbnail%22+AND+-
TYPE%3A%22cm%3Arating%22%29+AND+NOT+ASPECT%3A%22sys%3Ahidden%22&wt=json&fl=*%2Cscore&rows=502&df=keywords&start=0&locale=en_GB&fq=
ts%7DAUTHORITY_FILTER_FROM_JSON&fq=%7B%21afts%7DTENANT_FILTER_FROM_JSON

How to disable SSL communication between Solr and Alfresco

By default, the communication between Solr and Alfresco is encrypted, if you don’t need this encryption it’s a good idea to disable this in order to reduce complexity that contribute to increased performance.

On the Alfresco server, edit the alfresco-global.properties and set:

- solr.secureComms=none
- On the alfresco webapp deployment descriptor web.xml, comment out the security constraint.

<!-â€-

```
<security-â€constraint>

<web-â€resource-â€collection>

<url-â€pattern>/service/api/solr/*</url-â€pattern>

</web-â€resource-â€collection>

<auth-â€constraint>

<role-â€name>repoclient</role-â€name>

</auth-â€constraint>

<user-â€data-â€constraint>

<transport-â€guarantee>CONFIDENTIAL</transport-â€guarantee&gt

</user-â€data-â€constraint>

</security-â€constraint>

<login-config>

<auth-method>CLIENT-CERT</auth-method>

<realm-name>Repository</realm-name>

</login-config>

<security-role>

<role-name>repoclient</role-name>

</security-role>



-->
```

- For every Solr core that you have configured set alfresco.secureComms=none on the solcore.properties file.

- On the alfresco Solr deployment descriptor web.xml or solr.xml under Catalina/conf/localhost/solr.xml, comment out the security constraint as previously show

Detailed information can be found in the Alfresco customer Portal

How to set a transformation Timeout on Alfresco

To set a timeout limit (that it's not set by default) can help you with your tuning and troubleshooting activities.

Timeout (ms) Use this limit to set timeout on reading data from the source file to be transformed. This limit works with transformers that don't bulk read their source dat enforced by a modified InputStream that either throws an exception or returns an End of file (EOF) early. The property associated with this transformation limit is timeou

You can set this property on your alfresco-global.properties as the following example:

content.transformer.default.timeoutMs=180000

How to set transformation limits on Alfresco

Setting appropriate transformation limits can help you to fine-tune your transformations and to improve indexing performance.

In Alfresco 4.2d much of the configuration of transformers is done using Alfresco global properties. In the case of the Enterprise edition these may be changed dynamic without stopping the server. Prior to this it was possible to control Content Transformer limits to a more limited extent using Spring XML and a few linked Alfresco global

You can find detailed information on transformation limits at http://wiki.alfresco.com/wiki/Content_Transformation_Limits#Introduction

- How to Rebuild Solr Indexes

One useful action that is sometimes required is to rebuild the indexes from scratch. In order to rebuild the solr indexes, proceed as follows :

Do as follows:

1. Stop Tomcat that runs Solr web application
2. Remove index data of archive core at alf_data/solr/archive/SpacesStore
3. Remove index data of workspace core at alf_data/solr/workspace/SpacesStore
4. Remove cached content model info of archive core at alf_data/solr/archive-SpacesStore/alfrescoModels/*
5. Remove cached content model info of workspace core at alf_data/solr/workspace-SpacesStore/alfrescoModels/*
6. Restart Tomcat that runs Solr web application
7. Wait a bit for Solr to start catching up...

Note : index.recovery.mode=FULL is not used by Solr - only by Lucene

About Sizing on the Solr servers

Sizing your Solr servers depends a lot on the specific search requirements of your project. The important factors you need to consider are :

- Search Ratio, to get the search ratio you should divide the typical usage of the system in Read/Write/Search. Start with 100% and give a % to each of the oper
- Number of Documents in the repository
- Group Hierarchy
- Number of Acls
- Amount of Cpu Cores you have available (the more the better :))

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

Alfresco provides a formula that helps you to calculate the ammount of memory needed on your Solr servers, check the following url for guidance. http://docs.alfresco.com/community/concepts/solrnodes-memory.html

Below you find an excell file that will help you with the calculations (you need to rename the extension from .txt to xlsx)

Calculate_Memory_Solr Beta 0.2_xlsx

I hope you enjoyed this post, i've surely enjoyed writing it and i hope it can help you with your projects. More interesting posts from my field experience are coming to th stay tuned.

'The greatest ideas are opensource, together we are Stronger'

ATTACHMENTS

Screen-Shot-2014-06-10-at-22.50.49.png 294.7 KB
Screen-Shot-2014-06-10-at-22.47.36.png 249.1 KB
Screen-Shot-2014-06-10-at-22.44.49.png 204.0 KB
solr_commns.png 320.9 KB

Visibility: Luis Cabaceira's Blog • **1300 Views**

Last modified on Sep 26, 2016 8:27 PM

Tags: blog_lcabaceira

0

10 Comments

blog_commenter
Aug 15, 2014 12:32 PM

Great blog Luis. What is the largest size of index you have seen and how long did it take to create it. We are indexing a large amount of content and the index is tak long time to build but there don't seem to be any performance statistics anywhere for comparison.

Obviously I know performance is related to the specification of the machine and the configuration, but having a graph that your could extrapolate from would be rea handy.

Actions

blog_commenter
Sep 3, 2014 12:07 PM

Hi Luis,

very interesting article.

For the option 3 scenario 'Option 3 Solr server with a dedicated tracking Alfresco instance', do you know if it's something possible with the community version?

regards

Actions

lcabaceira

Sep 30, 2014 10:26 PM

Hi Steve, i don't see a reason why not. I've never tried it, but it should be straight forward.

Just install alfresco on the same server as Solr pointing to the same database and content store of your user facing alfresco node(s).

Actions

blog_commenter

Oct 28, 2014 6:17 PM

Hi Luis,

Very usefull article.

Can you tell me what is the best way to split the search and indexing tiers? I'm on the option 3 scenario “Option 3 Solr server with a dedicated tracking Alfresco inst and i would like to move to a better layout.

Thanks

Actions

blog_commenter

Nov 24, 2014 7:48 AM

Thanks for posting such a useful blog..

I've one doubt to you..

For our project client is using 4.0.e community version with MySql database. Now the requirement is

Actually here they are searching for records more than 1000 records.. while doing it taking time of 25-30sec approximately, but client asking for 5-10 secs. How to a that, can you please suggest me in detail.

System configurations:

OS: Windows server 2008 R2,

RAM : 32GB

Alfresco Version: 4.0.e Community

Actions

blog_commenter
Dec 10, 2014 4:19 PM

Thanks for your posts Carlos.

In our use case, we are using Alfresco 4.2 without any versioning, content is read-only, no full text search, and ACLs are static. Therefore, SOLR will be used to inc metadata, and the metadata will change from time-to-time. A custom app fronts Alfresco, the Share App will not be used - therefore no thumbnails or previews are needed.

My question about this use case is whether Alfresco will be performing any 'content transformation onto a plain text file' when there is no full text search, no version thumbnails/previews, ACL changes...I can't see that I have much of a transformation load, if any, on my Alfresco box.

If I am reading this right, Option 1 would be sufficient, or should I say, 'sufficient for a Release 1.0 of our infrastructure'.

Actions

blog_commenter
Apr 8, 2015 6:54 PM

Excellent article. We are using alfresco5+solr4. One of our typical usecase is to search for that content soon after its created. Solr4 currently takes anywhere betwe 60 seconds to index the new transaction. (even if there is no activity on the system). We have tried adjusting the tracker to run every5 seconds

alfresco.cron=0/5 * * * * ? *

increased ram,

1024

increased merge factor,

25

20

none of the seems to help. Any idea of how to reduce the time lag to less than 5-10 seconds ?

Actions

blog_commenter
May 13, 2016 10:33 PM

I was recommended this blog via my cousin. I'm no longer sure

whether this put up is written via him as nobody

else know such special about my problem. You are amazing!

Thanks!

Actions

blog_commenter
Jun 24, 2016 5:30 AM

Just simply wanted to emphasize I am lucky I stumbled in your web page!.

Actions

blog_commenter
Jun 30, 2016 5:12 AM

It is a nice article, thanks for sharing the information. We also provide SOLR Online training.Tekslate for indepth SOLR training

Actions

Related Content

- Alfresco Best Practices
- Monitoring your Alfresco solution
- Troubleshooting Alfresco Content Services Performance from the Bottom Up
- Using SSL with Alfresco Search Services and Solr 6
- Sharding with Alfresco Search Services and Solr 6

Recommended Content

- Database Configuration
- How to customize ActivityBehaviorFactory
- Configure reverse proxy for alfresco community 5.2 windows
- How to update file/folders metadata using Public Java API Services?
- Overriding Share

Incoming Links

- Alfresco Best Practices