Case Study: Improving Laserfiche Search Performance

White Paper

April 2010

Laserfiche®

Table of Contents

Background

This paper describes steps taken to analyze and improve search performance at a particular Laserfiche customer's system. We present this summary in order to help other customers who want to optimize their search performance.

The Case

The City of Boulder, Colorado uses Laserfiche to manage information throughout various City departments. Laserfiche Solutions Group (LSG) worked with reseller S. Corporation, Inc., to provide an analysis of the City of Boulder's existing Laserfiche system and make best practice recommendations related to the performance of searching in the City's Laserfiche system.

Research for this analysis was primarily gathered through:

- Search performance tests in the City's production Laserfiche system.
- Performance counters and audit logs taken in the City's production Laserfiche system.
- Interviews with key staff members.
- Historical search performance tests conducted by the city's IT staff.
- Search performance tests in a test environment at Laserfiche.

Laserfiche Search Architecture

Searching in Laserfiche has several elements, which are handled separately by different server components. An understanding of these pieces and how they relate informs our approach to troubleshooting and best practices.

- **Laserfiche Full-Text Indexing and Search Service (LFFTS):** An independent service that handles full-text searching of the repository.
- **SQL Search:** Retrieves searches for things other than full-text, such as names, dates, field values, annotation, tags, etc., that are stored in the SQL database.
- **Laserfiche Server** The Laserfiche Server merges the search results, determines which columns to display, and checks that the user has the rights to the data being returned.

Findings and Solutions

Analysis of the City of Boulder's system identified several factors that reduced search performance. Addressing them improved the performance in a test system and, when they were applied, in the City's system. Note that some of these solutions were specific to this installation; see the General Recommendations section for a general guide to analyzing and improving your own organization's search performance.

Optimize Search Types

Laserfiche Audit Trail was used to monitor searches and their duration. Searches using overly broad criteria may return more search results than needed and slow down the results of others' searches.

Searches can be directly specified by individual users or aided by integrations and customizations that use Laserfiche advanced search syntax. As much as possible, users should be trained to conduct effective searches, which will help them better find the information they are looking for as well as reducing the load on the system.

In the case of the City of Boulder, the following recommendations were made:

- When conducting a full-text search, specifying more detail will return more relevant results in a shorter time.

  > **Example:** A full-text search for *canyon* in the City's repository returns more than 26,000 results. The full-text search for *canyon creek* returns 2,117 results, a full-text search for *canyon creek master plan* returns 74 results, and a search for *fourmile canyon creek master plan* returns 54 results. The more specific the search, the faster and more relevant the results.

- An "AND" Search will return fewer and more specific results than an "OR" search.

  > **Example:** A Weblink search for *Boulder Meadows* timed out because it was searching for all documents that contained either word, whereas the user may have intended to search for items that contained the exact phrase *"Boulder Meadows"*.

- For field searches, searching within a single field is much faster than searching across all fields.

  > **Example:** In the case of searching for documents associated with a case number, searching in a

Laserfiche test system for an exact match within a single field took 0.026 seconds, compared to 0.072 seconds for searching for an exact match across all fields. Searching for a partial match across all fields took even longer: 15.097 seconds.

Adjust Customizations to Use the Most Efficient Search Syntax

In addition to searches constructed by individual users, the City of Boulder has several custom integrations that incorporate Laserfiche search syntax. Integrations and other customizations should be adjusted to make sure their code reflects the suggestions provided above. Any other operations that also use advanced search syntax rather than the search interface should also follow these recommendations.

Example 1: The City of Boulder has a hotkey integration that allows users to access documents in the Laserfiche repository from the LandLink case and parcel windows by querying the "case number" or "parcel number" that are captured from the screen. As much as possible, this integration should avoid using too many "OR" operators in full-text searches. When searching using the exact case number or parcel number, the queries should use the simplest and fastest option—an exact match in a single field.

Adjust Customizations to Display Results Efficiently

Optimizing customizations to use the most efficient methods of retrieving and displaying search results can improve the speed of many searches, thereby potentially improving performance throughout the repository.

Example 2: The City also has a custom integration called ECM Integrator, which is a server application designed to provide ECM (Electronic Content Management) batch functionality. The ECM integrator uses the Laserfiche Server Objects (LFSO) and DocumentProccessor libraries. This integration can be adjusted to use the following method:

```
LFSearchResultListing
GetSearchResultListing(LFSearchListingParams_SearchListingPar
ams, int RowsToPreload);
```

This code should help the searches run faster, since retrieving the field data from the SearchResultListing object should be more efficient than the previous method of accessing the LFFieldData object in this context. It also specifies which columns to display, which should reduce the time needed to display the search results.

Defragment Search Index

On September 6, 2009, the fragmentation level of the City of Boulder's's production search catalog was 18.42%. After reindexing using the Quick Reindex Utility, the fragmentation level dropped to zero. The average full-text search duration (for search phrases selected for test purposes) dropped from 13 seconds to 5 seconds, showing performance 2.4 times as fast. The search catalog will automatically undergo an optimization procedure if it reaches a certain level of fragmentation, but you can also reindex manually if you are finding that searching is slower than expected.

Use Faster CPU for SQL Server

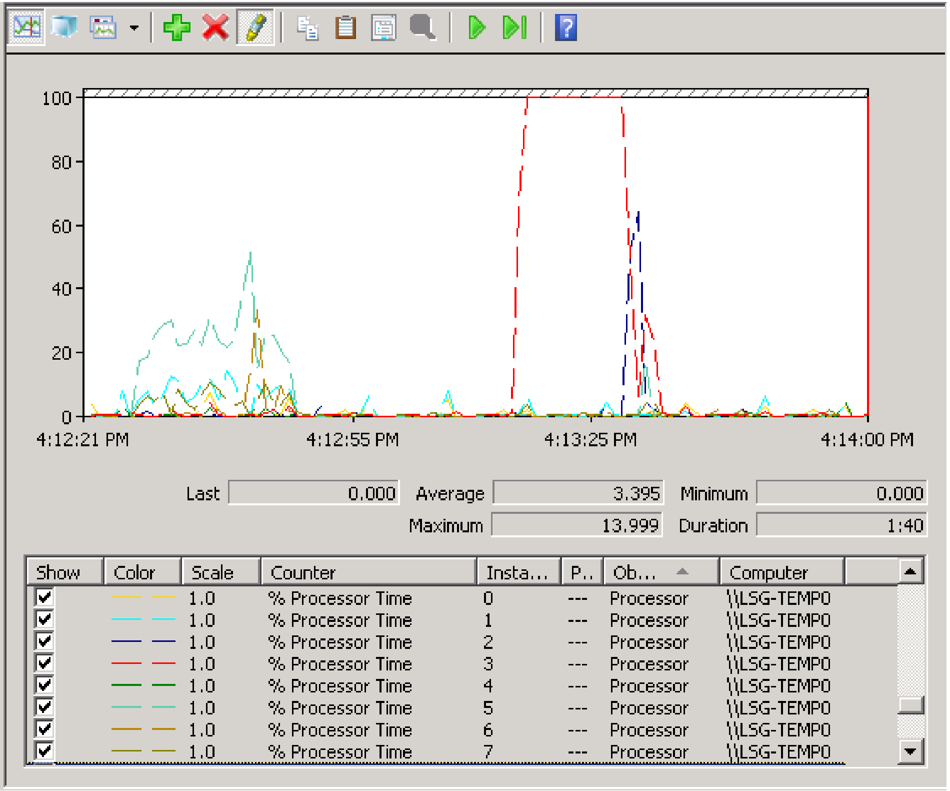
During the full-text search in the City's production environment, the SQL Server %Processor Time per CPU was less than 50%; during the field search, %Processor Time per CPU surged up to 100% for all eight CPU cores, indicating a CPU bottleneck on the SQL Server during field searches.

The same measurement was also taken in the Laserfiche test environment, where the SQL Server had equivalent hardware specifications for RAM and disk compared to the City's production SQL Server. The only difference between the two SQL Servers was the speed and type of CPU.

	City's Production SQL Server	**Laserfiche Test SQL Server**
Processor Name	Intel Xeon E5405	Intel Xeon x5365
Frequency	2GHz	3GHz
# of Processors	2	2
# of Cores/processor	4	4

We see that on the Laserfiche test SQL Server, during the field search, %Processor Time per CPU went up to 100% for only one CPU core, but stayed below 65% for the other seven CPU cores. Compared to the City's production SQL Server, the Laserfiche test SQL Server was much less constrained by the CPU bottleneck, thanks to the faster processors.

In fact, a comparison of search performance test results conducted in the City's production system and the Laserfiche test system showed that field searches were much faster in the Laserfiche test system than in the City's production system; the average field search duration in the Laserfiche test system was half of that in the City's production system. It is recommended that the City use a higher frequency CPU for their SQL server.



Last 0.000 Average 3.395 Minimum 0.000

Maximum 13.999 Duration 1:40

Show	Color	Scale	Counter	Insta...	P..	Ob...	Computer
☑	— —	1.0	% Processor Time	0	---	Processor	\\LSG-TEMP0
☑	— —	1.0	% Processor Time	1	---	Processor	\\LSG-TEMP0
☑	— —	1.0	% Processor Time	2	---	Processor	\\LSG-TEMP0
☑	— —	1.0	% Processor Time	3	---	Processor	\\LSG-TEMP0
☑	— —	1.0	% Processor Time	4	---	Processor	\\LSG-TEMP0
☑	— —	1.0	% Processor Time	5	---	Processor	\\LSG-TEMP0
☑	— —	1.0	% Processor Time	6	---	Processor	\\LSG-TEMP0
☑	— —	1.0	% Processor Time	7	---	Processor	\\LSG-TEMP0

General Recommendations

Laserfiche provided the following suggestions for a maintenance and troubleshooting plan to improve search performance. These steps helped identify the solutions described above, and have been generally shown to improve the speed of searching in Laserfiche.

Optimize Laserfiche Client Settings for Search Performance

Laserfiche Client settings can be customized according to a variety of needs. Of course, given that a particular organization or user may have multiple needs, these may have to be balanced. If the goal is the fastest possible search performance, the following settings are recommended, based on testing conducted by Laserfiche:

Make Searches Faster

- Consider adjusting the **Stop Words** list in **Indexing Properties** in the Laserfiche Administration Console to exclude frequently searched words that do not add meaning when a full-text search is performed.
- In Laserfiche 8.1.1 or higher, optimization was introduced to make searching in string fields significantly faster than before. Therefore, the best practice is to define string fields to be shorter than 100 characters whenever possible.

  Example: In a Laserfiche test environment, exact-match cross-field searches with short search phrases improved from taking 0.11 seconds to only 0.02 seconds (an 80% improvement) after the optimization.

- Disable the **Fuzzy Search** option by default, and enable it only when necessary. A quick way to find out if anyone has this option enabled is to execute the following SELECT query in the SQL database.

  ```
  select * from trustee_attr where attr_name = '[SEARCH]Option'
  and attr_val not like '0'
  ```

  If you do use fuzzy search, note that the greater the level of fuzziness, the more results will be returned, and the less likely they are to be relevant to the query. Setting the fuzziness to a lower percentage or smaller number of letters will return fewer, more relevant results that can be displayed faster.

- Disable the **Find partial matches when performing a basic search** option.

- Disable the **Include Shortcuts in Search Results** option. A quick way to find out if anyone has this option enabled is to execute the following SELECT query in the SQL database.

  ```
  select * from trustee_attr where attr_name =
  '[SEARCH]IncludeShortcuts'
  and attr_val like 'Yes'
  ```

Have Results Returned Faster

- Give the **Bypass Filter Expressions** privilege to the **Everyone** Group if no filter expression is used in the repository.
- Give **Bypass Browse** privilege to administrators who should have access to all entries in the repository or already have the **Manage Entry Access** privilege.
- Make entry access security configuration as simple as possible.
- Do not display resource-intensive columns. As of Laserfiche 8.1.1, these are **Path** and **Security**. These settings are stored in the system as user attributes and apply to both the thick and the Web clients. They can be changed by the individual users or, in version 8.0 and above, centrally managed by an administrator.

Design a System for Best Search Performance

It is a best practice to place the full-text search catalog on the fastest hard drive possible, and separate from the drive where Laserfiche volumes are stored.

The City of Boulder's current system matches the recommended configuration. They do plan to move toward iSCSI storage, an Internet Protocal-based storage networking standard for linking data storage facilities, in the future. In this case, it is recommended to keep the full-text search catalog local to the Laserfiche Server and only move the volumes to the iSCSI storage. Moving the volumes to non-local hard drives may affect the speed of indexing, which will make it even more important to perform the indexing after hours, as described below.

Perform Full-Text Indexing at a Convenient Time

Indexing can be a resource-intensive process. A best practice is to avoid full-text indexing while searching, as much as possible. For example:

- **Plan heavy scanning and document creation during non-production hours.**
- **Scheduled indexing**: Scheduled indexing is a new feature offered in Laserfiche version 8.1.1 or higher. It allows administrators to schedule indexing to exclude certain peak business hours.

Monitor the System for Heavy Load

You can monitor key performance counters on the servers and make sure they do not deviate from the baseline.

- **Check the Indexing Load:** You can determine if indexing is the cause of slow full-text searches by checking the index queue size in the Laserfiche Administration Console as well as the server performance counters related to indexing.
- **Monitor Concurrent Search Load**: A best practice is to periodically monitor the number of concurrent searches using the **Concurrent Searches** counter. If the number is frequently high, it may be time to perform a search performance benchmark test with multiple concurrent searches and analyze performance bottlenecks in such conditions.
- **Monitor Load from Other Applications Hosted on the Same SQL Server:** If the production SQL server also hosts other databases than the Laserfiche production database, activities in all the hosted databases can be checked at random times in the Activity Monitor in SQL Server Management Studio. If the load becomes too heavy, a dedicated SQL server should be used to host the Laserfiche database.
- **Check for Busy Network Traffic:** Compare the network performance counters to the **Current Bandwidth** counter for the Laserfiche, SQL, and Web servers to see if there is a network bottleneck.

Currently, the City of Boulder is not showing symptoms of heavy load on their system. As a best practice, it is recommended that they continue to monitor the system, and take advantage of the scheduled indexing feature in Laserfiche 8.1.1.

Perform System Maintenance

Monitoring and periodically defragmenting key components of the system can improve search performance.

- **Check SQL indexes:** Rebuild if fragmentation is greater than 40%. Reorganize if fragmentation is between 10% and 40%.
- **Monitor key SQL tables in the Laserfiche database:** Monitor tables such as *toc*, *doc*, and *propval*. Rebuild if necessary.
- **Monitor full-text search performance:** After reindexing, the search performance should be regularly monitored.
- **Remove full-text search index fragmentation:** As described above, search index fragmentation was a factor in the search performance at the City of Boulder. One tool available to perform this defragmentation is the Laserfiche 8 Quick Reindex utility (QRcmd.exe), a free component

that can be installed with Laserfiche Server 8. It quickly regenerates the search index files for a Laserfiche repository.

Troubleshoot Hardware Performance if Necessary

In most cases of reduced search performance, after checking the items discussed above, we should have a good idea of the cause of the slowdown. If necessary, we can troubleshoot on the hardware resources on the SQL Server and Laserfiche Server. This step is not recommended at the beginning of the investigation. Higher level issues, as listed above, should be exhausted first. For example, if there is a missing index in the database, it may cause the SQL Server to run full-table scans and appear to have an Input/Output (I/O) bottleneck.

If it is suspected to be a SQL Server performance issue, there is a good troubleshooting guide, *Troubleshooting Performance Problems in SQL Server 2005,* at http://technet.microsoft.com/en-us/library/cc966540.aspx. Most of the same performance counters and analysis used in troubleshooting SQL Server performance can also be applied to the identifying Laserfiche Server performance bottlenecks.

We followed the steps described in the above reference, using a specific search example ("canyon creek" as both a full-text and a partial match, all-fields search) in the City's production system. We tested the Laserfiche Server, the SQL Server, and the I/O subsystem.

As described above, the conclusion of analysis was that there is a CPU bottleneck on the SQL Server in the City's production system. Search performance can be improved by going with a higher frequency CPU on the SQL Server. In the future, the City can continue to follow this process to troubleshoot any additional hardware issues.

Keep Up to Date on Laserfiche Releases and Fixes

Laserfiche periodically releases Server hotfixes and version updates that include performance optimizations. It is recommended to periodically check the Laserfiche Support site at https://support.laserfiche.com for announcements of product hotfixes and releases.

Laserfiche®

Case Study: Improving Laserfiche Search Performance
April 2010

Authors: Fei Wang and Snow Tempest
Editor: Tammy Kaehler
Technical Editors: Jeff Huang, Jeanie Conner, and Gordon Wong

Laserfiche
3545 Long Beach Blvd.
Long Beach, CA 90807
U.S.A

Phone: +1.562.988.1688
www.laserfiche.com

Laserfiche is a trademark of Compulink Management Center, Inc. Various product and service names references herein may be trademarks of Compulink Management Center, Inc. All other products and service names mentioned may be trademarks of their respective owners.

Laserfiche makes every effort to ensure the accuracy of these contents at the time of publication. They are for information purposes only and Laserfiche makes no warranties, express or implied, as to the information herein.