



IBM® Watson™

## IBM Watson Explorer Content Analytics Version 11.0 Performance Tuning Guide

This document describes typical methods for performance tuning an IBM® Watson™ Explorer Content Analytics V11.0 system, and includes an explanation of the system architecture and resource requirements.

### **Disclaimer**

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Anyone attempting to adapt these techniques to their own environment does so at their own risk.

Performance data contained in this document were determined in various controlled laboratory environments and are for reference purposes only. Customers should not adapt these performance numbers to their own environments as system performance standards. The results that may be obtained in other operating environments may vary significantly. Users of this document should verify the applicable data for their specific environment.

**Note:** Before using this information and the product that it supports, be sure to read the general information under Notices.

**Edition Notice**

This edition applies to version 11, release 0, modification 0 of IBM Watson Explorer Content Analytics (product number 5725-I17) and to all subsequent releases and modifications until otherwise indicated in new editions.

© **Copyright International Business Machines Corporation 2015**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.



# Contents

Introduction.....	1
System architecture .....	2
System components .....	2
Crawler.....	3
Indexer .....	4
Document processor.....	8
Search server .....	9
Server configuration options.....	11
Tuning parameters.....	13
Conventions .....	13
Setting the maximum heap size .....	14
Setting the number of indexer threads .....	17
Setting the number of document processor threads for each server .....	18
Setting the document length limit .....	19
Setting the index buffer size.....	19
Adjusting the index commit interval.....	19
Setting the taxonomy cache type.....	20
Setting the number of search cache entries .....	21
Setting the number of threads for search servers .....	24
Conclusion .....	24
Notices.....	25
Trademarks .....	27



# Introduction

This document describes typical methods for performance tuning an IBM® Watson™ Explorer Content Analytics V11.0 system, and includes an explanation of the system architecture and resource requirements. To optimize your system, you need to understand its architecture and consumption of computational resources. This document provides step-by-step instructions for setting performance tuning parameters.

Determining tuning parameters for an expected usage scenario and workload is always a challenge because the effects on the performance of hardware resources, server configurations, and tuning parameters are related to each other. In addition, the optimal hardware configuration and tuning parameters often vary depending on the situation. Because this document covers only specific conditions and situations that we can predict, it is important that you carefully make your own decision for your situation at your own risk.

This document includes the following sections:

“System architecture” on page 2 explains the architecture of Watson Explorer Content Analytics so that you can better understand the performance characteristics of the system. This section includes descriptions of the basic components and server configuration options of Watson Explorer Content Analytics. The performance characteristics and memory requirements are described for each component.

“Tuning parameters” on page 13 describes the major tuning parameters and provides instructions about how to set these parameters.

## System architecture

The architecture of Watson Explorer Content Analytics is described in this section to help you understand the performance characteristics of the system. This section includes descriptions of the basic components and server configuration options. When you run the installation program, you select the server configuration option to determine the server roles and component locations.

### System components

Watson Explorer Content Analytics includes the following components: crawlers, indexers, document processors, and search servers. Watson Explorer Content Analytics supports multiple index servers, if the servers are installed as a shared disk configuration. The location of these components is based on the server configuration that you select at installation time:

- If you select the “All on one server” configuration, all components are initially created on a single, master server. The crawler components are restricted to the master server, but you can add an unlimited number of index servers (on a shared disk installation), search servers, and document processor servers as needed.
- If you select the “Distributed server” configuration, an indexer is created on the master server, and the crawler components and a search server are created on separate servers. All crawler components are restricted to the crawler server, but you can add an unlimited number of index servers (on a shared disk installation), search servers, and document processor servers as needed.

For more information about server configurations, see “Server configuration options” on page 11.

As shown in Figure 1, documents to be indexed are retrieved from the crawl spaces that are specified in the crawler configuration settings. The crawlers retrieve the documents that are found in the crawl spaces and send the documents to the indexers. The indexers extract linguistic information from these documents by using document processors and build the indexes from the extracted information. Search servers use the created indexes to provide search capabilities to users.

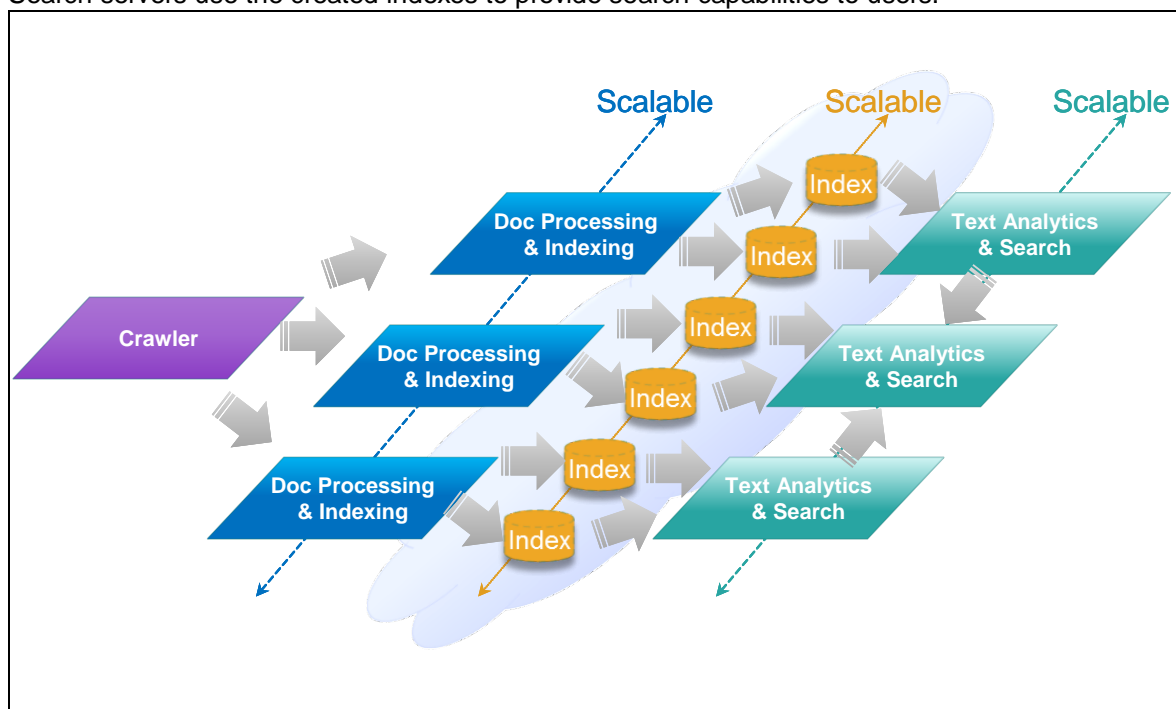


Figure 1. System Components

A set of documents that is defined by a crawl space is called a *collection*. Basically, the purpose of the system is to make the collection analyzable. Each collection has its own crawlers, indexers, document processors, and search servers. The documents in a collection are indexed into a directory that corresponds with the collection. Because separate indexes are created for each collection, creating additional collections increases the consumption of memory, disk space, and CPU time – factors that need to be taken into account when doing capacity planning.

The following sections describe characteristics that are related to the performance and resource consumption of each system component.

## Crawler

The crawler component retrieves documents from data sources and ingests the documents into the indexers. Data sources provide self-descriptive responses to find target documents. Crawlers can find documents not only by using locations that you specify, but also by extracting the locations of documents from the data source responses. For example, the web crawler can find locations of new pages by extracting URLs that are included in the retrieved pages.

You can create multiple crawlers in a collection. The crawlers have their own process, and consume computing resources only when they are running. To determine the required memory size of the crawler server in your capacity planning, add 100 MB for every concurrent crawler instance, except for web crawlers. Memory consumption of a web crawler instance can be up to 500 MB. Because all crawlers are created on a single server (either the master server or a crawler server), you must use a high performance computer if you plan to run many crawler instances at the same time.

Crawlers can collect and extract content from a variety of data source types. The performance characteristics of crawlers vary depending on the data source. The data sources often become a bottleneck if they are not appropriately tuned for crawling. Configuring and tuning the data source system is out of the scope of this document.

A common characteristic among crawlers is the use of a database. When crawling data sources incrementally, the crawlers store information such as visited locations, crawled time, and last modified date in database tables. The size of the database tables usually increases as you increase the number of retrieved documents. 1 GB of dedicated disk space should be sufficient for every 1 million documents to be crawled.

The crawler database is stored in the following directories on the crawler server (if you use the “Distributed server” configuration) or master server (if you use the “All on one server” configuration):

- UNIX: `$ES_NODE_ROOT/data/cloudscape/crawlers`
- Windows: `%ES_NODE_ROOT%\data\cloudscape\crawlers`

`ES_NODE_ROOT` is an environment variable that indicates the location of user data. This variable is set by the installation program. The default value of this variable is the `esdata` directory under the home directory of the user that you specify during the installation. On the master server, this directory contains the master configuration files. It is an important directory in regards to the tuning parameters.

When you retrieve a very large number of documents, it is recommended that you create multiple crawlers to separate the crawl space into multiple crawl spaces that contain approximately 10 million documents each. Having multiple crawlers helps prevent scanning of the database record from becoming the bottleneck. For example, let’s consider a file system data source that contains 8 directories, each containing 5 million documents, resulting in a total of 40 million documents. In this case, it is better to create 4 file system crawlers that each crawl 2 directories (10 million documents) instead of a single file system crawler to retrieve all 40 million documents.



## Indexer

The indexer component creates the index, which is the data structure that enables ingested documents to be searched. If you want to improve the end-to-end index building process, use your most powerful server as the master server and additional index servers, and carefully tune this component by using the parameters described in “Tuning parameters” on page 13. If you cannot use the distributed indexing configuration (for example, because your system doesn’t have a shared disk), and if you plan to use many document processors, you must allocate a high performance machine as the master server because only one indexer resides on the master server. The scalability of the document processors depends on the size of the transferred data size and the network communication throughput.

To efficiently utilize many document processors, the number of indexer threads is very important. At first, the indexer thread delegates the document to a document processor thread. Then the indexer thread waits until it receives documents that were processed by the document processor and are ready to be stored in the index (Figure 2). Thus, you must also take into account the number of waiting indexer threads. You can estimate that the number of waiting indexer threads is the same as the total number of document processor threads.

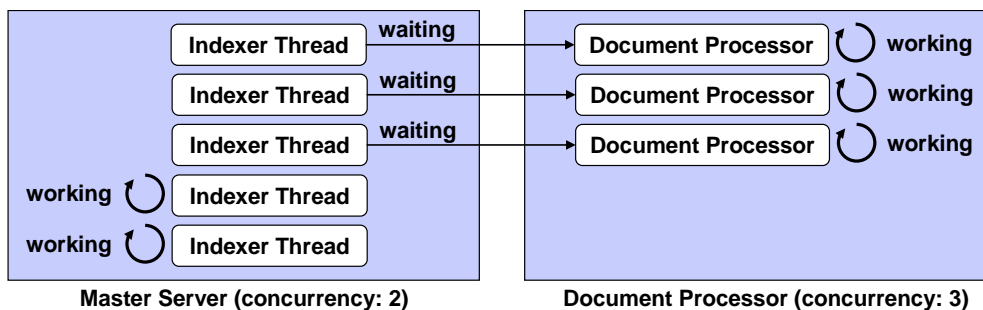


Figure 2. Indexer Threads and Document Processors

By default, the number of document processor threads per server is set to the same number as the number of indexer threads. If you use machines that have more than 2 cores as document processor servers, you may need to increase the number of threads. The recommended number of document processor threads is 2 times as many as the number of cores, because a modern processor has the capability to run 2 or more threads for each one.

$$\begin{aligned} \text{number\_of\_document\_processor\_threads\_per\_server} = \\ 2 * \text{number\_of\_CPU\_cores\_of\_document\_processor\_server} \end{aligned}$$

The total number of document processor threads can be defined as follows:

$$\begin{aligned} \text{total\_number\_of\_document\_processor\_threads} = \\ \text{number\_of\_document\_processor\_threads\_per\_server} * \\ \text{number\_of\_document\_processor\_servers} \end{aligned}$$

The default number of indexer threads is 1. In most cases, you need to increase the number of threads. For a distributed indexing configuration, the recommended number of indexer threads is equal to twice the number of CPU cores of the most powerful indexing server:

$$\text{number\_of\_threads\_of\_indexer} = 2 * \text{number\_of\_CPU\_cores\_of\_indexer\_server}$$

For a non-distributed indexing configuration, the recommended number of indexer threads is equal to twice the number of CPU cores of the master server plus the total number of document processors:

$$\text{number\_of\_threads\_of\_indexer} = 2 * \text{number\_of\_CPU\_cores\_of\_master\_server} + \text{total\_number\_of\_document\_processor\_threads}$$

For example, you have 1 master server that has 2 quad-core CPUs and 2 additional document processor nodes that each contain a 1 quad-core CPU. Based on this example,

$$\begin{aligned} \text{number\_of\_CPU\_cores\_of\_master\_server} &= 8 \\ \text{number\_of\_CPU\_cores\_of\_document\_processor\_server} &= 4 \\ \text{number\_of\_document\_processor\_servers} &= 2 \\ \text{number\_of\_document\_processor\_threads\_per\_server} &= (2 * 4) = 8 \\ \text{total\_number\_of\_document\_processor\_threads} &= (8 * 2) = 16 \\ \text{number\_of\_threads\_of\_indexer} &= (2 * 8) + 16 = 32 \end{aligned}$$

Many of the content analytics functions require facets that are generated by the indexer. Thus, generating facets is a mandatory and important responsibility of the indexer. According to estimates, tens of millions of facets are generated per 1 GB of ingested documents when using the default settings. Because such a large set of facets must be processed by indexers, facet generation usually becomes the bottleneck of the end-to-end pipeline to create the index. To estimate the indexing throughput accurately, you must have a clear idea about the number of generated facets. Fortunately, in many cases, the number of facets generated from a document can be estimated based on the byte size of the document. You can estimate the required time to complete indexing based on the total number of bytes in the document set. The following formula to calculate the required memory size assumes that the default facet configuration generates 1 new facet value every 100 bytes.

When you estimate the required memory size of the indexer, it is important to know that the indexer has two types of caches that are used to process facets. By default, the indexer uses the cache that consumes the specified size of memory (the LRU taxonomy cache) and requires approximately 20 MB for keeping it in memory. This memory consumption is included in the default maximum heap setting. Alternatively, you can use the cache that stores all the facets in memory (the TrieL2O and DA taxonomy cache). In this case, you must calculate the required memory size for the facets very carefully. According to our estimation, 250 MB is required for every 1 GB of document text. For instructions on how to configure the system to use a specific cache type, see “Setting the taxonomy cache type” on page 20.

Watson Explorer Content Analytics might automatically use the DA taxonomy writer cache if large memory is assigned to indexers. If you want to use LRU cache specifically to reduce memory consumption, you can specify the cache type explicitly as shown on page 20.

In addition, to store the structure of the facet hierarchy in memory, the indexer requires 50 MB memory for every 1 GB documents. When you use the LRU cache to process facets, use the following formula to calculate the required memory:

$$\text{memory\_for\_Taxonomy\_Cache} = 50 \text{ MB for every 1 GB documents}$$

When you use the TrieL2O or DA cache, use the following formula to calculate the required memory:

$$\text{memory\_for\_Taxonomy\_Cache} = 250 \text{ MB for every 1 GB documents}$$

While the document processors do the CPU-intensive tasks, indexing is a mixture of CPU-intensive tasks and disk I/O intensive tasks. The number of disk I/Os becomes the bottleneck of the indexing throughput because it can prevent concurrent processing. Most of the indexing task involves generating posting lists, which are the sorted document ID lists associated with respective terms. The size of the sorting buffer that is allocated in memory determines how much disk access is needed to build the index. Thus, a larger sorting buffer in memory reduces the number of disk I/Os and results in increased total throughput. In order to achieve the best indexing throughput, provide as large an index buffer as possible for your system. The index buffer is created for each partition.

The *buffer\_size* is the size of the buffer used by the index writer. Basically, it should be determined from the target indexing throughput. For simplicity, 50 MB is implicitly assigned for every 1 million documents.

500 MB is the base memory requirement and provides a safety margin for the other tasks of the indexer. If you plan to configure more than 10 threads for the indexer, recalculate this margin for memory consumption of indexer threads. You should estimate 50 MB memory for each indexer thread. Recalculate the required memory size and manually set the maximum heap size of the indexer component.

The memory size of the indexer can be calculated by using the following formula:

$$\text{Required\_Memory} = (500 \text{ MB or } \text{number\_of\_threads\_of\_indexer} * 50 \text{ MB}) + \text{memory\_for\_Taxonomy\_Cache} + \text{buffer\_size}$$

The index is built incrementally. In a short period of time (several minutes by default), new data that is generated from the added documents is written to the index. After this operation, new documents can be displayed in the search results. This operation of writing new data to the index is called the “index commit” operation. The period of time between each index commit operation is called the “commit interval”.

Even if you have a sufficient sorting buffer, the buffer is not fully utilized until the commit interval is properly set because the buffer is forced to be flushed when the index is committed to make it enabled for search. Frequent index commit operations cause many disk I/Os, which possibly becomes a bottleneck. On the other hand, as discussed above, when the sorting buffer becomes full before the index commit, the data in the buffer is temporarily written to the disk. Writing the data to the disk also increases the disk I/O as well as the frequency of index commits.

Changing the commit interval involves a tradeoff between the throughput and the response time. If you need the documents to be searchable as soon as possible, you should set the commit interval to a small value. For instructions, see “Adjust the index commit interval” on page 19.

The indexer consumes the largest amount of disk space. The indexer creates the main text index. The indexer also creates the document cache and thumbnails, if you choose these options when you create a collection. In addition, the main index might include other indexes, such as an anchor index if link analysis is enabled for the collection.

To estimate the disk size for the index, you need to consider two different types of document sizes:

- The original document size or binary document size indicates the total size of the data that is included in a file or a set of files that can be retrieved from the file system<sup>1</sup>.
- The textual data size indicates the amount of readable characters (including spaces and control characters) in a file or a set of files that is extracted and processed as the linguistic part of data.

The size of the main text index is almost the same as the size of the textual data included in the document. For a more precise determination of the index size, the size of the textual data should be investigated for each individual case. However, in typical cases, the textual data of PDF and Microsoft Word files is approximately 20% of the original document size.

While the size of the index is almost the same as the total size of the ingested textual data, the indexer also requires disk space for storing temporary files. Documents can be added to the index incrementally due to the segment database that handles segmented index files as a single large index. When the number of segment files reaches the specified number (known as the merge factor),

---

<sup>1</sup> On the Windows operating system, you can see the original document size as “size” in its property window.

the files are merged into a large single file for optimization. When merging segment files, the index can consume up to 3 times as much disk space as the current index size.

The largest temporary files are created when the number of merges performed equals a power of 10 (the  $10^1$ ,  $10^2$ ,  $10^3$  ... index merge). Watson Explorer Content Analytics requires having another copy of the index when it reads the index. Thus if you need to use the analytics functionality during index merge, the total required disk space becomes 4 times the original index size.

$$\text{Required\_Disk\_for\_Main\_Index} = \text{total\_textual\_data\_size} * 4$$

In addition to the main index, you need to have more space on the disk if you enable any of the following options: document cache, thumbnails, and static ranking by the number of links.

The document cache allows rebuilding the entire index from scratch without re-crawling documents from the data sources. The document cache is stored in the same segment database that is used by the index. The document cache enables you to add the compressed data incrementally into the single repository. As was required for the main text index, the document cache also requires disk space for the temporary files. If your collection consists of plain text, you can estimate the ratio of the compressed data size as 80% of the original binary documents.

$$\text{Required\_Disk\_for\_Document\_Cache} = \text{total\_document\_size} * 80\% * 4$$

If you enable support for thumbnails, you need to add the size of the thumbnails to the required disk size estimation. The average size of a thumbnail is 50 KB. Because the thumbnail is stored in the same segment database as the index, the thumbnail requires three times as much disk space as its total size. Because thumbnails are generated for Microsoft Office documents and PDF documents, you need to include them in the *number\_of\_rich\_formatted\_documents*.

$$\text{Required\_Disk\_for\_Thumbnails} = \text{number\_of\_rich\_formatted\_docs} * 50 \text{ KB} * 4$$

If you enable static ranking by the number of links to a document, you need additional space to store information about the links. The size of the link information depends on the number of links included in the ingested documents. In our tests of crawling average web pages, the link information becomes about 33% of the size of the text index. To be on the safe side, allocate disk space equivalent to 50% of the size of the main index to store the link information.

$$\text{Required\_Disk\_for\_Static\_Ranking} = \text{Required\_Disk\_for\_Main\_Index} * 50\%$$

Then the total disk size can be calculated by summing them up:

$$\text{Required\_Disk} = \text{Required\_Disk\_for\_Main\_Index} + \text{Required\_Disk\_for\_Document\_Cache} + \text{Required\_Disk\_for\_Thumbnails} + \text{Required\_Disk\_for\_Static\_Ranking}$$

For example, there are 1 million documents (40% are HTML and text documents and 60% are PDF and Word documents) with a total size of 10 GB (4 GB for HTML/text and 6 GB for PDF/Word). Based on this example,

$$\text{Total\_textual\_data\_size} = (4 \text{ GB HTML/text}) + (0.2 * 6 \text{ GB PDF/Word}) = 5.2 \text{ GB}$$

$$\text{Required\_Disk\_for\_Main\_Index} = \text{Total\_textual\_data\_size} * 4 = (5.2 \text{ GB} * 4) = 20.8 \text{ GB}$$

$$\text{Required\_Disk\_for\_Document\_Cache} = \text{Total\_document\_size} * 80\% * 4 = (10 \text{ GB} * 0.8 * 4) = 32 \text{ GB}$$

$$\text{Required\_Disk\_for\_Thumbnails} = \text{number\_of\_rich\_formatted\_docs} * 50 \text{ KB} * 4 = (600k * 50 \text{ KB} * 4) = 114 \text{ GB}$$

$$\text{Required\_Disk} = 20.8 \text{ GB} + 32 \text{ GB} + 114 \text{ GB} = 166.8 \text{ GB}$$

**Advanced:** If you add extra annotators or annotation rules, recalculate the required memory size by performing a sampling test of the generated facets. To estimate the required memory based on the size of the sample facet index:

1. Prepare a set of sample documents and note the size of the document text in megabytes.
2. Create a temporary collection for the sampling test.
3. Build an index by ingesting the set of sample documents.
4. Determine the size of the facet index that is created in the following directory on the master server:
  - UNIX: `$ES_NODE_ROOT/data/collection_name/index/facets`
  - Windows: `%ES_NODE_ROOT%\data\collection_name\index\facets`
5. Calculate the total amount of memory required in bytes by using the following formula:

$$\text{Required\_Memory} = 0.4 * \text{Facet\_index\_size} * \frac{\text{Total\_document\_text\_size}}{\text{Sample\_document\_text\_size}}$$

- *Facet\_index\_size* is the size of the facet index that is generated for the sample documents.
- *Total\_document\_text\_size* is the total size in bytes of the text in the target document set.
- *Sample\_document\_text\_size* is the size in bytes of the text in the sample document set.
- The factor 0.4 means that the facet data in memory is compressed into 40% of the data on disk.

Add this calculated number of bytes to the maximum heap size of the indexer instead of adding the 250 MB per 1 GB of document text mentioned above.

## Document processor

The document processor is a sub-component of the indexer. The document processor parses the document to extract text and metadata, and tokenizes the extracted text data to enable the text to be indexed. UIMA pipelines are also processed in the document processors, and a large number of annotations are generated. This component can be scaled-out to multiple additional servers. The number of document processors is logically not limited, but the maximum number of total document processors should be equal to the total number of indexer threads. If the system is configured with the distributed indexing option, the effective number of document processors is the same as the number of indexer threads at most. However, if the system is configured without the distributed indexing option, the effective number of document processors is often limited by the throughput of the indexer, which depends on the computing power of the master server.

By default, a document processor is created on a master server to enable you to build an index before you add additional document processor servers. After adding one or more document processor servers, you can stop the document processor on the master server. Stopping the document processor on the master server helps allocate more computing resources to the indexer component for a non-distributed indexing configuration. For a distributed indexing configuration, stopping the document processors is not recommended because each indexer tries to use local document processors (running on the same server), which gives you the best performance in most cases.

You can create multiple document processor threads on a document processor server. Each document processor thread requires memory proportional to the largest document size. Increase the maximum heap size if you increase the number of document processor threads in a document processor server. For instructions, see “Setting the number of document processor threads for each server” on page 18.

The required memory size for each document processor instance can be calculated from the size of the largest amount of textual data in the ingested documents. Allocate 100 times more memory than the largest data size for each document processor instance. In addition, 100 MB is required for the dictionary data.

Use the following equation to calculate the total required memory size:

$$\text{Required\_Memory} = \frac{\text{largest\_textual\_data\_size} * 100}{\text{number\_of\_document\_processor\_threads}} + 100 \text{ MB}$$

If you enable sentiment analysis, add 200 MB to the preceding formula, as it involves the deep parser for some languages.

If the collection includes many rich-formatted documents such as PDF and Microsoft Office documents, set the number of Stellent parsers to equal the number of document processor instances. The peak amount of memory consumed by a Stellent process is estimated at 50 MB plus 10 times the file size of the largest document size (that is, the total document size, not the textual data size) that will be stored in the working memory.

$$\text{Required\_Memory} = \text{largest\_document\_file\_size} * 10 * \text{number\_of\_Stellent\_parser\_instances} + 50 \text{ MB}$$

For example, to process a collection in which the largest document is a 10 MB PDF document, the memory consumed by each Stellent process should be estimated as 150 MB. For instructions, see “Setting the number of document processor threads for each server” on page 18.

## Search server

Search servers provide the search and analytics capabilities to users. This component can be scaled-out to multiple additional servers. Watson Explorer Content Analytics provides two index sharing alternatives: copy and share nothing, or share all by using GPFS/SAN.

It is recommended that you copy the index and share nothing if you have many concurrent users and search throughput is more important than indexing throughput and index size. In many cases, search throughput with the “copy and share nothing” option is higher than when using the “share all by using GPFS/SAN” option.

It is recommended that you share the index by GPFS/SAN with enterprise class storage (higher than DS4000 series) if the index is large and indexing throughput is of higher priority than search throughput. You can set up index sharing by specifying the GPFS shared directory as the data directory at installation time. You need a load balancer to distribute the users’ requests to the multiple search servers. It is strongly discouraged to use slow I/O systems like NFS/NAS, because doing so can result in poor response time.

Watson Explorer Content Analytics supports a distributed search option (select “Distribute index to separate search servers” when you create a collection). To enable the option, the index needs to be shared by GPFS/SAN. When this option is enabled, more than one search server can process a query request separately on assigned partitions, then gather the results and return them back. Doing so may reduce response time if the collection is very large, but it may cause lower query throughput (queries per second) because total workload is larger than it is with the non-distributed search option (e.g., a search server needs to merge results from other search servers).

It is recommended that you enable the distributed search option if all of the conditions below are true:

- You have very large collections
- You have several search servers
- You don’t have many concurrent users
- You really need better response time

Because the total workload is larger than that of the non-distributed search option, if your system needs to serve many concurrent users, you should use non-distributed search to get the best query throughput from the system. Ideally, the response time could be 1/n (where n means the number of search servers) compared to the non-distributed search option. However, this approach has some

overhead and the time required for the search operation for each partition will not be equal in reality (a search server must wait for the results from all search servers), so the response time will be far lower than the ideal response time.

Query performance depends on the size of the result set. Because the number of possible results increases with the number of ingested documents, the average query performance is largely dependent on the total number of documents in the index.

The required memory size of the search servers also depends on the number of ingested documents and the number of terms (variations of words) found in the ingested documents. However, because the number of terms increases as the number of documents increase, we can estimate the required memory size based on the number of documents.

In addition, the search servers store facet data for all documents in memory. To store facet data in memory, add 150 MB for every 1 GB of document text.

*Required\_Memory* = 150 MB for every 1 GB document +  
500 MB for every 10 million documents + 500 MB

If you expect that the variation of frequently requested queries is low, the search result cache can help improve the response time and throughput of the search servers. On average, the default 3000 entries in the result cache consume about 100 MB of heap memory. The result cache is stored with weak-references that are removed in the garbage collection that is executed when running out the heap memory.

Each additional search server requires sufficient disk space for a replica of the index. The replica includes the main text index and the thumbnails, which require the majority of the disk space. This disk space for the index replica is not needed if you use a shared disk configuration.

## Server configuration options

A server configuration option specifies the server roles and component locations. Watson Explorer Content Analytics offers two options of server configuration:

- **All on one server:** In this server configuration (Figure 3), all required components are created on the master server. Note that if you need to build the index while crawlers and search servers on the master server are simultaneously running, the number of document processors that can be used is reduced.

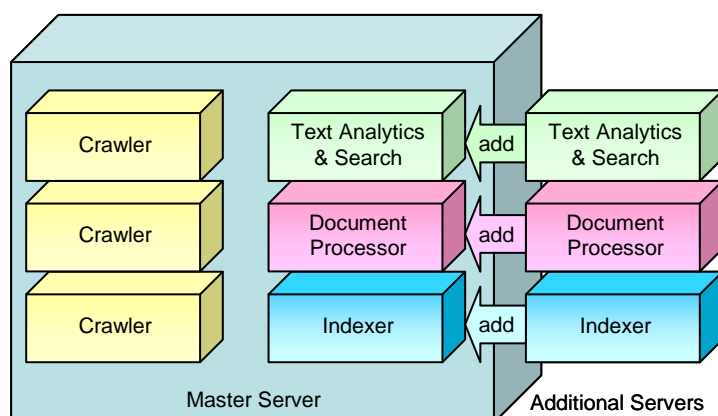


Figure 3. All on one server configuration

Select this server configuration if you want to use a single high-performance computer instead of increasing the number of additional servers. In this case, you can have only one indexer, one document processor, and one search server on the same computer. Thus you need to configure the system to serve many threads for these components.

- **Distributed server:** In the distributed server configuration (Figure 4), the major workload of the master server is related to the indexing task. Because the crawler component and search servers for crawlers and search servers, e.g., one additional node must be dedicated to crawler role, and another node must have the search role (which may have either the document processing role or index role in addition to the search role). This configuration allows for higher indexing performance and greater scalability. This configuration is also useful if you have many crawlers and run them independently from the indexer.

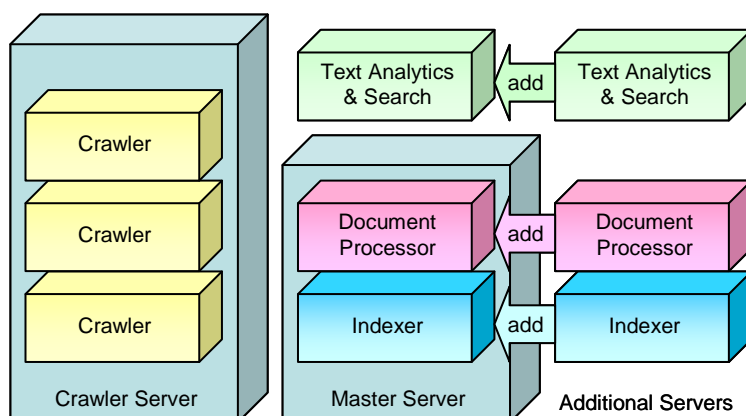


Figure 4. Distributed server configuration



In either configuration, crawlers must be in one server, which should be the master server in an all on one server configuration or the crawler server in a distributed server configuration.

Both configurations allow you to set up additional servers according to your needs. If your performance requirements change, you can add and remove additional servers. You select the server configuration option by selecting a server type when you install the master server, as shown in Figure 5.

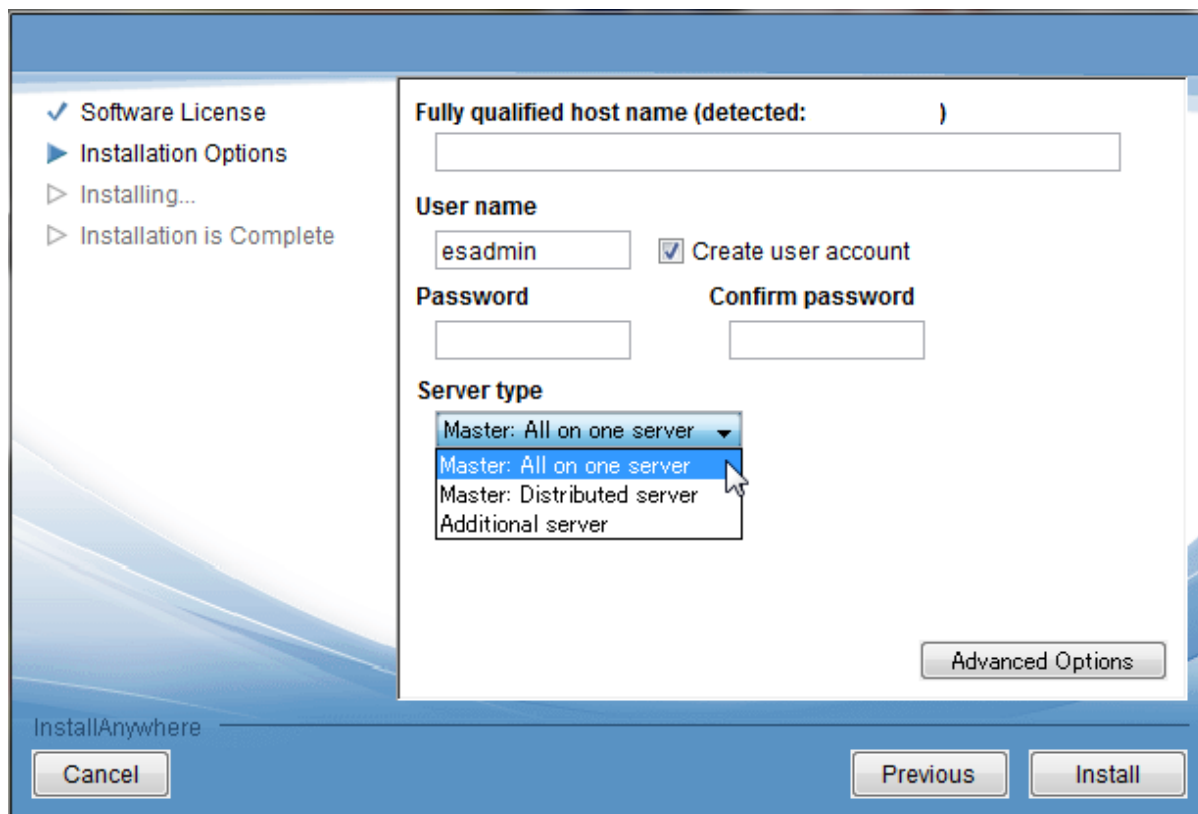


Figure 5. Selecting the server type in the installation program

To add an additional server to the system, select **Additional server** when you install Watson Explorer Content Analytics on the target server. You specify the role of the server by using the administration console to add the server to the system topology.

# Tuning parameters

This section describes the major tuning parameters and provides instructions for setting these parameters. After you install Watson Explorer Content Analytics, you must create a collection and then set the following parameters according to the guidelines in the sections below.

The basic tuning strategy of Watson Explorer Content Analytics is similar to the general tuning strategy of your servers. Monitor the CPU utilization, I/O frequency, and throughput, and balance these parameters by removing any bottlenecks.

Indexers and document processors require many threads to utilize many cores. Network and disk I/O wait time can cause CPU sleep time. To fill these sleep times, it is usually better to have 2 - 3 times as many threads as the number of cores.

The bottleneck of an average analytics collection is caused by facet processing on the indexers. When the CPU utilizations of additional document processor servers are low, stopping the document processor on the master server can often improve the indexing throughput if the distributed indexing option is not enabled.

Some of the parameters described below are automatically configured, e.g., default values of some parameters (mostly related to indexing performance) are calculated from other parameters. It is critical to set such key parameters to let Watson Explorer Content Analytics know how large system resources can be consumed to process data for the collection. You should carefully configure the parameters below:

- The number of indexer threads
- The memory size for parsing and indexing, and document processing

Watson Explorer Content Analytics will adjust other parameters to utilize assigned resources more effectively if you assign large numbers to these parameters. The default values are subject to change without notice in future fix packs or releases to improve the performance, so some of these values are not published in any documentation, including this document. If you want to fix the parameters, you can always override them by specifying values explicitly as described below.

If you migrate collections from previous releases, some of the parameters have been specified already even if you have not changed them explicitly. In such cases, the specified value will be used and will not be adjusted automatically.

## Conventions

The following conventions are used in this section.

**collection\_ID** is the ID of the collection for which you want to set a parameter value. Note that this ID is not the collection name that you specified when you created the collection. You can specify the collection ID in the advanced options when you create the collection. Otherwise, the system automatically assigns the ID a random number, such as col\_81071. To determine the collection ID after a collection is created, open the Collections view in the administration console, locate the collection that you want to see, and click **Actions > Settings > View collection settings**.

**node\_ID** is the ID of the server. The master server is assigned the ID `node1` and additional servers are assigned IDs such as `node2` and `node3` in the order in which they are added to the system.

**session\_ID** is based on the collection ID, node ID, and the component name. For example:

- Indexer session ID: *collection\_ID.indexservice*
- Document processor session ID: *collection\_ID.docproc.node\_ID*
- Search server session ID: *collection\_ID.searcher.node\_ID*

You can determine the session IDs by entering the `esadmin check` command:

```
bash-3.2$ esadmin check
Session ID           Node ID      PID      State
-----
a.docproc.node1     node1       -        -
a.exporter          node1       -        -
a.indexservice.node1 node1       -        -
a.searcher.node1    node1       -        -
a.stellent          node1       -        -
...
```

Note that a *session* indicates an instance of a component. A session consists of a process and is managed by the common communication service (CCL) of Watson Explorer Content Analytics.

**config\_ID** is the ID of the session configuration that is automatically assigned in the session configuration file (`$ES_NODE_ROOT/master_config/collection_ID_config.ini`), in the format *sessionNumber*. The following description is an example of a session configuration:

```
session1.clone_sid=a.indexservice
session1.collectionid=a
session1.configDir=a.indexservice
session1.dataDir=
session1.description=Index Document Processing Session
session1.displayname=Index Document Processor - Collection a (node1)
session1.domain=.
session1.flags=0
session1.hard_max_heap=10,8192
session1.id=a.docproc.node1
session1.max_heap=8192
session1.nodeid=node1
session1.sectiontype=session
session1.subtype=docproc
session1.type=indexservice
```

This document does not explain all of these properties. For the purposes of this document, you only need to know how to find the `max_heap` property of the session for which you need to change the maximum heap size (as described in the following section).

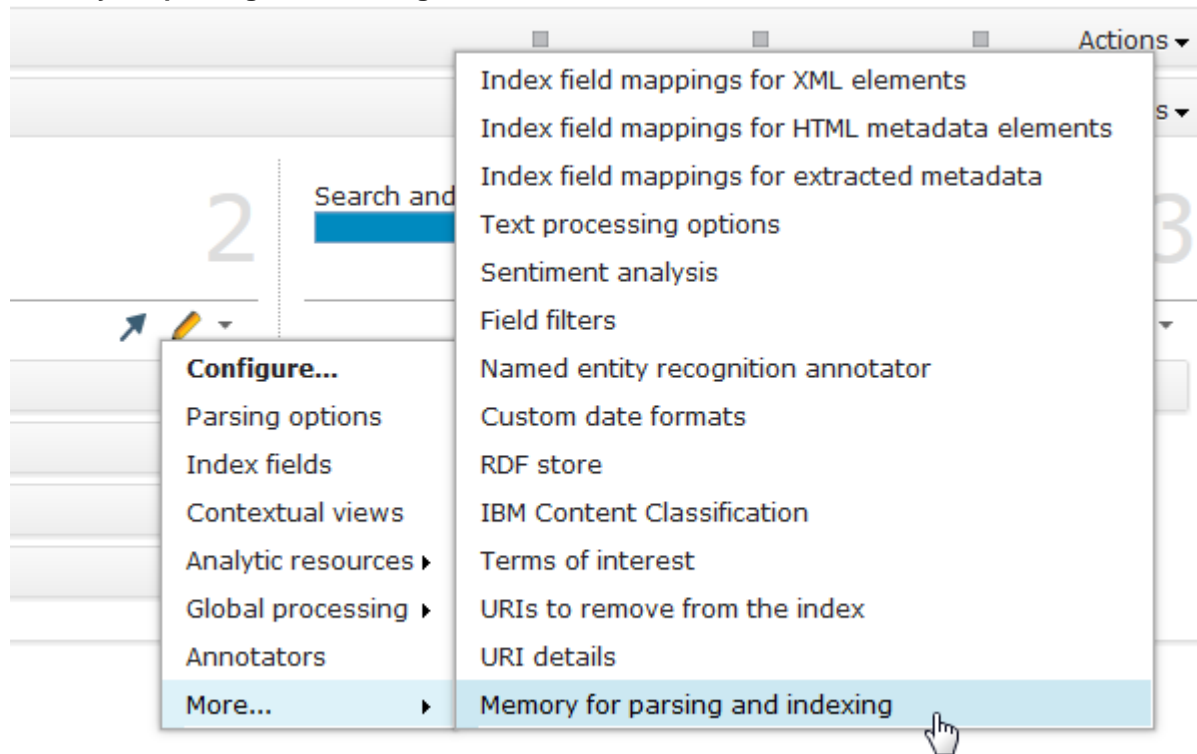
## Setting the maximum heap size

If you are analyzing large collections, you typically need to configure the maximum heap size of indexers, document processors, and search servers. By default, the system sets their memory sizes to 1024 MB. See “System architecture” on page 2, to understand how much memory needs to be assigned for each component.

You can change the maximum heap size of sessions for (i) parsing and indexing, (ii) document processor, and (iii) search server, per collection, from the administration console. If you want to change it for other sessions, determine the collection ID and the session ID of the session that you want to configure.

To set the maximum heap size from the administration console:

1. Log in to the administration console and click **Collections** in the toolbar to open the Collections view.
2. In the list of collections, locate the collection that you want to edit.
3. If you want to change the maximum heap size of parsing and indexing and/or document processor, follow the steps below:
  - a. Click the edit button (pencil icon) in the **Parse and Index** area, and click **More > Memory for parsing and indexing**.



- b. Modify the value in “**Memory size for parsing and indexing (MB)**” and/or “**Memory size for each document processor (MB)**”, and click **OK**.

## Memory Configuration for Parsing and Indexing

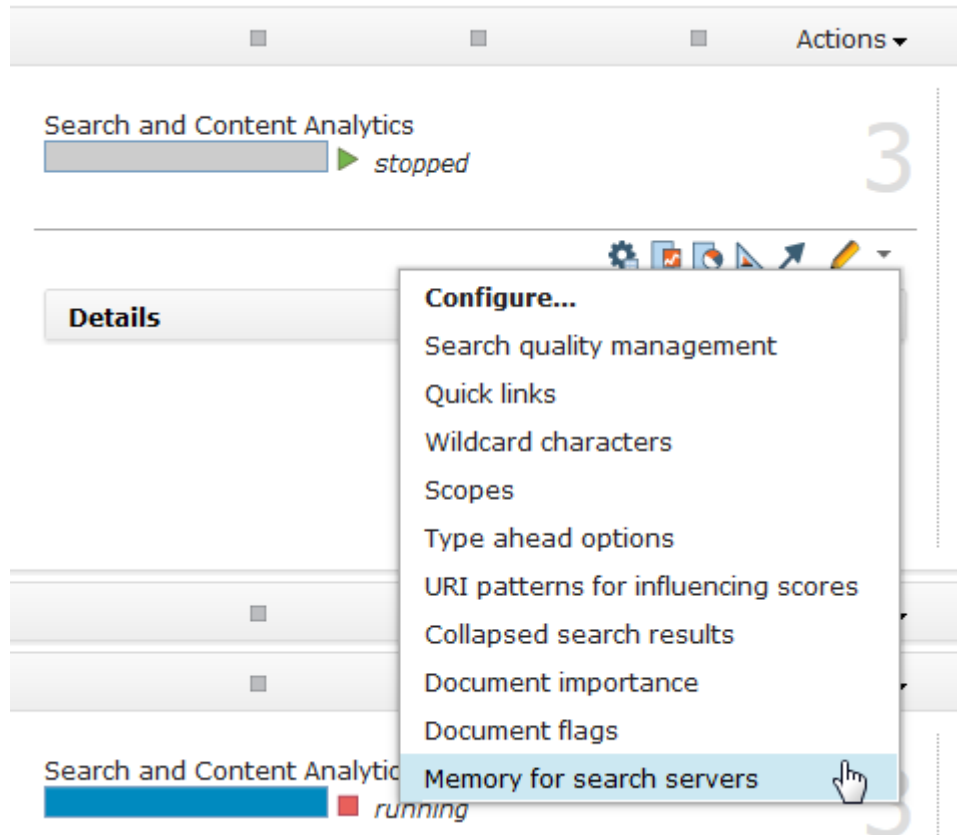
You can configure the maximum amount of memory to use for parsing :  
The default value is 1024 MB. If the value is too large or too small, par

Memory size for parsing and indexing (MB):

Memory size for each document processor (MB):

The example above is to set 4096 MB for parsing and indexing (indexservice) and 2048 MB for document processor (docproc).

4. If you want to change the maximum heap size of the search servers, follow the steps below:
  - a. Click the edit button (pencil icon) in the **Search** area and click **Memory for search servers**.



- b. Modify the value in “**Memory size for each search server (MB)**” and click **OK**.

## Memory Configuration for Search Servers

You can configure the maximum amount of memory to be used by the search servers. The default value is 1024 MB. If the value is too large or too small, the search servers may not start.

Memory size for each search server (MB):



5. Restart “Parse and Index” if you change the memory or parsing and indexing, and restart “Search and Content Analytics” if you change the memory for search servers.

To set the maximum heap size for other sessions:

1. Enter the `esadmin system stopall` command to stop the system.

2. In the `$ES_NODE_ROOT/master_config/collection_ID_config.ini` file, find the session configuration for which you want to change the heap size. The session configuration includes the following lines:

```
config_ID.id=session_ID
config_ID.max_heap=max_heap_size_in_MB
```

For example, the following lines are from a session configuration of an indexer session for a collection named col1:

```
session3.id=coll.indexservice
session3.max_heap=1024
```

3. Change the value of the `max_heap` property and save the file.
4. Enter the `esadmin system startall` command to restart the system.

## Setting the number of indexer threads

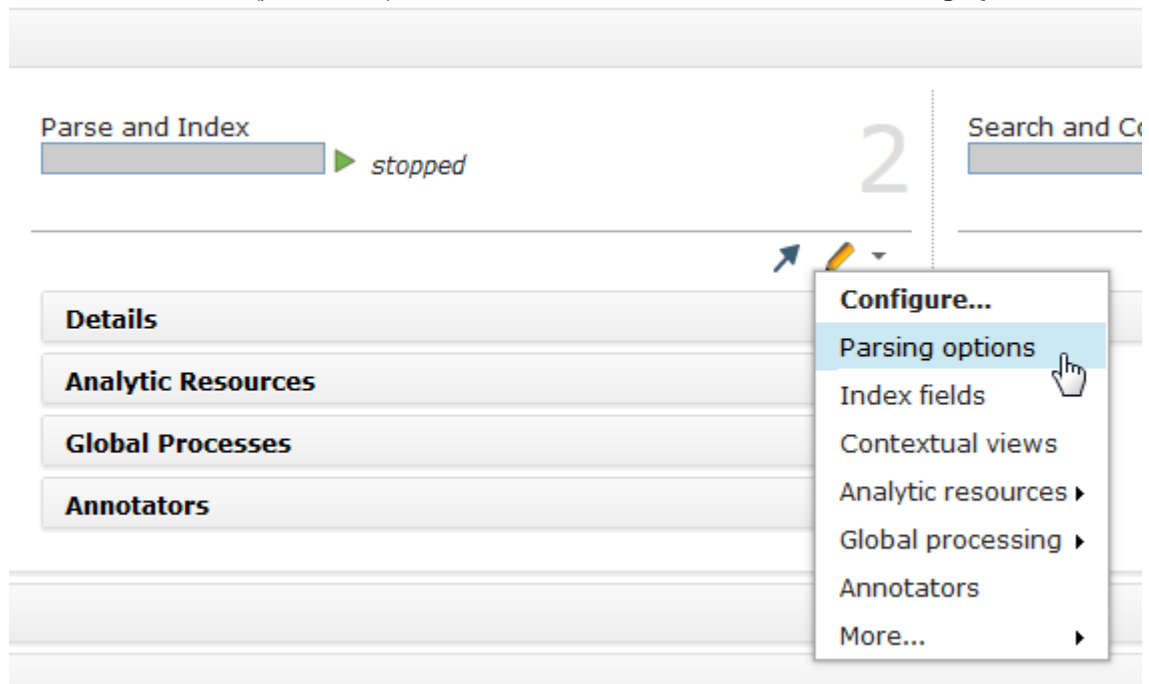
The default number of indexer threads is 1. In most cases, you need to increase the number of threads. Typically, the recommended number of indexer threads is equal to twice the number of CPU cores, plus the total number of document processors for a non-distributed indexing configuration:

$$\text{Number\_of\_threads} = 2 * \text{number\_of\_CPU\_cores} + (\text{number\_of\_document\_processor\_servers} * \text{number\_of\_document\_processors\_per\_server})$$

The indexer threads delegate the ingested documents to the document processor. The threads sleep during document processing until they receive the processed documents that are ready to be stored in the index. Thus, you must also take into account the number of sleeping (waiting) indexer threads. You can estimate that the number of waiting indexer threads is the same as the number of document processors.


To change the number of indexer threads:

1. Log in to the administration console and click **Collections** in the toolbar to open the Collections view.
2. In the list of collections, locate the collection that you want to edit.
3. Click the edit button (pencil icon) in the **Parse and Index** area and click **Parsing options**.



4. Modify the value in the **Number of index threads** field and click **OK**.

## Parsing Options

[Learn more](#) 

- Specify whether users can use native XML search (such as XPath or XML fra
- You can improve search quality for languages like German and Korean by er
- Before you increase the number of index threads, ensure that your system
- Specify whether documents are to be detected automatically and parsed im

Native XML search:

Compound terms:

Number of index threads:

N-gram segmentation:

N-gram segmentation policy:

Incoming documents:

Character length, including system-preserved length for each document that the p:

## Setting the number of document processor threads for each server

The default number of document processors for each server is 4. If you use powerful servers for the additional document processor servers, you might want to increase the number of document processor instances for each server to utilize the many cores. Set the number of document processors equal to 2-4 times the number of CPU cores. This parameter is automatically adjusted to the number of indexer threads unless otherwise below.

To set the number of document processors for each server:

1. In the `$(ES_NODE_ROOT)/master_config/collection_ID.indexservice/collection.properties` file, modify the `NumberOfDocumentProcessors` property. If you need to increase the number of parsers for rich text documents, change the value of the `NumberOfStellentParsers` property to 4.
2. In the `$(ES_NODE_ROOT)/master_config/collection_ID.indexservice/collection.xml` file, find the `<tokenizer>` element under the `<config>` and `<collection>` elements.
3. Add the `<numberOfTokenizers>` element in the `<tokenizer>` element, as follows:

```
<tokenizer>
  <numberOfTokenizers>8</numberOfTokenizers>
</tokenizer>
```

4. Modify the number of tokenizers. The number of document processors and the number of tokenizers must be the same.
5. Save the files.
6. Restart "Parse and Index" for the collection.

## Setting the document length limit

You can change the document length limit by adding the `<documentLengthLimit>` element to the `collection.xml` file. Extracted document text that exceeds this limit is truncated by the specified length. Change this parameter only if you need to process extracted document text that is larger than the default length of 16 million characters. If you increase the document length limit, you must assign more heap memory to the document processors.

To set the document length limit:

1. In the `$ES_NODE_ROOT/master_config/collection_ID.indexservice/collection.xml` file, find the `<parser>` element under the `/config/collection` element.
2. Add the `<documentLengthLimit>` element to the `<parser>` element:
 

```
<parser>
  <!-- the limit of the document length -->
  <documentLengthLimit>262144</documentLengthLimit>
</parser>
```
3. Save the file.
4. Restart "Parse and Index" for the collection.

## Setting the index buffer size

The default value of the index buffer size can be adjusted automatically, but you can explicitly set the index buffer size, by following the instructions below:

1. Edit the `$ES_NODE_ROOT/master_config/collection_ID.indexservice/collection.xml` file.
2. Under the `/config/collection/indexes` element, find the `<index>` element with a `<type>` element value of `Text`.
3. Find the parameter named `BufferSize` under the `<index>` element, and rewrite its value with a new value.

```
<indexes>
  <index>
    <type>Text</type>
    <path>text</path>
    <enabled>true</enabled>
    <indexMode mode="normal"/>
    <property name="SweetSpotLength" value="50"/>
    <property name="BufferSize" value="2048"/>
    . . . .
```

4. Save the file.
5. Restart "Parse and Index" for the collection.

## Adjusting the index commit interval

To adjust the interval of the index commit operation, you need to change two values in the system. The first parameter is `numberOfDocumentsTilFlush`, which specifies the number of ingested documents that triggers the index to perform the index commit operation. By setting a small value for this parameter, the ingested documents can be searchable after a short period of time, but the throughput is decreased. If maximum throughput is required, this parameter can be set as `-1`, which means that index commit will not be triggered based on the number of ingested documents.



The commit interval in bytes must be smaller than 1/4 of the buffer size of index. The recommended value for the numberOfDocumentsTilFlush parameter is calculated by using the following formula:

$$\text{numberOfDocumentsTilFlush} = (\text{indexBufferSize} * \text{numberOfPartitions} / 4) / \text{average\_document\_size}$$

For example, if indexBufferSize = 2048 MB, numberOfPartitions = 1, and average\_document\_size = 4096 bytes, then numberOfDocumentsTilFlush = (2048 \* 1024 \* 1024 \* 1 / 4) / 4096 = 131,072 documents

To set the numberOfDocumentsTilFlush parameter:

1. In the `$ES_NODE_ROOT/master_config/collection_ID.indexservice/collection.xml` file, find the <indexer> element under the /config/collection element.
2. Add the <numberOfDocumentsTilFlush> element to the <indexer> element:  

```
<indexer>
  <numberOfDocumentsTilFlush>50000</numberOfDocumentsTilFlush>
</indexer>
```
3. Save the file.
4. Restart "Parse and Index" for the collection.

The second parameter for adjusting the index commit interval controls the maximum size of raw data store (RDS) files. RDS files are used internally in the queue between the crawlers and the indexer. RDS files temporarily store crawled documents on the disk to safely keep the crawled documents until they are written in the index. Crawled documents are written in several chunked RDS files, and each RDS file is removed when all documents included in the file are successfully written in the index. Thus you need to enlarge the size of the RDS files if you maximize the indexing throughput. On the other hand, using large RDS files might result in slow crawler throughput, so enlarge this value if you determine that indexer throughput is the bottleneck throughout the system.

To set the maximum size of RDS files:

1. Enter the `esadmin system stopall` command to stop the system.
2. Open the `$ES_NODE_ROOT/master_config/datalistener/dlConfig.prp` file.
3. Add the `DL_RDS_File_Limit` property with the new size in KB.  

```
DL_RDS_File_Limit=1024
```
4. Save the file.
5. Enter the `esadmin system startall` command to restart the system.

**Note:** The commit interval in bytes must be smaller than 1/4 of the buffer size of the index. As previously discussed, the recommended value of numberOfDocumentsTilFlush is calculated by using following formula:

$$\text{numberOfDocumentsTilFlush} = (\text{indexBufferSize} * \text{numberOfPartitions} / 4) / \text{average\_document\_size}$$

## Setting the taxonomy cache type

The taxonomy cache stores generated facets that are used by the indexer component. Watson Explorer Content Analytics provides three types of taxonomy caches:

**LRU:** Partially in memory. Scalability is limited.

**TrieL2O:** Completely in memory. This cache type is 2–5 times faster than LRU.

**DA:** Completely in memory. This cache is 5 - 10% faster than TrieL2O.

By default, the LRU cache or the DA cache is used depending on the size of memory assigned to the indexer. If you have enough memory, use a complete cache. If you want to process a large document set over a longer duration with a smaller memory footprint, you can configure the system to use the LRU cache that partially loads the taxonomy index in memory.

To set the taxonomy cache type:

1. In the `$ES_NODE_ROOT/master_config/collection_ID.indexservice/collection.xml` file, find the `<index>` element with a `<type>` element value of `Facet`. The XPath of this element is: `/config/collection/indexes/index[type=Facet]`
2. Add the `CacheType` property to the `<index>` element and set its value to `DA`, `TrieL2O`, or `LRU`:

```
<index>
<type>Facet</type>
<path>facets</path>
...
<property name="CacheType" value="LRU"/>
</index>
```
3. Save the file.
4. Restart "Parse and Index" for the collection.

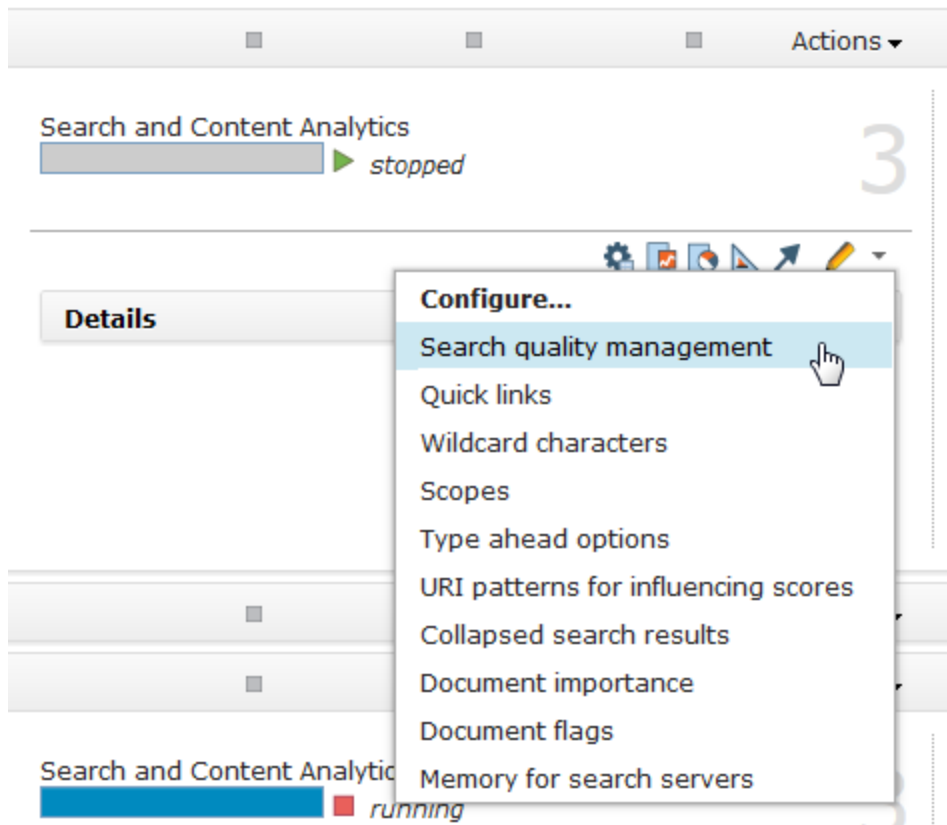
If you use the DA taxonomy cache, you can optionally set the number of partitions. Because concurrent read/write operations can be performed for each partition, setting the number of partitions enables the CPU resource to be efficiently utilized when processing facets. The valid range of the number of partitions is 1 – 36. To set the number of partitions of the DA taxonomy cache, add `<property name="NumberOfCachePartitions" value="16"/>` as a sibling element of the `<property name="CacheType" value="DA"/>` element that you inserted.

## Setting the number of search cache entries

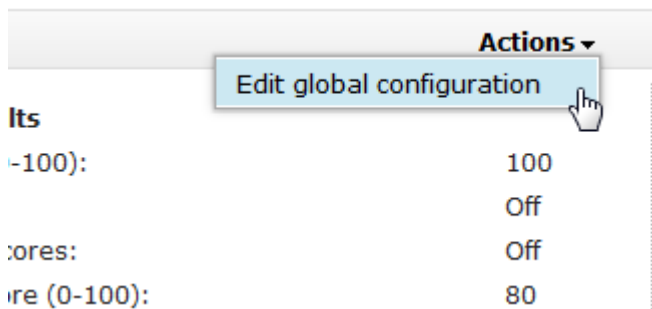
If you enable search servers to cache the result sets of the requested queries, you can set the maximum number of entries that are saved in the cache.

To set the maximum number of entries in the search cache:

1. Log in to the administration console and click **Collections** in the toolbar to open the Collections view.
2. In the list of collections, locate the collection that you want to edit.
3. Click the edit button (pencil icon) in the **Search** area (Enterprise Search Collection) or **Search and Content Analytics** area (Content Analytics Collection), and click **Search quality management**.



4. On the Search Quality Management page, click **Actions > Edit global configuration**.



5. Ensure that the **Use the search cache** check box is selected and modify the value in the **Maximum number of entries in the cache** field.

**Edit Global Configuration**

[Learn more ?](#)

**Options for searching the collection**

Maximum number of retrieved results:  
 documents

Interval for index update detection:  
 seconds

Maximum number of threads allocated for query processing (1-8; default value is 1):  
(Note: Increasing this value can result in high system resource usage)

**Options for search results**  
If you change these options, restart the search servers for the collection.

Use the search cache

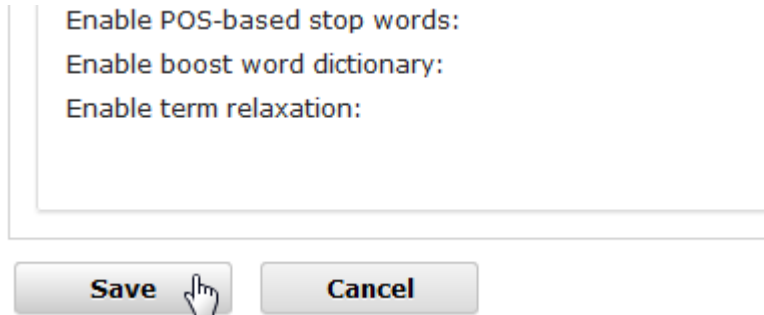
Maximum number of entries in the cache:

Maximum length of a sentence for document summaries (sentences are divided when the length exceeds this value):  
 characters

6. Click **Apply** in the dialog to apply the change.

Boost word dictionary name:

7. Click **Save**.



The image shows a configuration dialog box with three options: "Enable POS-based stop words:", "Enable boost word dictionary:", and "Enable term relaxation:". Below the options are two buttons: "Save" and "Cancel". A mouse cursor is pointing at the "Save" button.

8. Restart the search server.

## Setting the number of threads for search servers

If you use the default (embedded) application server, and if you expect that a large number of concurrent users will use the system as search or analytics users, you may need to increase the number of threads allowed for the application server. With the current implementation of the server, it is recommended to set the number of threads as follows:

$$\text{number\_of\_search\_threads\_per\_server} = \text{maximum\_number\_of\_concurrent\_users} * 2$$

To set the number, follow the steps below:

1. Enter the `esadmin system stopall` command to stop the system.
2. In the `$ES_NODE_ROOT/master_config/searchapp/wlp/server.xml` file, find the `<executor>` element.
3. Modify `coreThreads` attribute to the number calculated above:  
`<executor maxThreads="80" coreThreads="40">`
4. Save the file.
5. Enter the `esadmin system startall` command to restart the system.

You may also see a "server busy" error because of a lack of worker threads in the search server. In such a case, you can increase the number of worker threads by following the steps below. It is recommended to set the number larger than maximum number of concurrent users, at a minimum.

1. Enter the `esadmin system stopall` command to stop the system.
2. In the `$ES_NODE_ROOT/master_config/searchserver/dock/dock.xml` file, find the `<maxPoolSize>` element (note that there are two active occurrences of the element; you must change both).
3. Modify the value to the new number.
4. Enter the `esadmin system startall` command to restart the system.

## Conclusion

This document provided information and typical methods to do performance tuning for Watson Explorer Content Analytics V11.0. Watson Explorer Content Analytics can efficiently utilize a large memory, large storage, multiple cores, and a number of additional servers to process millions of documents. It is your responsibility to plan appropriate hardware resources that correspond with your requirements and configure the tuning parameters to fully utilize the hardware resources. Carefully review this paper to tune the parameters to gain the largest benefit from the invested resources.

## Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing  
IBM Corporation  
North Castle Drive  
Armonk, NY 10504-1785  
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing  
Legal and Intellectual Property Law  
IBM Japan Ltd.  
19-21, Nihonbashi-Hakozakicho, Chuo-ku  
Tokyo 103-8510, Japan

**The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:** INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation  
J46A/G4  
555 Bailey Avenue  
San Jose, CA 95141-1003  
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

#### COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows: © (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. 2004, 2015. All rights reserved.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

## ***Trademarks***

IBM, the IBM logo, and [ibm.com](http://ibm.com) are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at “Copyright and trademark information” at [www.ibm.com/legal/copytrade.shtml](http://www.ibm.com/legal/copytrade.shtml).

The following terms are trademarks or registered trademarks of other companies:

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.