

Update on CERN Search based on SharePoint 2013

E Alvarez¹, S Fernandez¹, A Lossent¹, I Posada¹, A Wagner¹,

¹CERN IT/CDA, 1211 Geneva 23, Switzerland

E-mail: andreas.wagner@cern.ch

Abstract. CERN's enterprise Search solution "CERN Search" provides a central search solution for users and CERN service providers. A total of about 20 million public and protected documents from a wide range of document collections is indexed, including Indico, TWiki, Drupal, SharePoint, JACOW, E-group archives, EDMS, and CERN Web pages. In spring 2015, CERN Search was migrated to a new infrastructure based on SharePoint 2013. In the context of this upgrade, the document pre-processing and indexing process was redesigned and generalised. The new data feeding framework allows to profit from new functionality and to facilitate the long term maintenance of the system.

Keywords

CERN, Search, Sharepoint 2013, FAST for Sharepoint

1. Introduction

CERN's Enterprise Search Solution "CERN Search", provides a central search solution for users and CERN service providers [1]. A total of about 20 million public and private documents from a wide range of document collections is indexed, including INDICO, TWiki, Drupal, SharePoint, JACOW, E-group archives, EDMS, and CERN Web pages.

FAST for SharePoint 2010 was used as search engine with success several years in a stable form. But, knowing that Microsoft had given an end date within their product lifecycle [2], it was decided to start planning the update to the next version: SharePoint 2013.

Soon it was clear that re-design and re-code some core parts of the system were needed, so taking advantage of that, most of the decisions were made having maintainability as the main goal of the update.

2. Service Architecture

Wanting to provide same service levels as we had, the architecture should ensure both high availability and high reliability. Microsoft architecture recommendations were used as a starting point [3]. CERN had indexed about 20 million documents, so it was decided that medium farm was the adequate one to baseline the design.

The final setup is built on physical servers. Instead of mixing two virtual servers into one physical, it was decided to maintain a one-on-one matching to allow for growth with the architecture remaining intact.

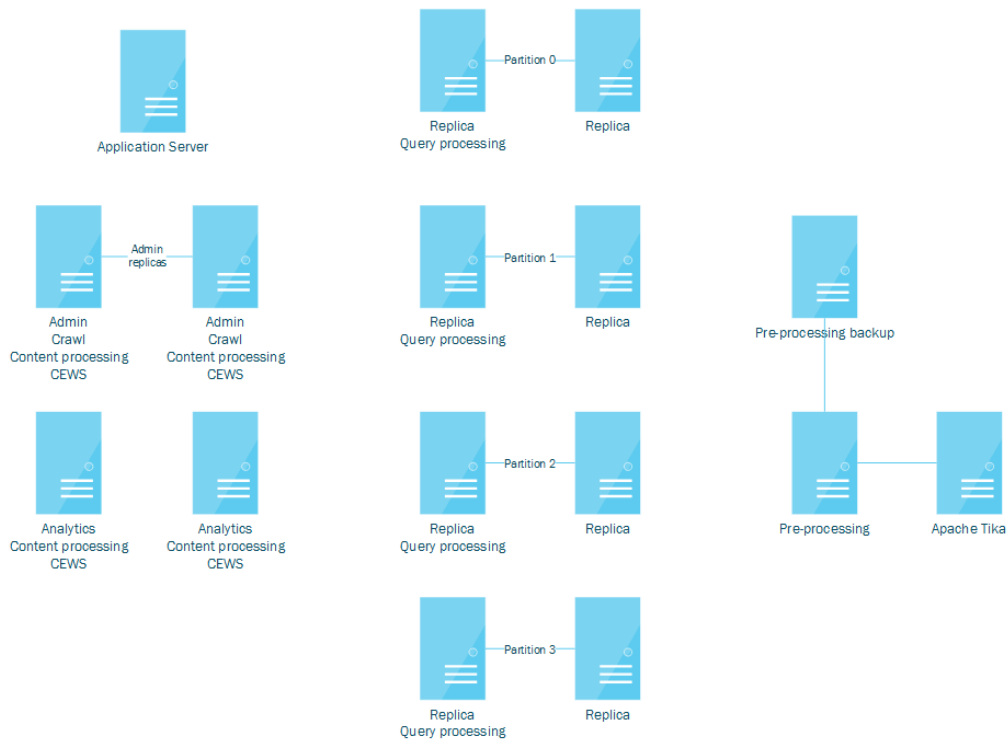


Figure 1. CERN Search Service Architecture.

Data reliability is guaranteed at two different levels: first one is done at search engine levels. By adding replicas of each partition, it is ensured that in case of a replica down, or corruption of data, data remains in at least one node while the replica recovers.

In case that all the replicas of a given partition are corrupted, the whole index will be unstable and only a complete re-indexation of all the data can be done to recover it. This is also a time consuming operation, so the service could be unavailable during a very long period. And this is not the only issue that could cause the service to be down (like maintenance operations, or the topology change it was mentioned earlier), so it was very important to guarantee availability in those cases.

That high availability was achieved having a staging infrastructure used with two purposes: the first one was the most obvious for a staging installation, which is serving as a preproduction testing infrastructure, so tests with real data (volume and contents) could be done without messing around with production; the second one is to serve as a failover farm in case the main one is not available for whatever reason.

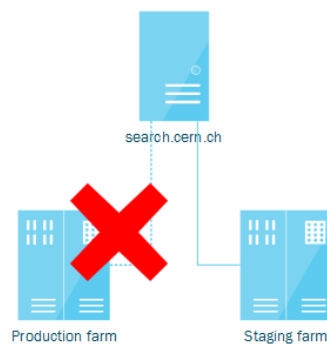


Figure 2. Overview of switching farms.

This was accomplished creating a second farm which is a replica of the production one. It is based on virtual machines, and has the same data as in production. Using *Service Application Associations* [4]

allows to switch backend farm very fast, enabling high availability even in case of complete disaster of the production farm.

Scaling out is quite easy with this architecture. In case the number of documents grow, a redesign of the topology can be done adding more partitions and replicas. This has to be carefully designed because it is a time consuming operation to perform, and could trigger more changes like adding more processors, crawlers, etc. In case the number of queries grows, the solution is simpler because it is only needed to add more query components.

3. Pre-processing

Within the pre-processing stage, two parts can be well-differentiated: Public web data and Private data.

3.1. Public web data

CERN provides an internal web sites hosting solution where all the public pages are indexed in CERN Search. In FAST for SharePoint 2010 there was an external enterprise crawler which took start URIs from configuration files. A tool was developed to fill those configuration files dynamically taking on account all site creations/deletions. In SharePoint 2013 the external crawler was removed having instead an integrated one with no configuration files. This change has triggered a complete redesign of this part.

The main constraint now is that SharePoint 2013 only allows 500 start addresses per content source [5], while there are about 7000 sites to be indexed. This situation implied a dynamic management of content sources: creating one when needed, removing one when empty, adding start addresses to an existing one, or removing start addresses from one when a site has been removed. To accomplish this, a new tool was put in place automating all the operations above.

It gets a list of websites from CERN hosting. Then they are mapped to a database including the name of the content source which they are going to belong to. To avoid fragmentation and growing number of content sources, if any of the existing content sources is not full, then it will store a new address there. If all web content sources are full, a new one is created. In case of removal, the tool will know which content source has the address and is able to remove from it. If after removing start addresses a content source becomes empty, it is deleted.

Also, this system is fully configurable, allowing to select which kind of websites are required (at CERN there is a heterogeneous offer for different types of hosting), and vary the limit of the entries per web content source (in case that the boundaries are changed [5]). Performance of this process is not an issue, because it is executed twice a day. This has been established as a good interval for having the sites indexed and up to date.

3.2. Private Data

This part was already done for the previous system and could be ported almost untouched for the update.

Basically two types of file (documents from now on) are being indexed: entities and binary files. Entities do not exist as files in the original data sources. They are abstract concepts which only have sense in the domain of the origin (like an asset, a place, an event, etc.) and they are represented as a collection of key-values in xml files. Binary files comprise all the files that exist in the content source as files. This includes pdf, doc, ppt, pptx... even xml.

The main problem with FAST for SharePoint 2010 (and SharePoint 2013) is that it is not possible to send those documents to the index directly [6] (at least at a good pace [7]). The approach taken to overcome this issue was storing the documents to be indexed in a private space, and sending them to the index using *File Shares* crawler. But in order to be able to do that, additional operations are needed, which is the part that adds complexity to this stage.

First, those documents are received through a web service. This acts as an entry point for the system and allows to control the accesses and security to the process. Content source owners push their data each time they have to create or update a document in the index. One of the key points here is to ensure that the documents are pre-processed in the same order that they come. If not, in case of two updates of the same entity arriving in the same cycle could cause that the oldest one is processed after the newest, resulting in outdated information sent to the index. This could have even access security implications.

The pre-processing then starts. Documents come in bunches of entities, so it splits them into individual ones and stores them in the private space. Now it is turn of checking if the entity has a binary file associated. In that case, it downloads it from the source, and stores it in the private space. It creates a special metadata file for each binary file downloaded, in order to set properties well when indexing.

An additional step was added with the goal of reducing redundant storage space. Binary files like ppt could come with lots of images and videos embedded on them. Those are far from useful when indexing, but take lots of space. An Apache™ Tika [8] server was set up to process those files and extract plain text and metadata from them. This way, space needed was drastically reduced (imagine a 1GB ppt reduced to 11MB).

Now that everything is properly stored, it is time to set the permissions to the file. The ACL setting was done at indexing time, but with the new version, the only way to set the ACL is using the Windows API and changing the file permissions. These will be used automatically when indexing the file setting the final ACL for searching.

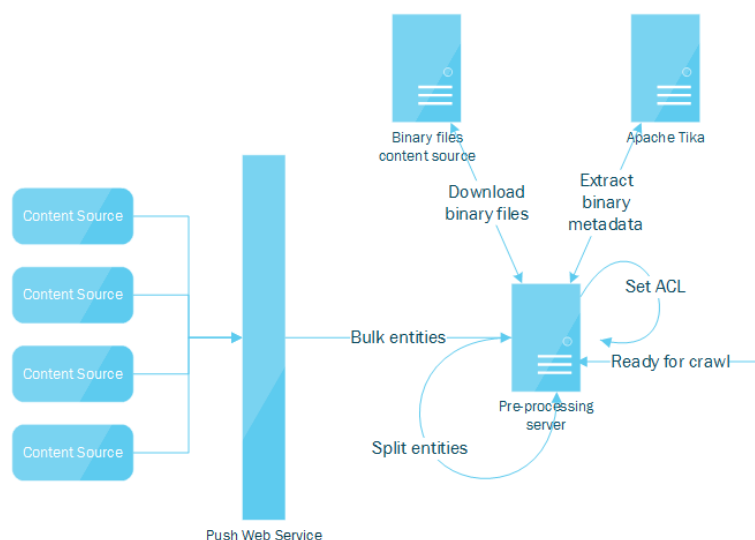


Figure 3. Pre-processing workflow

4. Indexing

It is not worth mentioning the indexation of public web data, since it is supported out-of-the-box by SharePoint 2013 and it works without any customization.

The mechanism used to customize the Content Processing is called Content Enrichment Web Service (CEWS ongoing). It is a stage of the indexing pipeline which can be configured to be called under certain conditions (it is called the trigger). It has to be implemented as a web service, with all the implications it entails (possible bottleneck of indexing, bandwidth, availability, load balancing, can be outside SharePoint environment...).

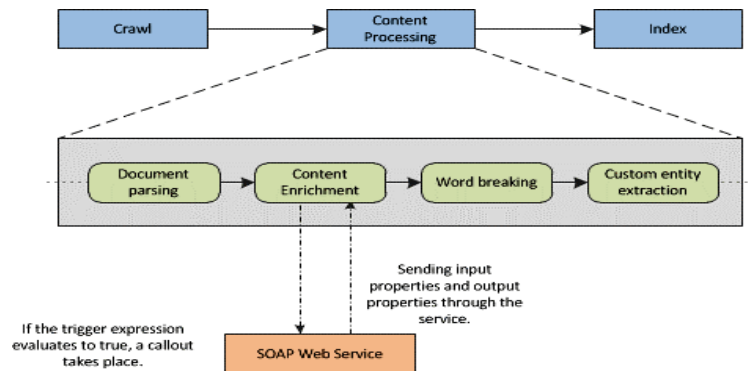


Figure 4. Custom content processing with the Content Enrichment web service callout.

But this stage cannot be avoided. This customization is needed because at this point, the document has already been parsed in the pipeline and many properties have been set with default values. As an example, in case of entities, those default values are extracted from their representational xml file having nothing to do with the real values (such as Title containing the xml filename as default value). Additional processing needs to be done here to properly set the metadata.

4.1. Scaling CEWS

To avoid a single point of failure (apart from throughput considerations), it was decided to setup CEWS on different machines. There were different options to accomplish that. But each of the standard solutions implied more complexity to the architecture, while maintainability was the most important goal.

The final approach consists in taking advantage of the combination of two properties of the system. First one is related to how documents are spread through Content Processing components. If there are more than one defined, SharePoint will try to balance the load between all of them. If the components are assigned to different machines, the documents will be then processed in different machines. Second one relates to trigger configuration. It is done at Service Application level, so it means that the configuration is the same for all Content Processing components thus being able to set up one endpoint for all. If localhost is specified as endpoint for CEWS call, it will cause that each Content Processing component will call CEWS to localhost. Configuring a basic instance of IIS in each machine which had configured a Content Processing component will allow for these architecture to work.

This results in an almost for free auto load balanced (done at Content Processing component by SharePoint) CEWS. If there is a failure in one of the web servers, all the documents in that machine will fail, which will cause SharePoint to redirect the documents to the other Content Processing components. It is not the most efficient approach, but it's good enough for CERN needs and it is very easy to maintain.

4.2. Implementing CEWS

The starting point of the implementation was Microsoft official documentation for CEWS [9]. The first problem was that several content sources with different processing needs were present in the system. To solve that, it was decided to have one main entry point serving as hub to redirect the processing to the corresponding one. Extracting the content source name, it uses it to load the proper class which is going to process those type of documents.

Once in the proper processing unit, it checks whether the document is an entity or a binary file. It only matters at this point because depending on the answer, it must read the metadata information directly from the raw data of the call (case of an entity), or go to the shared space disk to read the metadata file associated to the binary file (established at pre-processing). In any case, it will be processed then as if the precedence was the same.

Remember that at this stage of the processing binary files have been detected as that by SharePoint and it has performed some extraction operations. Therefore, the original contents of the binary file will be preserved, with the addition of certain properties required by the search service to work.

Now it is time to do the job: mapping properties. To accomplish this, it was developed the most complicated part of the whole service: the dynamic property mapper. In previous version, properties were mapped one by one in code for each content source. This caused a source of code repetition and code un-maintainability, since a bug discovered for a content source should be fixed for the rest of content sources. Always with maintainability in mind, it was decided to invest some time refactoring the code to benefit in the long term.

The way it works is by taking the mapping information from an external definition file. For each property it is defined: xpath needed to read it from the source, the destination property name, the type of data, and type of operation to perform (set or append).

It is possible to read a list of strings and append it to the previous value in the property, or append a string to an existing string, or, of course, set new dates, integers, strings or list of any of those types. All the type casting and handling is performed by the mapper. This allows new properties to be added without the need to recompile and redeploy the code: editing a text file is enough. And bug fixing is not a matter of a certain content source or property, but something global and easily detectable.

But of course there are some special properties which cannot be mapped directly, usually because they have calculated values from other properties. For these special cases it is needed to map them manually in the code. It is done after the standard properties have been mapped so it has all the values ready to be taken from.

5. Conclusion and Future Work

After having deployed the new version of SharePoint we realized several things. One of the most important ones for us is that the system is way easier to maintain. Taking on account that it was the main goal when planning the migration, we are quite satisfied to verify that it has become real.

This reduction of maintenance increased the time we had focus on other features that improve the quality of the service.

Some improvements were internal, like the automation of several operation procedures. This in turn allowed us to free more time in the long term.

Other improvements impact directly in search capacities. We added best bets, manual promotion of most important sites at CERN, define and maintain a thesaurus file with most used acronyms within the organization... All of this measures reflected in an improvement of results quality for users.

We have a clear roadmap of improvements for the future. Regarding the internal ones, there is one very needed, which is a big refactor of the pre-processing stage. There are many constraints that blocked a direct reimplementation, but once all of them are solved, a new system should be put in place.

We are working in more integrations to offer to users, like including a map in the results to help users to find locations, and a route planner to help users moving within the site.

Exciting future ahead.

References

- [1] "CERN Search Portal," [Online]. Available: <http://search.cern.ch>.
- [2] Microsoft Corporation, "FAST for Sharepoint 2010 product support lifecycle information," [Online]. Available: <https://support.microsoft.com/en-us/lifecycle?p1=14944>.
- [3] Microsoft Corporation, "Enterprise Search Architecture in SharePoint Server 2013," [Online]. Available: <https://technet.microsoft.com/en-us/library/dn342836.aspx>.
- [4] Microsoft Corporation, "Add or remove service application connections from a web application in SharePoint 2013," [Online]. Available: <https://technet.microsoft.com/en-us/library/ee704550.aspx>.
- [5] Microsoft Corporation, "Software boundaries and limits for SharePoint 2013," [Online]. Available: <https://technet.microsoft.com/en-us/library/cc262787.aspx#Search>.
- [6] R. Williams, C. Callahan, C. Givens, J. M. Gross, B. Alderman and J. Barrera, Microsoft SharePoint 2013 Administration Inside Out, Microsoft Press, 2014.
- [7] Microsoft Corporation, "Docpush reference," [Online]. Available: [https://technet.microsoft.com/en-us/library/ee943508\(v=office.14\).aspx](https://technet.microsoft.com/en-us/library/ee943508(v=office.14).aspx).
- [8] "Apache Tika," The Apache Software Foundation, [Online]. Available: <https://tika.apache.org/>.
- [9] Microsoft Corporation, "How to: Use the Content Enrichment web service callout for SharePoint Server," [Online]. Available: <https://msdn.microsoft.com/en-us/library/office/jj163982.aspx>.