Semantic Sitemaps: Efficient and Flexible Access to Datasets on the Semantic Web

Richard Cyganiak, Holger Stenzhorn, Renaud Delbru,
Stefan Decker, and Giovanni Tummarello

Digital Enterprise Research Institute (DERI),
National University Ireland, Galway

Abstract. Increasing amounts of RDF data are available on the Web for consumption by Semantic Web browsers and indexing by Semantic Web search engines. Current Semantic Web publishing practices, however, do not directly support efficient discovery and high-performance retrieval by clients and search engines. We propose an extension to the Sitemaps protocol which provides a simple and effective solution: Data publishers create Semantic Sitemaps to announce and describe their data so that clients can choose the most appropriate access method. We show how this protocol enables an extended notion of authoritative information across different access methods.

Key words: Sitemaps, Datasets, RDF Publishing, Crawling, Web, Search, Linked Data, SPARQL

1 Introduction

Data on the Semantic Web can be made available and consumed in many different ways. For example, an online database might be published as one single RDF dump. Alternatively, the recently proposed Linked Data paradigm is based on using resolvable URIs as identifiers to offer access to individual descriptions of resources within a database by simply resolving the address of the resources itself [1]. Other databases might offer access to its data via a SPARQL endpoint that allows clients to submit queries using the SPARQL RDF query language and protocol.

If several of these options are offered simultaneously for the same database, the choice of access method can have significant effects on the amount of networking and computing resources consumed on the client and the server side.

For example, a Semantic Web search engine that wants to index an entire database might prefer to download the single dump file, instead of crawling the data piecemeal by fetching individual Linked Data URIs. A client interested in the definition of only a few DBpedia [2] resources would, on the other hand, be well-advised to simply resolve their URIs. But if the client wants to execute queries over the resources, it would be better to use the available SPARQL service.

Such choices can have serious implications: For example, on February the 2nd 2007, Geonames[1] was hit by what appeared to be the first distributed denial of service attack against a Semantic Web site[2].

What happened, however, was not a malicious attack but rather a Semantic Web crawler bringing down the site by rapid-firing requests to Geonames' servers up to the point where the site's infrastructure could not keep up. One could simply judge this as a case of inconsiderate crawler behavior but, it highlights an important issue: What crawling rate should a crawler have on a Semantic Web site and would this be compatible with the size of the datasets? Considering that results are generated from semantic queries, it might be sensible to limit the query rate to one document per second, for example. It has to be noted however that crawling Geonames' 6.4M RDF documents would take 2.5 months under this condition, so that periodic recrawls to detect changes would take the same unsatisfactory amount of time and it would be impossible to have data in a fresh state.

Conceptually, the Geonames incident could have been avoided: Geonames offers a complete RDF dump of their entire database so this could have been bulk imported instead of crawling. But how is an automated crawler able to know about this dump and to know that the dump contains the entire Geonames database?

This paper describes a methodology with the primary goal to allow publishers to provide exactly such information and thus enable the smart selection of data access methods by clients and crawlers alike.

The methodology is based on extending the existing Sitemap Protocol (section 2) by introducing several new XML tags for announcing the presence of RDF data and to deal with specific RDF publishing needs (section 3).

2 The Sitemap Protocol and `robots.txt`

The Sitemap Protocol[3] defines a straighforward XML file for automatic agents (e.g. crawlers) that holds a list of URLs they should index. This is possible through tags that describe the location of each crawlable resource along with meta-information such as, for example, the expected rate of change for each individual URL or the date when this was last modified. An example of a sitemap is shown in the following listing:

```
<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">
  <url>
    <loc>http://www.example.com/</loc>
    <lastmod>2005-01-01</lastmod>
    <changefreq>monthly</changefreq>
```

[1] `http://geonames.org/`
[2] `http://geonames.wordpress.com/2007/02/03/friendly-fire-semantic-web-crawler-ddos/`
[3] `http://www.sitemaps.org/protocol.php`

```
    <priority>0.8</priority>
  </url>
</urlset>
```

Once a sitemap has been created, it must be saved in a file on the server. The protocol defines a way to extend the `robot.txt` file, so that a robot can find the location of the sitemap file on a given site.

3 The State of RDF Publishing

Even though there is no universal agreement on how information should be best published on the Semantic Web, we can characterize some of the options that are in widespread use today; list some of the limitations imposed by those options; and look at existing proposals to address these limitations.

3.1 Access methods to RDF data

Several different access methods to RDF data are in widespread use today. They vary widely along several dimensions. Two dimensions warrant closer attention, namely discoverability and cost of access.

The RDF Web and Linked Data. The traditional World Wide Web can be characterized as the HTML Web. It consists of a large number of HTML documents made available via the HTTP protocol and connected by hyperlinks. By analogy, the RDF Web consists of the RDF documents that are available via HTTP. Each RDF document can be parsed into an RDF graph. Often, different documents share URIs, and therefore merging the documents results in a connected graph. Unlike HTML hyperlinks, statements in these RDF graphs usually do not connect the URIs of documents, but instead they connect the URIs of resources described in those documents.

RDF uses URIs to identify resources of interest, but does not prescribe any particular way of choosing URIs to name resources. Experience shows that it is a good idea to choose URIs so that a Web lookup (an HTTP GET operation) on the URI results in fetching the RDF document that describes them. This effect is typically achieved by either appending a fragment identifier (e.g. `#me`) to the document URI, or by employing HTTP's `303 See Other` status code to redirect from the resource's URI to the actual document.

Datasets on the RDF Web are typically served so that resolving the URI of a resource will return only the RDF statements closely describing the resource, rather than resolving to a file containing the entire dataset. This is usually achieved by using a properly configured server which creates such descriptions on demand.

The cost of accessing this publishing method is usually relatively low. Descriptions of a single resource are usually much smaller than the description of an entire knowledge base. Similarly, the computational complexity for generating resource descriptions is limited, albeit non negligible.

RDF dumps. Like documents on the RDF Web, RDF dumps are serializations of an RDF graph made available on the Web. But they usually contain descriptions of a large number of resources in a single file, the resource identifiers usually do not resolve to the dump itself, and the file is often compressed. Many RDF datasets are published in this way on the Web and they cannot be easily browsed, crawled or linked into. It should be noticed, however, that:

- dumps are obviously useful to provide the entire knowledge behind a dataset at a single location (e.g. the Geonames dump as a single file to process it directly)
- RDF datasets that are not crawlable or do not contain resolvable resources may exist for perfectly valid technical reasons.
- Producing RDF dumps is technically less challenging than operating a SPARQL endpoint or serving Linked Data which makes RDF dumps a popular option.

Again, the cost of accessing this publishing method is computationally very low, as dumps are usually precomputed and can be easily cached. Since dumps represent the entire knowledge base, they can be however expensive in terms of network traffic and memory requirements for processing them.

SPARQL endpoints. SPARQL endpoints can provide descriptions of the resources described in a knowledge base and can further answer relational queries according to the specification of the SPARQL query language.

While this is the most flexible option for accessing RDF data, the cost of this publishing method is high:

- For simple tasks such as obtaining a single resource description, it is more involved than a simply resolving a URI. A query needs to be written according to the correct SPARQL syntax and semantics, the query needs to be encoded, the results need to be parsed into a useful form.
- It leaves a Semantic Web database open to potential denials of service due to queries with excessive execution complexity.

Multiple access methods. It is a common scheme for publishers to provide large RDF datasets through more than one access method. This might be partially explained by the early state of the Semantic Web, where there is no agreement on the best access method; but it also reflects the fact that the methods' different characteristics enable fundamentally different applications of the same data. Our proposal embraces and formalizes exactly this approach.

3.2 Current Limitations

In this section we list some limitations imposed by current RDF publishing practices which we want to address with the Semantic Sitemaps proposal.

Crawling performance. Crawling large Linked Datasets takes a long time and is potentially very expensive in terms of computing resources of the remote server.

Disconnected datasets. Not all datasets form a fully connected RDF graph. Crawling such datasets can therefore result in an incomplete reproduction of the dataset on the client side.

Scattered RDF files. To find scattered, poorly linked RDF documents, an exhaustive crawl of the HTML Web is necessary. This is likely not cost-effective for clients primarily interested in RDF, leading to missed documents.

Cataloging SPARQL endpoints. Even a full HTML Web crawl will not reveal SPARQL endpoints because support for the SPARQL protocol is not advertised in any way when requests are made to a specific SPARQL endpoint URI.

Discovering a SPARQL endpoint for a given resource. If all we have is a URI then we can resolve it to reveal some data about it. But how can we discover a potentially existing SPARQL endpoint for asking more complex queries about the resource?

Provenance. Provenance is a built-in feature of the Web, thanks to the grounding of HTTP URIs in the DNS. But control over parts of a domain is often delegated to other authorities. This delegation is not visible to the outside world.

Identifying RDF dumps. An HTML Web crawl will reveal links to many compressed archive files. Some of them may contain RDF data when uncompressed, but most of them will most likely contain other kinds of files. Downloading and uncompressing all of those dumps is likely not cost-effective.

Closed-world queries about self-contained data. RDF semantics is based on the open-world assumption. When consuming RDF documents from the Web, we can never be sure to have complete knowledge and therefore cannot with certainty give a negative answer to questions like "Does Example, Inc. have an employee named Eric?"

This list will serve a double purpose: First, it motivates several requirements for Semantic Sitemaps. Second, it serves as the basis for the evaluation of our proposal, as discussed in section 5.

3.3 Related Work

Most of the problems listed above have emerged just recently due to an increase in the amount and diversity of data available on the Semantic Web. The emergence of generic protocols for accessing RDF, especially the SPARQL protocol and Linked Data, enables the development of generic clients such as Tabulator [1] letting the user explore RDF documents anywhere on the Web. Equipped with a SPARQL API and a working knowledge of the query language, developers can interrogate any SPARQL endpoint available on the Web. As a result of these standardized protocols and tools, talking to data sources has become much easier. By contrast, efficient discovery and indexing of Semantic Web resources has

become an important bottleneck that needs to be addressed quickly. A number of protocols and ideas have been evaluated and considered to address this.

It is clear that the problem of indexing datasets is tightly linked to the problem of indexing the Deep Web. Our Semantic Sitemaps proposal "piggy backs" on the Sitemap protocol, a successful existing approach to indexing the Deep Web..

Semantic Web Clients such as Disco[4] and Tabulator [1] are also somewhat related since they could ideally make use of this specification, e.g., to locate SPARQL endpoints. Use cases to serve search engines such as Sindice [3], SWSE [4] and Swoogle [5] have been of primary importance in the development of these specifications.

There many examples of services which index or provide reference to collections of RDF documents using diverse methodologies. PingTheSemanticWeb[5], for example, works by direct notifications from data producers of every file put online or updated. SchemaWeb.info[6] offers a large repository of RDF documents submitted by users. Even Google provides a good amount of RDF files when asked specifically with a `filetype:rdf` query.

Related proposals include POWDER[7], a method for stating assertions about sets of resources. A POWDER declaration can assert that all resources starting with a certain URI prefix are `dc:published` by a specific authority. In fact, a GRDDL transform could be used to turn a Semantic Sitemap into a POWDER declaration.

Finally, the Named Graph proposal provides a vocabulary for assertions and quotations with functionalities which can be used in conjunctions with these specifications [6].

4 The Semantic Sitemaps Extension

This section introduces the Semantic Sitemaps proposal. We will start by clarifying the notion of a *dataset*, then list the key pieces of information that can be provided in a Semantic Sitemap, and finally look at two specific issues: obtaining individual resource descriptions from a large dump; and the topic of authority on the Semantic Web.

4.1 Datasets

A dataset represents a knowledge base which is a very useful notion for the following reasons:

– Different access methods, such as those mentioned above, can and should indeed be simply thought of as only "access methods" to the same knowledge base.

[4] `http://sites.wiwiss.fu-berlin.de/suhl/bizer/ng4j/disco/`
[5] `http://pingthesemanticweb.com/`
[6] `http://schemaweb.info`
[7] `http://www.w3.org/2007/powder/`

– The knowledge base is more than just the sum of its parts: by itself, it is of informational value whether a piece of information belongs to a dataset or not and queries using aggregates can make this explicit. For example, the question "Is `Berlin` the *largest* city mentioned in DBpedia?" cannot be answered by only getting the description of the resource itself (in case of Linked data though resolving their `Berlin` URI). On the contrary, such queries can be answered only the entire content of the dataset is available.

The Semantic Sitemap extension has the concept of dataset at its core: Datasets are well defined entities which can have one or more access methods. It is well defined what properties apply to a certain access method and what properties apply to a given dataset. Therefore properties that apply to that dataset will be directly related to all the data that can be obtained, independently from the access method.

This statement of existence of an underlying dataset implies that in case of multiple offered access methods, then they will provide information consistent with each other. The specification mandate that this in fact must be the case, with the exceptions only limited to:

– operational issues such as delays in the publication of dumps.
– information that pertains only to a certain access method, such as `rdfs:seeAlso` statements that link together the documents of a linked data deployment.

A publisher can host multiple datasets on the same site and can describe them independently using different sections of the Semantic Sitemap. While there is nothing that prevents information overlap or contradictions between different datasets, it is expected that this is not the case.

4.2 Adding dataset descriptions to the Sitemap protocol

The Semantic Sitemap extension allows the description of a dataset via the tag `<sc:dataset>`, to be used at the same level as `<url>` tags in a regular sitemap. Access options for the datasets are given by additional tags such as `<sc:dataDump>`, `<sc:sparqlEndpoint>` and `<sc:linkedDataPrefix>`. If a sitemap contains several dataset definitions which are treated independently. The following example shows a sitemap file applying the extension.

```
<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9"
        xmlns:sc="http://sw.deri.org/2007/07/sitemapextension">
  <sc:dataset>
    <sc:datasetLabel>
      Example Corp. Product Catalog
    </sc:datasetLabel>
    <sc:datasetURI>
       http://example.com/catalog.rdf#catalog</sc:datasetURI>
    <sc:linkedDataPrefix sc:slicing="subject-object">
```

```
      http://example.com/products/</sc:linkedDataPrefix>
    <sc:sampleURI>
      http://example.com/products/widgets/X42</sc:sampleURI>
    <sc:sampleURI>
      http://example.com/products/categories/all</sc:sampleURI>
    <sc:sparqlEndpoint sc:slicing="subject-object">
      http://example.com/sparql</sc:sparqlEndpoint>
    <sc:dataDump>
      http://example.com/data/catalogdump.rdf.gz</sc:dataDump>
    <sc:dataDump>
      http://example.org/data/catalog_archive.rdf.gz</sc:dataDump>
    <changefreq>weekly</changefreq>
  </sc:dataset>
</urlset>
```

The dataset is labeled as the *Example Corp. Product Catalog* and identified by `http://example.com/catalog.rdf#catalog`. Hence it is reasonable to expect further RDF annotations about the dataset `http://example.com/catalog.rdf`.

The "things" described in the dataset all have identifiers starting with `http://example.com/products/`, and their descriptions are served as Linked Data. A dump of the entire dataset is available, split into two parts and the publisher states that dataset updates can be expected weekly.

RDF dataset dumps can be provided in formats such as RDF/XML, N-Triples and N-Quads (same as N-Triples with a fourth element specifying the URI of the RDF document containing the triple; the same triple might be contained in several different documents). Optionally, dump files may be compressed in GZIP, ZIP, or BZIP2 format.

4.3 Other elements

Other interesting elements in the specifications include:

<sc:sparqlGraphName>. If this optional tag is present, then it specifies the URI of a named graph within the SPARQL endpoint. This named graph is assumed to contain the data of this dataset. This tag must be used only if `<sc:sparqlEndpointLocation>` is also present, and there must be at most one `<sc:sparqlGraphName>` per dataset. If the data is distributed over multiple named graphs in the endpoint, then the publisher should either use a value of `*` for this tag, or create separate datasets for each named graph. If the tag is omitted, the dataset is assumed to be available via the endpoint's default graph.

<sc:datasetURI>. An optional URI that identifies the current dataset. Resolving this URI may yield further information, possibly in RDF, about the dataset, but this is not required.

<sc:datasetLabel>. An optional label that provides the name of the dataset.

<sc:sampleURI>. This tag can be used to point to a URI within the dataset which can be considered a representative sample. This is useful for Semantic Web clients to provide starting points for human exploration of the dataset. There can be any number of sample URIs.

4.4 Defining the DESCRIBE operator

Publishing an RDF dataset as Linked Data involves creating many smaller RDF documents, each served as a response when accessing the URI itself via HTTP.

This is usually the result of a SPARQL DESCRIBE query performed giving as argument the resource's identifier. But the SPARQL specification does not specify how DESCRIBE queries are actually answered: it is up to the data provider to choose the amount of data to return as a description.

It is important that the Semantic Sitemap provides a way to specify how this description process is most likely to happen. Knowing this enables a process that we call "slicing" of the dataset, i.e. turning a single large data dump into many individual RDF models served online as description of the resource identifiers. This process is fundamental for creating large indexes of online RDF documents descriptions, as illustrated in section 5.1.

For this, the `<sc:linkedDataPrefix>` and `<sc:sparqlEndpointLocation>` tags can have an optional `sc:slicing` attribute taking a value from the list of slicing methods below and which mean that the description of a resource X includes:

- `subject`: all triples whose subject is X.
- `subject-object`: all triples whose subject or object is X.
- `CBD`: the Concise Bounded Description [7] of X.
- `SCBD`: the Symmetric Concise Bounded Description [7] of X.
- `MSG`: all the Minimal Self-Contained Graphs [8] involving X.

Publishers that want to use a slicing method that is not in the list should pick the value that most closely matches their desired method, or they may omit the `sc:slicing` attribute. If the slicing method is very different from any in the list, it is recommended to publish a dump in N-Quads format.

4.5 Sitemaps and Authority

While there is no official definition of authoritative information specifically for the Semantic Web, many currently agree on extending the standard definition given in *Architecture of the World Wide Web, Volume One* [9], which links authority to the ownership of the domain name in which the URIs are minted.

For example, a piece of information coming from the owner of the domain `example.com` would be considered authoritative about the URI `http://example.com/resource/JohnDoe`. Following this definition, any information obtained by resolving the URI of information resources served as Linked Data can be considered authoritative, as it comes from the same domain where the URI is hosted.

However, no mechanism is currently defined to get information about such an authority: The only possibility for this is via DNS domain records, which are outside the actual Web architecture.

For this reason the Semantic Sitemap extension proposes an `<sc:authority>` element, which is used at the top level of a sitemap file to specify a URI identifying the person, organization or other entity responsible for the sitemap's URI space. This is only useful if the URI is resolvable and yields additional information about the authority. The authority URI has to be within the sitemap's URI space, which makes the authority responsible for providing their own description.

The semantics of `<sc:authority>` is such that, given an authority URI a, for any document d within the sitemap's URI space, there is an implied RDF statement d `dc:publisher` a. For example, if a sitemap file in the root directory of `http://example.com/` declares an `sc:authority` of `http://example.com/foaf.rdf#me`, then the entity denoted by that URI is considered to be the publisher of all documents whose URI starts with `http://example.com/`. It has to be kept in mind that publication does not necessarily imply endorsement; it merely means that a is responsible for making d available. To express a propositional attitude, like assertion or quotation, additional means are necessary, such as the Semantic Web Publishing Vocabulary [6].

Delegation of authority. The boundaries of DNS domains do not always match the social boundaries of URI ownership. An authority sometimes delegates responsibility for chunks of URI space to another social entity. The Semantic Sitemap specification accounts for this pattern. If another sitemap file is placed into a sitemap's URI space, then the second sitemap's URI space is considered to be not under the authority of the first. For example, publishing a sitemap file at `http://example/~alice/sitemap.xml` delegates the subspace rooted at `http://example.com/~alice/`.

Furthermore, `robots.txt` must link to a sitemap index file that in turn links to both of the sitemap files. This ensures that all visitors get identical pictures of the site's URI space. The approach is limited: It only allows the delegation of subspaces whose URIs begin with the same URI as the secondary sitemap file.

This feature proves to be very useful in properly assigning authority to Semantic Web information published in popular URI spaces like `purl.org`.

Joining URI spaces. Besides partitioning of URI spaces, it is sometimes necessary to join URI spaces by delegating responsibility of additional URIs to an existing sitemap. This is particularly useful when a single dataset spans over multiple domains, for example, if the SPARQL endpoint is located on a different server. The joining of URI spaces is achieved by placing a sitemap file into each of the spaces and having the `<sc:subSitemap>` element point to the other's URI. The sub-sitemap also needs a reciprocating `<sc:parentSitemap>` element. This prevents fraudulent appropriation of URI spaces.

Authoritative descriptions from SPARQL endpoints and dumps. An RDF description of a URI u is considered authoritative if resolving u yields an RDF

document containing that description. An authoritative description is known to originate directly from the party owning the URI u, and thus can be considered to be definitive with regard to the meaning of u.

Providing authoritative information thus requires the publication of RDF in the "RDF Web" style. Descriptions originating from SPARQL endpoints or RDF dumps cannot be considered authoritative, because it cannot be assumed that information about u from a SPARQL endpoint at URI e is indeed authorized by the owner of u. The presence of a Semantic Sitemap, however, changes this. If e and u and d are in the same sitemap's URI space, then they originate from the same authority, and therefore we can treat information from the SPARQL endpoint as authoritative with respect to u even if u is not resolvable. This has two benefits. First, it allows RDF publishers to provide definitive descriptions of their URIs even if they choose not to publish RDF Web-style documents. Second, it allows RDF consumers to discover authoritative descriptions of URIs that are not resolvable, by asking the appropriate SPARQL endpoint or by extracting from the RDF dump.

5 Evaluation

We now revisit the challenges identified in section 3.2 and will show how the Semantic Sitemaps proposal can address each of them.

Crawling performance. Crawling large Linked Data deployments takes a long time and is potentially very expensive in terms of computing resources of the remote server. An RDF publisher can make a dump of the merged Linked Data documents available and announce both the `<sc:linkedDataPrefix>` and `<sc:dataDump>` in a sitemap file. Clients can now discover and download the dump instead of crawling the site, thus dramatically reducing the required time.

Disconnected datasets. Not all datasets form a fully connected RDF graph. Crawling such datasets can result in an incomplete reproduction of the dataset on the client side. Sitemaps address this in two ways. Firstly, clients can again choose to download a dump if it is made available. Secondly, by listing at least one ¡sc:sampleURI¿ in each component of the graph, the publisher can help to ensure a complete crawl even if no dump is provided.

Scattered RDF files. To find scattered, poorly linked RDF documents, an exhaustive crawl of the HTML Web is necessary. This is likely not cost-effective for clients primarily interested in RDF, leading to missed documents. Site operators can provide an exhaustive list of those documents using `<sc:dataDump>` or `<sc:sampleURI>`.

Cataloging SPARQL endpoints. Even a full HTML Web crawl will not reveal SPARQL endpoints, because support for the SPARQL protocol is not advertised in any way when requests are made to a SPARQL endpoint URI. However, if a sitemap has been provided, the crawler can discover the services via `<sc:sparqlEndpoint>`.

Discovering a SPARQL endpoint for a given resource. If all we have is a URI, we can resolve it to reveal some data about it. But to discover a potentially existing SPARQL endpoint for asking more complex queries we can look for a sitemap on the URI's domain and inspect it for `<sc:sparqlEndpoint>` elements.

Provenance. Provenance is a built-in feature of the Web, thanks to the grounding of HTTP URIs in the DNS. But control over parts of a domain is often delegated to other authorities. Semantic Sitemaps allow site operators to make this delegation visible, by appropriate placement of multiple sitemap files, and also by employing `<sc:subSitemap>` and `<sc:parentSitemap>` elements.

Identifying RDF dumps. An HTML Web crawl will reveal links to many compressed archive files. Some of them may contain RDF data when uncompressed, but most of them will most likely contain other kinds of files. Downloading and uncompressing all of those dumps is likely not cost-effective. Semantic Sitemap files explicitly list the locations of RDF dumps in the `<sc:dataDump>` element and therefore allow crawlers to avoid the cost of inspecting dumps that turn out not to contain any RDF.

Closed-world queries about self-contained data. RDF semantics is based on the open-world assumption and hence, when consuming RDF documents from the Web, we can never be sure to have complete knowledge. So we cannot with certainty give a negative answer to the question "Does Example, Inc. have an employee named Eric?". Datasets defined in Semantic Sitemaps are natural candidates for applying local closed-world semantics. Drawing on `<sc:datasetLabel>`, this allows us to give a stronger answer: "Example, Inc. has not given information about such an employee in the dataset labeled *Example, Inc. Employee List*."

5.1 Processing of Large RDF Dumps

A major motivation for our proposal is its potential to reduce the cost, both in raw time and computing resources, of indexing the largest RDF datasets currently available on the RDF Web. These datasets are usually also available as RDF dumps, and Semantic Sitemaps enable clients to download the dump and avoid crawling. This section presents the results of some experiments into quantifying the benefits of this approach and showing the general feasibility of processing very large RDF dumps. We employ the Hadoop framework[8] for parallel data processing on a "mini" cluster of low cost machines (single core, 4gb ram).

The issue with crawling large collections of RDF documents from a single host, such as the UniProt datasets available on `http://purl.uniprot.org`, is that a crawler has to space its request by a reasonable amount of time in order to avoid overloading the server infrastructure. Semantic Web servers are likely

[8] `http://lucene.apache.org/hadoop/`

to be experimental and not optimized for high load. Therefore submitting more than one request per second seems to be unadvisable for RDF crawlers.

The UniProt dataset[9] contains approximately 14M RDF documents. A full crawl at this rate would take more than five months, and a full re-crawl to detect updated documents would take another five months.

UniProt provides a dump (over 10 GB in size, compressed). The time for gathering the data can thus be reduced to downloading of the dump files, plus subsequent local data processing to slice the dumps into parts that are equivalent to the individual documents. Each part describes a single resource. The parts can be passed on to our indexing system grouped in .TAR.GZ files each containing 10000 individual RDF files.

The first step in our processing was to convert the dumps from the RDF/XML format to N-Triples. The line-based N-Triples format can be more easily processed with Hadoop's off-the-shelf classes. We observed processing rates of about 60k triples/s for this format conversion. We did not attempt to further optimize or parallelize this step.

For the actual slicing of the dumps we used Hadoop's MapReduce [10] facility. In the mapping phase, each triple is sorted into buckets based on the URIs mentioned in the subject and object position. In the reduce phase, the triples in each bucket are written to a separate file.

To gather some proof-of-concept results, we first processed the file `uniref.nt` (270M triples) on a single machine, then two machines and then four. We then ran a test on the complete UniProt dataset (1.4B triples) on four machines.

# Machines	1	2	4	4
# Triples	272M	272M	272M	1.456M
Conversion	1:17:01	1:19:30	1:19:10	6:19:28
MapReduce	10:04:35	5:18:55	2:56:54	16:36:26

These preliminary results show that processing of a very large RDF dump to obtain slices equivalent to the documents that are being served as description of the resources using the Linked Data paradigm is feasible. The processing task can be parallelized, and the results indicate that, unlike with the crawling approach, the data retrieval step is unlikely to be a bottleneck.

6 Adoption

Currently, the Sitemap Specification is currently available in its 5th release[10]. The specification creation process involved a great deal of interaction with data

[9] available from `ftp://ftp.uniprot.org/pub/databases/uniprot/current_release/rdf`

[10] `http://sw.deri.org/2007/07/sitemapextension/`

producers and developers alike. Most of this interaction happened in specialized Mailing Lists and trough technical workshop dissemination [11].

The process has so far been very interactive, with data producers proposing features and clarifications they considered important such as, for example split datasets for DBpedia, and details on how to use a sitemap on shared domains like purl.org.

Thanks to this direct interaction, adoption at Data Producer level has so far been very satisfactory. Most large datasets provide a Semantic Sitemap and in general we report that data producers have been very keen to add one when requested given the very low overhead and the perceived lack of negative consequences.

At consumer level and as discussed earlier, the Sindice Semantic Web indexing engine adopts the protocol [3] and thanks to it has indexed, as today, more than 26 million RDF documents.

We can report that the SWSE Semantic Web Search Engine [4] will also soon be serving data obtained thanks to dumps downloaded using this extension.

7 Future Work

While it is unlikely that the current specifications will change profoundly, we envision that future versions of the Semantic Sitemaps will address issues such as: Data published in formats such as RDFa and using GRDDL transformations, datasets available in multiple versions, enhance SPARQL endpoint descriptions, mirrored datasets and copyright and legal related information.

8 Conclusion

In this paper we have discussed an extension to the original Sitemap Protocol to deal with large Semantic Web datasets. We have highlighted several challenges to the way Semantic Web data have been previously published and subsequently showed how these can be addressed by applying the proposed specifications, thus allowing both clients and servers alike to provide and important novel functionalities.

We have further verified the feasibility of an important use case: server side "dump splitting" to efficiently process and index millions of documents from the Semantic Web. We have showm that this task can be performed efficiently and in a scalable fashion even with modest hardware infrastructures.

9 Acknowledgements

Many people have provided valuable feedback and comments about Semantic Sitemaps including Chris Bizer (Free University Berlin), Andreas Harth (DERI Galway), Aidan Hogan (DERI Galway), Leandro Lopez (independent), Stefano

Mazzocchi (SIMILE - MIT), Christian Morbidoni (SEMEDIA - Universita' Politecnica delle Marche), Michele Nucci (SEMEDIA - Universita' Politecnica delle Marche), Eyal Oren (DERI Galway), Leo Sauermann (DFKI)

References

1. Berners-Lee, T.: Tabulator: Exploring and Analyzing linked data on the Semantic Web. In: Procedings of the The 3rd International Semantic Web User Interaction Workshop. (2006)
2. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: Dbpedia: A nucleus for a web of open data. In Aberer, K., Choi, K.S., Noy, N., Allemang, D., Lee, K.I., Nixon, L., Golbeck, J., Mika, P., Maynard, D., Mizoguchi, R., Schreiber, G., Cudré-Mauroux, P., eds.: The Semantic Web. Volume 4825 of LNCS., Springer (2007) 722–735
3. Tummarello, G., Oren, E., Delbru, R.: Sindice.com: Weaving the open linked data. In Aberer, K., Choi, K.S., Noy, N., Allemang, D., Lee, K.I., Nixon, L.J.B., Golbeck, J., Mika, P., Maynard, D., Schreiber, G., Cudr-Mauroux, P., eds.: Proceedings of the 6th International Semantic Web Conference and 2nd Asian Semantic Web Conference (ISWC/ASWC2007), Busan, South Korea. Volume 4825 of LNCS., Berlin, Heidelberg, Springer Verlag (November 2007) 547–560
4. Harth, A., Hogan, A., Delbru, R., Umbrich, J., O'Riain, S., Decker, S.: Swse: Answers before links! In: Semantic Web Challenge 2007, 6th International Semantic Web Conference. (2007)
5. Ding, L., Finin, T., Joshi, A., Pan, R., Cost, R.S., Peng, Y., Reddivari, P., Doshi, V., Sachs, J.: Swoogle: a search and metadata engine for the semantic web. In: CIKM '04: Proceedings of the thirteenth ACM international conference on Information and knowledge management, New York, NY, USA, ACM (2004) 652–659
6. Carroll, J., Bizer, C., Hayes, P., Stickler, P.: Named Graphs. Journal of Web Semantics **3(4)** (2005) 247–267
7. Stickler, P.: CBD - Concise Bounded Description. `http://www.w3.org/Submission/CBD/` (2005) Retrieved 09/25/2006.
8. Tummarello, G., Morbidoni, C., Puliti, P., Piazza, F.: Signing individual fragments of an rdf graph. In: WWW '05: Special interest tracks and posters of the 14th international conference on World Wide Web, New York, NY, USA, ACM (2005) 1020–1021
9. Jacobs, I., Walsh, N.: Architecture of the World Wide Web, Volume One - W3C Recommendation. `http://www.w3.org/TR/webarch/` (2004) Retrieved 09/25/2006.
10. Dean, J., Ghemawat, S.: Mapreduce: Simplified data processing on large clusters. In: OSDI'04: Proceedings of the 6th conference on Symposium on Operating Systems Design and Implementation. 137–150
11. Tummarello, G.: A sitemap extension to enable efficient interaction with large quantity of linked data. Presented at W3C Workshop on RDF Access to Relational Databases (2007)