Search Metadata

Storage and Sharing Considerations

Robert Schuler and Ann Chervenak

Background

In the federated ESG architecture currently being developed, metadata had been divided into two classes. Each gateway will maintain a full metadata catalog (likely using a relational database) for the data sets published at that gateway. In addition, a subset of the metadata that is needed for data discovery, called *search metadata,* will be harvested from the full metadata catalog and shared with the other gateways. (Search metadata may be stored as RDF triples in a triple store.) Thus, each gateway will receive search metadata from all the gateways, so that a client making a discovery query at any gateway can locate the gateway(s) that host the data sets of interest. In addition, the client may conduct more detailed searches based on the full metadata catalog at the gateway that manages the desired data sets.

In earlier meetings and email threads, we discussed the need for a sharing mechanism for the search metadata. The sharing mechanism would allow the search metadata to be collected from each gateway and propagated in some manner to the other gateways so that users at each gateway could find any collection of data throughout the collaborating ESG sites. We discussed the possibility of using RDF for search metadata, exploring OAI as a sharing mechanism, and also considering generalization of the RLS state sharing and indexing mechanism for sharing the search metadata.

This document presents some considerations for the sharing mechanism between the gateways and the global services.

Storage and Exchange Format

We may use the same or different data formats for storage and exchange. For instance, we may store the search metadata in a relational database yet exchange it in RDF format. Some RDF repositories take this approach. Alternately, the search metadata could be stored in RDF also.

Time stamps

Each record in the search metadata repository, whether stored in RDB, RDF, or XML, will need a minimal set of time stamps associated with it in order to share the metadata among sites. For each record, we will need to know when the record was created and when it was last updated. Depending on the sharing mechanism, we may also need to know if it was deleted.

Master or Multi-Master

In our federated use cases document from October 2007, we assumed a structure with a single master in a master-secondary replication scheme where a global service acts as the master and each gateway acts as a secondary site.

More recently, the ESG architecture has moved toward a multi-master scheme, where each gateway acts as the master site for the data collections that it manages and the related metadata, including the search metadata for those data sets. In this scheme, each gateway replicates the search metadata for which it is the master at the other gateways. In addition, each gateway acts as a secondary site for search metadata that is managed at other gateways.

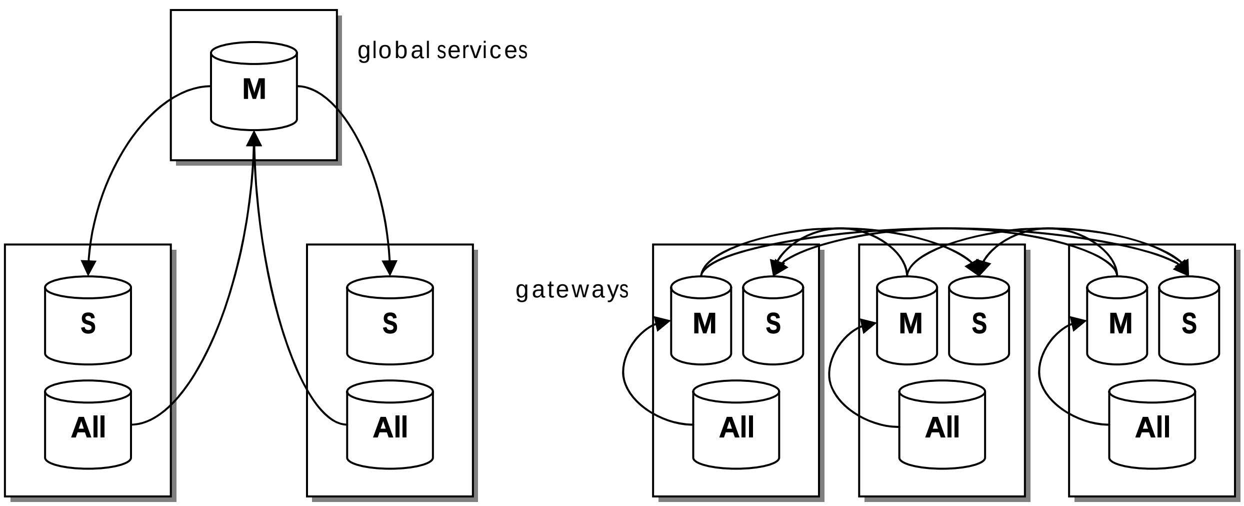

Figure 1: left, each gateway updates a master copy of search metadata at a global service, then the master copy is replicated to each gateway as a secondary copy; right, each gateway updates its own master copy and then each gateway's master copy of its search metadata is replicated to the other sites as secondary copies. All, the full collection of metadata; M, the master copy of search metadata; and S, the secondary copy of search metadata.

Update topology

In the case of the single master approach (see Figure 1, left), a natural update topology is a star configuration, with updates harvested at each gateway and sent to the single master, and then secondary copies replicated to each gateway. In the case of the multi-master approaches (see Figure 1, right), the topology could be a dense, fully-connected graph (as depicted) or a sparsely-connected, spanning tree in a P2P fashion.

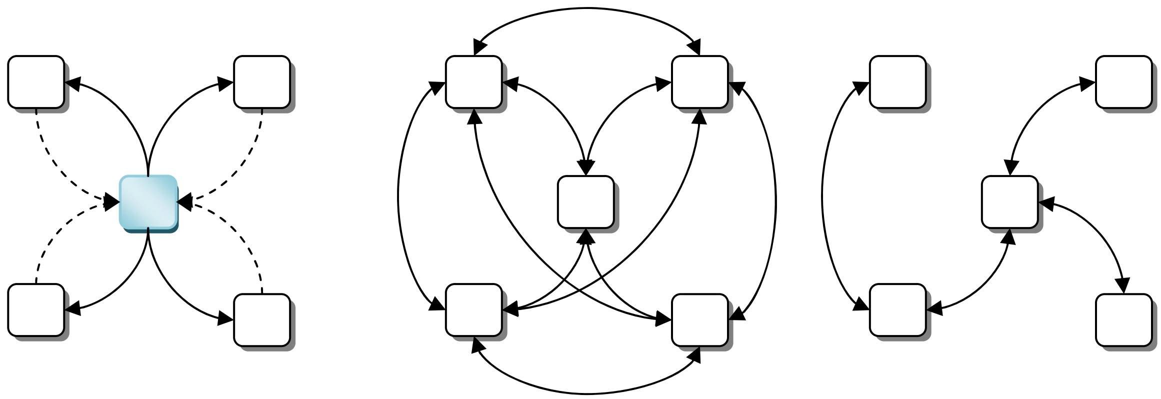

Figure 2: left, a star topology for updates where each gateway (clear boxes) updates a global service (shaded box) and then replicas are pushed back to the gateways; center, a fully-connected update graph where each gateway sends updates to all other gateways; right, a sparsely connected update graph where updates are propagated along shorter edges.

Update frequency

Depending on the characteristics of the gateways, we may consider different update frequencies for the master site (or sites) to update the secondary sites. If we expect many small updates to the metadata, for instance from interactive user edits, then small frequent updates may be appropriate. If we expect large batch-driven changes to the metadata then a scheduled bulk update may be more appropriate. The sensitivity to the lag time between metadata changes and accurate reflection at secondary sites must also be considered.

Soft or Hard State

The secondary copy of search metadata at each site (in any master-secondary scheme or update topology structure) can be considered soft or hard state. If it is a *soft state*, then by definition, the search metadata is considered stale at some point and expires. Thus, the search metadata needs to be refreshed periodically in a soft state scheme. One consequence is an increase in network communication among sites.

Alternatively, if the search metadata is considered *hard state*, then we expect the secondary copies to remain valid unless notified by the authoritative (master) site that they should be invalidated.

In the case of soft state, we can avoid the need to propagate “deleted” search metadata entries because they will be purged from the secondary sites eventually. Another advantage of soft state is that if a gateway goes offline, its secondary copies will expire. When the gateway reconnects, its search metadata can be quickly reconstructed using normal soft state update mechanisms.

Compression

The Replica Location Service uses soft state mechanisms along with Bloom filter compression to minimize the size of updates exchanged among RLS local catalogs and index servers. One important consideration is that Bloom filter compression is a lossy compression scheme. It performs hash functions and sets corresponding bits in a bit map. So, for the RLS example, it is impossible to use a Bloom filter to retrieve the logical file names that were used to create the Bloom filter. Queries on the Bloom filter are performed by hashing the name and seeing whether the corresponding bits are set.

Bloom filter compression may not be an appropriate technique for use with search metadata, since it may be desirable to share entire RDF triples among the gateways, rather than just bitmaps representing hashed values for those triples.

We will try to evaluate the size of search metadata to be exchanged to determine whether the size of these updates is prohibitive. In addition, we will explore the use of lossless compression schemes for RDF triple information.

Full vs. Incremental Updates

Related to the frequency of updates and the type of information exchanged among gateways is the question of whether full or incremental updates are appropriate. It may be desirable to send incremental updates describing recent changes to search metadata. Alternatively, if soft state is used, then search metadata will expire, and full updates must be performed periodically to refresh the search metadata.

Software

Some existing software may provide at least part of the solution that ESG needs. These include Open Archives Initiative (OAI) based software or RDF repositories. OAI specifies the OAI Protocol for Metadata Harvesting (OAI-PMH). It specifies a client-server architecture where "harvesters" (the client) request updated metadata from "repositories" (the server). Using OAI implementations, each ESG gateway might setup an OAI repository for their search metadata and then other sites (other gateways or the global service) would run a harvester to collect metadata updates. Alternatively, open source RDF repositories exist. "Boca" is a RDF repository developed by IBM and released as an open-source project. It supports a central RDF repository with clients that can cache the repository contents locally. ESG could explore the use of Boca where each gateway (or the global service) runs a RDF repository and the other gateways act as clients of the repository.

- OAI Tools: http://www.openarchives.org/pmh/tools/tools.php

- Boca: http://ibm-slrp.sourceforge.net/2006/11/20/boca-the-rdf-repository-component-of-the-ibm-semantic-layered-research-platform/