

A Flexible Replica Catalog Framework Within The Globus Environment

In the paper I co-wrote with Heinz Stockinger, we presented a flexible and efficient method to handle replica catalog entries. The method relied on HTTP redirection protocol to convert a replica catalog logical url to a physical url that bound that entry to a specific server that could supply the data associated with the url.. This allowed a site to relocate data at will without engaging the replica catalog with updates. While this essentially distributed the catalog it also provided what we called “administrative scalability” in that a site could perform administrative functions on replicated data virtually in isolation; thus, providing for a no more complicated environment than existed without the notion of replication.

Furthermore, we suggested that the HTTP redirection protocol be handled at the client level since, in our view, it was far more flexible for the client to make decisions on which replica was the most suitable one. That is, we were predisposed to driving the decision making to the lowest possible level. Since that time, various alternative “administratively scalable” schemes have been proposed. Thus far, all of these rely on central action (e.g., replica servers or ftp proxy servers). While we feel that these alternatives come close to addressing various scalability issues they still fall far short of being sufficiently flexible. This is especially significant for evolving grid environments. As a case in point, I present a notion of “virtual replication” that has the potential of further scaling grid environments without adding additional administrative overhead.

Virtual Replication

In virtual replication, the recipient of a url does not necessarily have beforehand knowledge of what that url actually represents other than a source of data of interest. Currently, the commonly accepted definition for a replica catalog entry ties a single entry to a single data stream, usually a file. I propose that this mapping is far too restrictive and a framework needs to be developed that will allow extended mapping to be investigated.

For instance, a url within a replica catalog could represent a collection of files. Replication in this case would be based on collections not files. Each site that chose to host a replica would be responsible for locally maintaining that collection. Thus, as the collection changed over time, there would be no need to update the replica catalog. The notion of replicating collections, as opposed to files, is not a substantially new idea, and several people have proposed that managing collections on a global scale is, in many ways, easier and far more scalable than managing individual files.

Another possibility is that a url can represent a computation that needs to be performed in order to obtain the data of interest. Such schemes can be grouped under the rubric of virtual data. Regardless of how the data is obtained, a mechanism must be developed that is sufficiently flexible so as not to preclude future research on logical as well as physical replication strategies.

In short, the grid environment needs to be able to accommodate diverse definitions of

replicated data; some of which have not yet been envisioned. Furthermore, it needs to do so in a way that allows new replication strategies to be widely tested, easily deployed, and done in a manner that does not impact the current production environment.

Given that the current grid environment of choice is Globus, I propose a modest extension to the current replica catalog processing that would make virtual replication possible.

Dynamic Replica Catalog Protocols

I suggest that the current replica catalog processing scheme be left in tact. However, when the catalog returns a url, the client-side catalog interface first check if the returned protocol identifier is one of the commonly supported types (e.g., gridftp). If it is not, one of the supported types, the interface searches a set of defined directories for a shared library that can handle the protocol. The first such library encountered is dynamically loaded into memory and the url is forwarded to a well-defined method for re-writing the url to a form that yields a commonly accepted protocol. The method is free to return any number of re-written urls. This provides for a 1-to-n mapping function essential for supporting virtual collections. If a suitable protocol cannot be found the process terminates with an error.

For instance, the environmental variable `globus_REPCAT_PATH` can define the list directories to be searched, with a suitable default provided. The form can be similar to that used by the loader (i.e., `LD_LIBRARY_PATH`). The dynamic library interface (i.e., `dlopen()`), supported by practically all systems currently in use, can be used to manage the shared libraries. Libraries are named in the form of `librc_protocol.so` where *protocol* is the name of the replica catalog protocol that the library defines.

This framework is very flexible. To test a new protocol simply requires the installation of a shared library in the library search path. To deploy a new protocol for large-scale testing is equally simply. In fact, new replication schemes can be developed and tested without impacting the current working globus environment.

The down-side to this proposal is two fold:

- 1) it involves modifications to the client-side replica catalog interface, and
- 2) requires that each client that wants to use a new replication strategy have access to the appropriate shared library.

I suggest that first down-side is not only minor but also necessary. First, the client-side changes need to occur only once. New protocols do not require any additional changes to the client api. They only require the installation of the appropriate protocol. Furthermore, only the client side is best suited to handle arbitrary url mappings. For instance, an ftp proxy server would not be able to handle a 1-to-n mapping while a client can trivially do so. While it would be possible for the replica catalog server to handle dynamic protocols, doing so would make it difficult, if not impossible, to use many commercial database systems as the basis of a replica catalog system. However, nothing in this proposal

precludes moving the concept of virtual replication into the catalog server, even though I strongly believe that doing so would result in a much less flexible framework.

The second down-side is potentially more serious since it potentially requires wide-spread distribution of reasonably synchronized code. However, solutions for similar code distribution problems are well known. Therefore, it would seem that the benefits of using dynamic replica catalog protocols outweigh the distribution problems.