LAVOISIER: A DATA AGGREGATION AND UNIFICATION SERVICE

S. Reynaud, G. Mathieu, P. Girard, F. Hernandez, IN2P3/CNRS Computing Centre, France
O. Aidel, CS-SI, France

Abstract

It is broadly admitted that grid technologies have to deal with heterogeneity in both computational and storage resources. In the context of grid operations, heterogeneity is also a major concern, especially for worldwide grid projects (LCG, EGEE…). Indeed, the usage of various technologies, protocols and data formats induces complexity, which increases the risk of unreliable code, lack of reactivity and a extremely low level of reusability.

To reduce these risks, we need an extensible tool for reliably aggregating heterogeneous data sources, for executing cross data sources queries and for exposing the collected information for several usages.

In this paper we present "Lavoisier", an extensible service for providing an unified view of data collected from multiple heterogeneous data sources. This service transforms the collected data and represents them as XML documents: this allows for transparently and efficiently executing XSL queries on them. The service also exposes the data through standard protocols and interfaces (WSRF). In addition, its efficiency can be optimized by tuning the provided cache mechanisms according to both the characteristics and the usage profile of the data coming out of each source (access frequency, amount of data, latency, ...), and the consistency of cached data can be ensured by specifying their inter-dependencies.

We also present some cases where Lavoisier has proven effective, like the usage that is now done by the "LCG/EGEE CIC portal" a central tool for the operations of the LCG/EGEE grid infrastructure.

RELATED WORK

Software systems dealing with heterogeneous data (in terms of data sources, data models, data formats and protocols for accessing them) are inherently complex. A classical approach for significantly reducing this complexity is to use a common format to represent the data and to use a unified mechanism to make queries on the data collected from each source.

Like other comparable tools, Lavoisier uses XML as the common data format. One major advantage of this format is the open data model it proposes, and the fact that the various tools we need to process the data exist and use open standards such as XML Schema [4], XSLT [5], XPath [6], XQuery [7], etc.

Several other tools relying also on XML exist. Xcalia Intermediation Platform [9] offers a standard JDO [8] interface and can be queried with the JDO Query Language (JDOQL). The XQuery language is used as the mechanism to access to the data by some systems such as BEA AquaLogic Data Services Platform [10], Tamino [11], e-XMLMedia software suite [12], GAEL's Data Request Broker API [13] and XQuare Fusion [14].

Besides not all of them being open source, the main drawback of all those tools is that the proposed query languages (XQuery or JDOQL) are not the more appropriate for all use cases, such as the one presented below. In order to be able to use other languages (e.g. XSL) than those supported, we would need to introduce intermediate data structures and queries, which would lead to inefficiency and complexity.

In addition, most of the data sources latencies are not acceptable for our use case. We therefore need to locally cache collected data, while managing dependencies between them to ensure the coherency of the aggregated view. No known solution offers this feature.

OVERALL SERVICE ARCHITECTURE

The Figure 1 presents the four layers of the Lavoisier service architecture: the heterogeneous data sources, the adapters, the views manager and the Web Service interface.

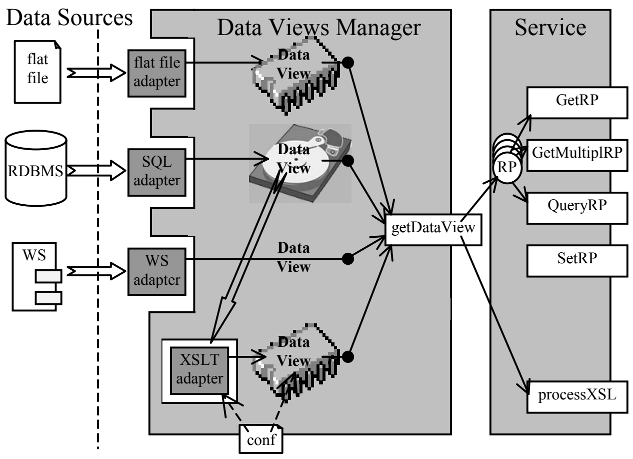

Figure 1 : Overall architecture of Lavoisier

In general terms, **Data Sources** might be any kind of system providing access to data which are expected to be exposed by the Lavoisier service. There is no requirement on their location, format, protocol for accessing the data nor the technology used for implementing the source. For example, it could be a file system, a data base management system, a web service, or even the Lavoisier service itself.

Adapters are plug-ins which interact with data sources and produce a XML document as their output. This XML document represents a view of the queried data source. Several reusable adapters have been already implemented to facilitate the integration with common technologies used for accessing data. Using the existing adapters is just a matter of configuration. The current provided reusable adapters include:

- The JNDI [19] adapter which is currently tested with LDAP sources only. It is a generic adapter, which is specialized (e.g. into an LDAP adapter) by choosing both a JNDI driver and a JNDI to XML mapping description file.
- The SQL adapter, based on Qizx/open [20]. It supports simultaneous connections to several relational database management systems. This adapter can be configured using an extended version of the XQuery language.
- The flat (text) file adapter which uses pattern matching rules to generate its XML output.
- The Local XML file adapter which loads and parses the content of a locally (from the Lavoisier service point of view) available XML file, and can notify Lavoisier when this file is modified.
- The Remote XML file adapter which downloads and parses the content of a remote XML file. It currently supports HTTP and HTTPS protocols with X509 certificate-based authentication.
- The Local XML directory adapter which represents a XML files tree as a single XML document by converting directories to XML elements.
- The XSL Transform adapter which applies a XSL stylesheet on a XML data source. In particular, this adapter could apply the transformation on one or more Lavoisier's data views to create another data view. This is very convenient to maintain an aggregated view of different data sources.
- Data View Index adapter which enables Lavoisier introspection by providing information about each configured data view managed by Lavoisier itself.

The **data views manager** layer is responsible for invoking the adapters to provide corresponding XML data views, managing the views cache when required, and maintaining cached data up to date according to the views dependencies and cache update triggering rules (see the Service Administrator role below). Caching the data of a particular view is optional; if no cache is configured for a data view, its adapter is invoked each time the data view is requested by a client application of Lavoisier.

Finally, **Lavoisier service** layer provides its client applications the interface to request XML data views. As this interface is a WSRF-compliant [17] **Web Service interface**, it makes the service easy to request from any application.

ROLES WITH LAVOISIER

As a result of such a decomposition in layers, Lavoisier enables a clear separation in three roles: the plug-ins developer role, the service administrator role and the user role.

The data consumer role

As presented above, Lavoisier exposes its services as standard WS-Resource Properties [17] operations. By exposing its services in a standardized way, client applications of Lavoisier can be based on standard tools. For instance, commands from the Globus Toolkit 4.x [18] can be used for consuming data aggregated by the service.

Moreover, the additional 'processXSL' web service operation (see Figure 1) allows the consumer application to submit a XLS stylesheet for processing on several data views and get the results in one of several possible formats (XML, HTML or plain text). The submitted stylesheet is included in the parameters of the operation.

The service administrator role

The service administrator is responsible for configuring both the existing generic adapters and the new adapters provided by the developer. Each adapter has its own configuration syntax and content. For example, most adapters may at least require the service administrator to configure the location of the data source.

Configuring the data cache management policy for each particular data view is also a task of the service administrator. This includes choosing the cache location for the data view (memory or disk) and defining the set of rules triggering cache updates.

When the data view's cache is located in memory, it is represented as a DOM tree and no further processing is needed when it is accessed. When the data view's cache is located on disk, it is stored in a single XML file or as a tree of XML files ("hierarchical" cache).

Updating the cache for a particular view can be triggered by several types of events and can be configured for each view. The service administrator can configure a view so that its cache is updated:

- when the data view is initialized,
- when a time-based event occurs ("cron-like"),
- when the data source is updated through Lavoisier,
- when the data source is updated by an external actor and the adapter is notified of it,
- when the data view is about to expire (proactive update),
- when the data view is both accessed and expired,
- when the cache of another data view has been refreshed.

This later rule allows for ensuring coherency between caches of several data views. Indeed, this rule does not only enable describing cache dependencies, it also supports delaying the exposition of the new content of a view until some of its dependant view's cache is fully updated. This may be used, for example, to keep the cache of a particular data view coherent with the cache of another data view containing index information about the first one.

These advanced configuration possibilities offered by Lavoisier allows the service administrator to tune data view access efficiency according to the characteristics and the usage profile of both the data source (e.g. amount of data used to build the view, update frequency, latency) and its corresponding data view (e.g. total amount of data, time to live of the content, expected maximum latency). This fine-tuning is performed without impacting adapter developers (see below) and consuming applications.

The administrator may also provide the XML Schema describing the structure of the XML generated for each data view. In addition, the provided schemas can be exposed as a new data view in order to help the consumer application in building its queries. The schemas can also be used to automatically validate the XML data generated by the adapters.

The adapter developer role

Lavoisier has been designed to be extensible. This can be done by adding plug-ins that we call "adapters". It is the responsibility of the adapter developer to provide new adapters if the provided ones do not satisfy his specific needs.

An adapter must implement one of the read interfaces defined within the framework: the DOMAdapter interface expects the adapter to return the generated XML view as a DOM [16] tree, while the StreamAdapter interface expects the adapter to send it to an output stream. Both interfaces may be implemented if the provided implementation is more efficient than a basic DOM tree serialization or de-serialization.

An adapter may require some additional information to be provided in order to generate the XML view. This information is passed to the adapter as arguments. Finally, an adapter may implement additional interfaces in order to provide optional capabilities. Those optional features include updating the data source through Lavoisier and notifying Lavoisier when the data source has been updated without it.

EXAMPLE USE CASE: THE LCG/EGEE DAILY OPERATIONS PORTAL

The LCG[1]/EGEE[2] daily operations web portal (also know as CIC portal) [15] is a tool designed to be a single entry point for all the operational needs of grid actors (end-users, virtual organisation managers, site managers, regional operation centre's managers, etc.). It provides up-to-date information on the operational status of the whole grid, by extracting and composing information from several sources.

The information is presented by the portal in different views, each one intended for a particular actor. As a consequence, the same data have to be presented in several ways according to the actor's role. The portal retrieves data from many heterogeneous sources (relational databases, LDAP servers, flat files, R-GMA [21], web services...) and executes cross queries to build high level views.

The initial implementation of the portal was written as a set of scripts (PHP, Perl and shell), each dedicated to a particular task. This approach presented a very low level of reusability and happened to become more and more complex as long as the portal's functionalities evolved (see Figure 2). The great amount of specific code rapidly made the portal difficult to maintain. It became then clear that a radically different approach was needed. This was the starting point and motivation for developing the Lavoisier service.

Lavoisier was progressively introduced in the architecture of the CIC web portal. The first use case was to use it to aggregate data from several different relational data bases for building the weekly operational reports of each EGEE geographical region. The SQL adapter was used to query the GOC-DB [22] (the central data base for grid resources information), the SFT [23] results database (output of grid functional tests run on several components of each grid site) and the portal's own database.

Since then, Lavoisier is being used for other tasks. A good example of the smooth migration enabled by the modular design of Lavoisier is the creation of a high level view of resources and services available per virtual organisation. In this use case, data extracted from various heterogeneous sources need to be aggregated to provide a single coherent overall view including the list of sites of the grid and their grid components (computing elements, storage elements, resource brokers, …), the status of each component as seen by the SFT [23] automated testing service and the information system monitoring service (GIIS monitor [24]). As a first step, the legacy mechanism (cron tasks) was still used for generating the XML data. The Remote XML file adapter was then used to retrieve the generated data, and the initial code for displaying them was progressively replaced by XSL stylesheets applied on the retrieved data. The view's cache was then configured and the cron tasks were no longer needed. Finally, the adapters for querying LDAP-based directories, flat files and relational databases were smoothly introduced and the original code was no longer used (see Figure 3). All these modifications were performed in a very short time and this part of the portal's code is now one of the easiest to maintain and to extend.

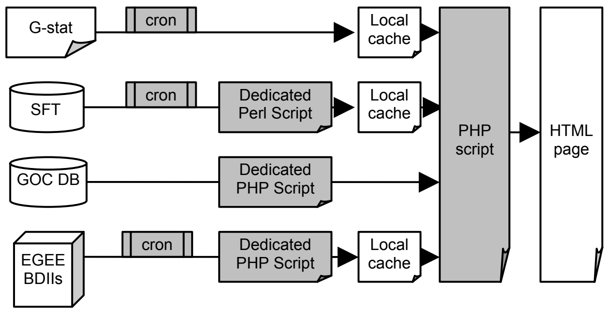

Figure 2 : Simplified view of the CIC web portal, before using Lavoisier

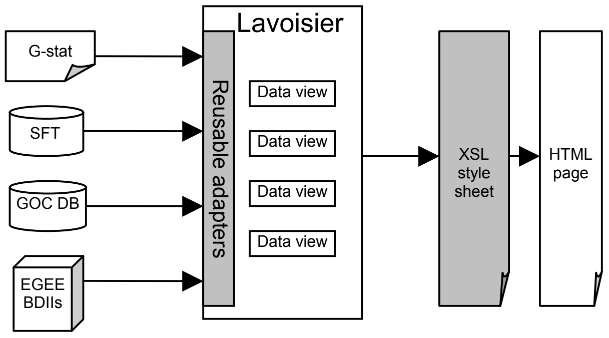

Figure 3 : Overall architecture of the CIC web portal showing the role of Lavoisier

Lavoisier has been also used for generating other views of operational information provided by the portal and integrated from data provided by several external sources ([26] , [27]).

These examples show that the approach of separating roles within Lavoisier not only eases maintenance of the code but also allows for a gentle integration of it in existing services.

FUTURE WORK

We are developing some new adapters to support specific data sources, such as WSRF-compliant [17] web services and R-GMA [21] service. We will also extend our Remote XML file adapter capabilities to support building a data view from an entire tree of XML files instead of a single XML file.

Some additional adapters for generating new data views from other existing views will be included, such as an XQuery adapter and a view composer adapter. The later will not be as flexible as the XSL Transform and XQuery adapters, but its main advantage will be its capability to dynamically generate the XML stream of the composed view, without creating any intermediate XML document. This feature will make it suitable for aggregating several large data views into a single XML document. As a consequence, this adapter may be, in some cases, more efficient if its result is not kept in cache.

Improving the cache mechanisms is also in the development roadmap of the service. This is especially relevant for hierarchical caches, for supporting incremental updates and tolerating partial failures when updating hierarchical caches. Both simple and hierarchical caches will support automatic retry of failed updates.

In a longer term perspective, we consider developing additional cache mechanisms for data views, such as a LRU (Least Recently Used) cache, if needed by some future use cases.

CONCLUSION

Lavoisier has proven effective in increasing maintainability of the LCG/EGEE CIC portal. Indeed, the unified view enabled reducing considerably the amount and complexity of its code. The Lavoisier design made easier writing reusable code and is features allowed the portal developers to considerably increase the performance by appropriately tuning the cache mechanisms in an absolutely transparent way for the portal code.

Moreover, we have observed that Lavoisier can be easily and progressively integrated to an existing application.

Our Lavoisier instance can be used by several client applications in the LCG/EGEE scope. Finally, the Lavoisier service is not restricted to grid-related applications; it can also be deployed and used in several other contexts.

REFERENCES

[1] http://lcg.web.cern.ch/LCG/.
[2] http://public.eu-egee.org/.
[3] http://www.w3.org/XML/.
[4] http://www.w3.org/XML/Schema.
[5] http://www.w3.org/TR/xslt.
[6] http://www.w3.org/TR/xpath.
[7] http://www.w3.org/TR/xquery/.
[8] http://java.sun.com/products/jdo/.
[9] http://xcalia.com/products/platform.jsp.
[10] http://www.bea.com/content/products/aqualogic/data_services/index.htm.
[11] http://www.tamino.com/.
[12] http://www.e-xmlmedia.fr/.
[13] http://www.i-space.fr/fps/fps_lbz08.htm.
[14] http://xquare.objectweb.org/fusion/index.html.
[15] http://cic.in2p3.fr/.
[16] http://www.w3.org/TR/1998/REC-DOM-Level-1-19981001/.
[17] K. Czajkowski, D. F. Ferguson, I. Foster, J. Frey, S. Graham, I. Sedukhin, D. Snelling, S. Tuecke and W. Vambenepe, “The WS-Resource Framework”, 2004.
[18] http://www.globus.org/toolkit/.
[19] http://java.sun.com/products/jndi/.
[20] http://www.xfra.net/qizxopen/.
[21] http://www.r-gma.org/.
[22] http://goc.grid-support.ac.uk/.
[23] https://lcg-sft.cern.ch:9443/sft/lastreport.cgi.
[24] http://goc.grid.sinica.edu.tw/gstat/.
[25] http://gus.fzk.de/.
[26] H.Cordier, G.Mathieu, J.Novak, P.Nyczyk, F.Schaer, M.Schulz and M.H.Tsai, "Grid Operations: the evolution of operational model over the first year", CHEP, 2006.
[27] T.Antoni, F.Donno, H.Dres, G.Mathieu, P.Strange, D.Spence, M.H.Tsai and M.Verlato, "Global Grid User Support: the model and experience in the Worldwide LHC Computing Grid", CHEP, 2006.