MONITORING CMS TRACKER CONSTRUCTION AND DATA QUALITY USING A GRID/WEB SERVICE BASED ON A VISUALIZATION TOOL

G. ZITO, M.S. MENNEA, A. REGANO
Dipartimento Interateneo di fisica di Bari & INFN sezione di Bari, Italy

ABSTRACT

The complexity of the CMS Tracker (more than 50 million channels to monitor) now in construction in ten laboratories worldwide with hundreds of interested people, will require new tools for monitoring both the hardware and the software. In our approach we use both visualization tools and Grid services to make this monitoring possible. The use of visualization enables us to represent in a single computer screen all those million channels at once. The Grid will make it possible to get enough data and computing power in order to check every channel and also to reach the experts everywhere in the world allowing the early discovery of problems . We report here on a first prototype developed using the Grid environment already available now in CMS i.e. LCG2. This prototype consists on a Java client which implements the GUI for Tracker Visualization and two data servers connected to the tracker construction database and to Grid catalogs of event datasets. All the communication between client and servers is done using data encoded in xml and standard Internet protocols.

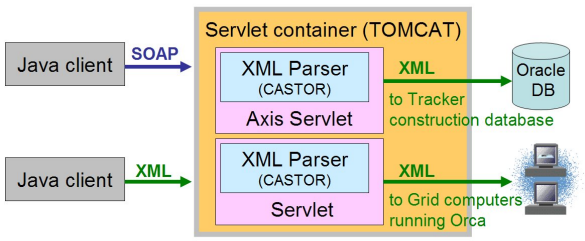
1. Why Web Service

The use of web services is essential to ensure the interoperability of our visualisation application with different data sources. Some of these sources (the event store) are on the Grid and their use requires Grid services which are based on Web services. Other data sources are relational databases. In both cases **Web services let you establish a standard protocol for data exchange between disparate data sources and a consumer.** They serve as a layer of abstraction isolating from the details of data representation, exchange and query. For example the visualisation client handles in the same way tracker data coming from the construction relational data base and from a grid dataset.

2. System Architecture

The system consists of a highly portable lightweight client capable of running on many different platforms connected to Internet.

The server application works as a gateway in direct contact with the construction database or the event store where standard analysis program



(Orca – Cms reconstruction project) write their data. The connection between client and server uses standard internet protocols. To extract tracker data from xml files we have used the technique called xml data binding using the free source program Castor [1].

3. Data Server Implementation

To implement the data server we used Tomcat [2] + Axis [3]. Tomcat can be used as a normal web server with http protocol.
In order to test both the two most popular approaches to web services we have implemented the data server In two different ways:

1 To get events from Montecarlo federations we use the so called REST approach to web services. This consists in naming all resources provided with standard web addresses (URI) and using the commands already available in http protocol (i.e. GET, PUT, DELETE, POST) to manage the resources. (Tomcat implements all of them). The client requests events or other data to the server by putting the information about the request directly in the URL with the GET directive. The server answers by sending the data in the XML format.

2 To access the data in the construction database we use instead Axis (working as a Tomcat servlet) to provide web services following the W3C definition with SOAP and WSDL. In this case both the request and the answer are sent in a SOAP envelope.
The service itself is described in a WSDL file.

Up to now the data servers can provide data from two different sources :

- federations of Montecarlo events for the full detector.
- data from the construction data base

A data server providing real data from a prototype in a test beam is foreseen in the future but not yet implemented.

4. Visualization Client

The visualization client, which we named Tmon, is implemented in Java. It can execute on any computer with Java Runtime 1.4 installed. This allows the use of our software from any computer connected to Internet without the need to install special software. This client communicates with data servers using standard Internet protocols. All files exchanged contain data encoded in xml.

This client implements the user graphics interface and fulfills these main tasks:

- Once the user has selected a data server, the client requests the tracker description to the server.
- Interprets the data in the xml file that describe the tracker, configuring its graphics interface for the specific tracker.
- Allows the user to request new events or data from construction database.
- Displays the information sent in a 2D window representing either the complete tracker or the single layer.
- Allows the possibility to print the graphical representation on the screen.

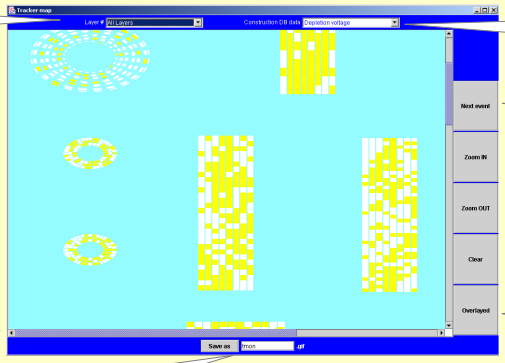
MONITORING CMS TRACKER CONSTRUCTION AND DATA QUALITY USING A GRID/WEB SERVICE BASED ON A VISUALIZATION TOOL

5. GUI of Client Prototype

The user can choose to see all the tracker or only one of the 41 layers.

It is possible to select as data source the construction database. Many kind of data can be downloaded.

The main representation for monitoring is a 2D representation where you can see the whole detector with each one of 17000 modules visible. A kind of tracker map.



Request of a new event

It is possible to zoom when a single layer is displayed

The tracker map can be examined using the two scrollbars

Possibility to represent the information in two way: overlayed (not shown here) and separated. See 4 for a detailed explanation.

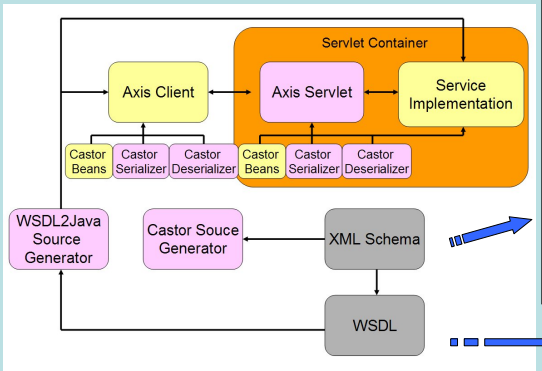
The representation can be printed as .gif

6. Xml data binding with Castor

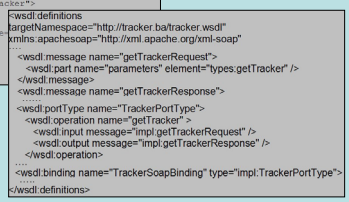
Castor uses the Xml-schema description of the tracker or the event data model to generate authomatically the java classes implementing it (Castor beans).

The program WSDL2Java (provided with the Axis distribution) used the WSDL description of the Web service to generate the Java code implementing it.

Serialization/Deserialization (converting data to/extracting data from) xml files is done by Castor.



Elements created by hand



7. Performance

The data servers are run on a PC with a CPU Pentium IV. We have tried the visualisation client in Java on many Linux and Windows PC with Java Runtime 1.4 installed. The time needed to fetch the description of the whole tracker and unmarshal it is 10 seconds. This time is reasonable taking in account that the xml file sent is more than 1 Mb large and the Java program has to validate and load the data. Reading a single event or the data from the construction database will take a time that depends on the data size but never exceeds 10 seconds. This time is obtained of course because both reading events and data from database are done by servlets that work as gateway to the real data provider caching the result of query.

8. Lesson learned and conclusion

This was first of all a didactic exercise to get ready for the Grid. Infact the next step is to trasform the web service in a Grid service. Implementing web services requires learning to use a plethora of xml dialects and languages. Also making Tomcat, Axis and Castor working toghether is rather tricky.
Anyhow after a steep learning curve the software used has proven to be reliable and with a good performance.

Is monitoring a detector like CMS tracker feasible from Internet? Yes when CMS develops a standard Web and Grid interface to its monitoring data. Developing our application we have found that the access to construction data base was very easy because Oracle databases have a standard web interface. Instead the access to CMS event data was difficult because CMS has yet no web interface defined and we had to invent it from scratch by, for example, defining a XML format of CMS events. In our opinion, developing such standard web interface, is absolutely essential to CMS especially if it wants to fully exploit the capabilities of the Grid.

References

1. http://www.castor.org
2. http://jakarta.apache.org/tomcat/
3. http://ws.apache.org/axis/
4. Mennea,M.S; Regano,A; Zito,G. "*CMS Tracker Visualisation*", CMS-NOTE2004/009; Geneva, CERN, 08Jun2004