A SCIENTIFIC RESEARCH RESULTS MANAGEMENT SYSTEM I. ARHITECTURE

MIRONESCU Ion-Dan

University "Lucian Blaga" of Sibiu

Abstract:
The exponential growth of the amount of data generated by the scientific research requires the implementing of a data management system. The article describe the development of such a system based on Open Source software for a small sized (and budgeted) research team . This first part concentrates on the system architecture.

1.INTRODUCTION

The fast growing amount of data that results from the activities of even a small sized research team impose the use of a data management system in order to provide:

- reliable and secure storage of large amount of data from multiple source;
- concurrent access to the stored data;
- access control system;
- support for metadata association (at least time stamp and source URI) and metadata-based data retrieval;

The high cost of a commercial solution has made it in the past unreachable exactly for these small teams. The rise of reliable and competitive (proved in commercial use!) Open Source software and the effort toward open standards (like the XML related ones) give the possibility to implement such solution with in-house teams. Although not negligible the cost of implementation are affordable.

2. CURRENT PRACTICE

The figure 1 describe the actual form of "data management". Data are produced as result of two type of research activities:

- experimental measurement – data is in the proprietary (legacy) format of the data acquisition system and is stored locally. Where possible, transformation to a more "universal" format (CSV, xls) is performed locally – often conducing to data redundancy. Metadata (time, author, facility used, experimental setup etc.) is "stored" in the file system (directories and files names), as separate descriptive text file or on paper so that the data retrieving based on metadata is rudimentary. In this form distributed and concurrent access to data and metadata, without the direct support of the data "owner" is impossible;

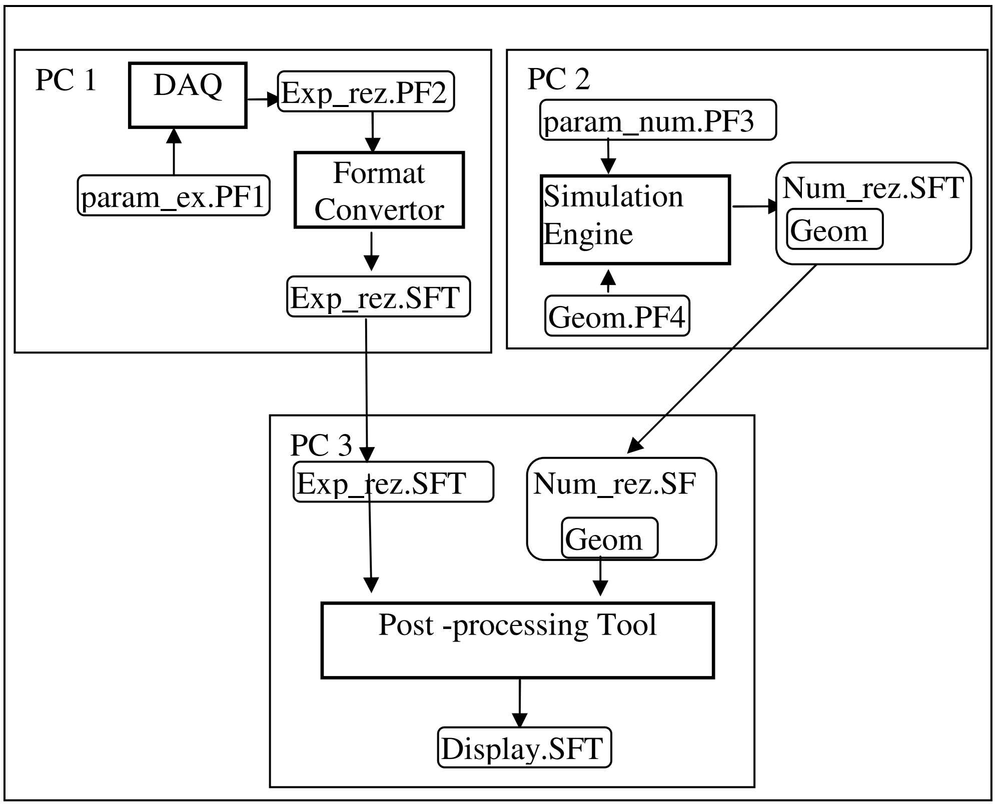

Figure 1: Current data workflow

- numerical simulation - the simulation data (initial solution, boundary condition, simulation parameter) is prepared with proprietary pre-processing tools; the simulation results are stored direct in formats that proprietary post processing tool can read from; in this form for using other pre- or post-processing tool a special export or import routine must be developed. Often this routine also performs a transformation (for example interpolation) on the data so that raw data are lost. Because the post processing file must also be self-contained, some information (for example discretisation mesh) is stored redundant. Metadata handling is similar to that for experimental data. The same access limitation applies.

The data resulting from the two activities must be often combined (for example for validating the result of a simulation). With data located on different computer, stored in different (often incompatible) physical formats this is a supplementary time and resources consuming task.

3. THE PROPOSED SOLUTION

For such a system a three tiered solution seems to be optimal:

- a storage back-end - in most cases a relational RDBMS
- a front-end – that provide access and visualisation of stored data
- a middle tier that interconnect the two layer and also provides the interface between the data producers and the RDBMS

If “use cases“ make easier the choice - based in first place on the system requirements - of the back-and front-end, no standard solutions or guide lines for the implementation of the middle tier exist due the great variety of proprietary (and often legacy) format used by the data producers (data acquisition and simulation system). The emerging of the eXtensible Mark-up Language (XML) and XML derived languages (Harold et al.2002) as a standardised interchange format seems to indicate the way to go.
The middle tier application of choice due to the requirements is the Apache Web server. The open plug-in architecture permits to flexible extend and customise the server-side functionality without hawing to modify the main Apache source Code. The proposed architecture make use of this mechanism for:

- transforming the HTTP server in a Distributed Authoring and Versioning (WebDAV) server allowing to the data produces to put their data and metadata in a central repository in a familiar way(mod_dav module) (Kim 2004) ;
- replacing transparently the traditional file based storage with a Data Base storage that support searching more efficiently (mod_dav_dbms module) (Kim 2004)
;

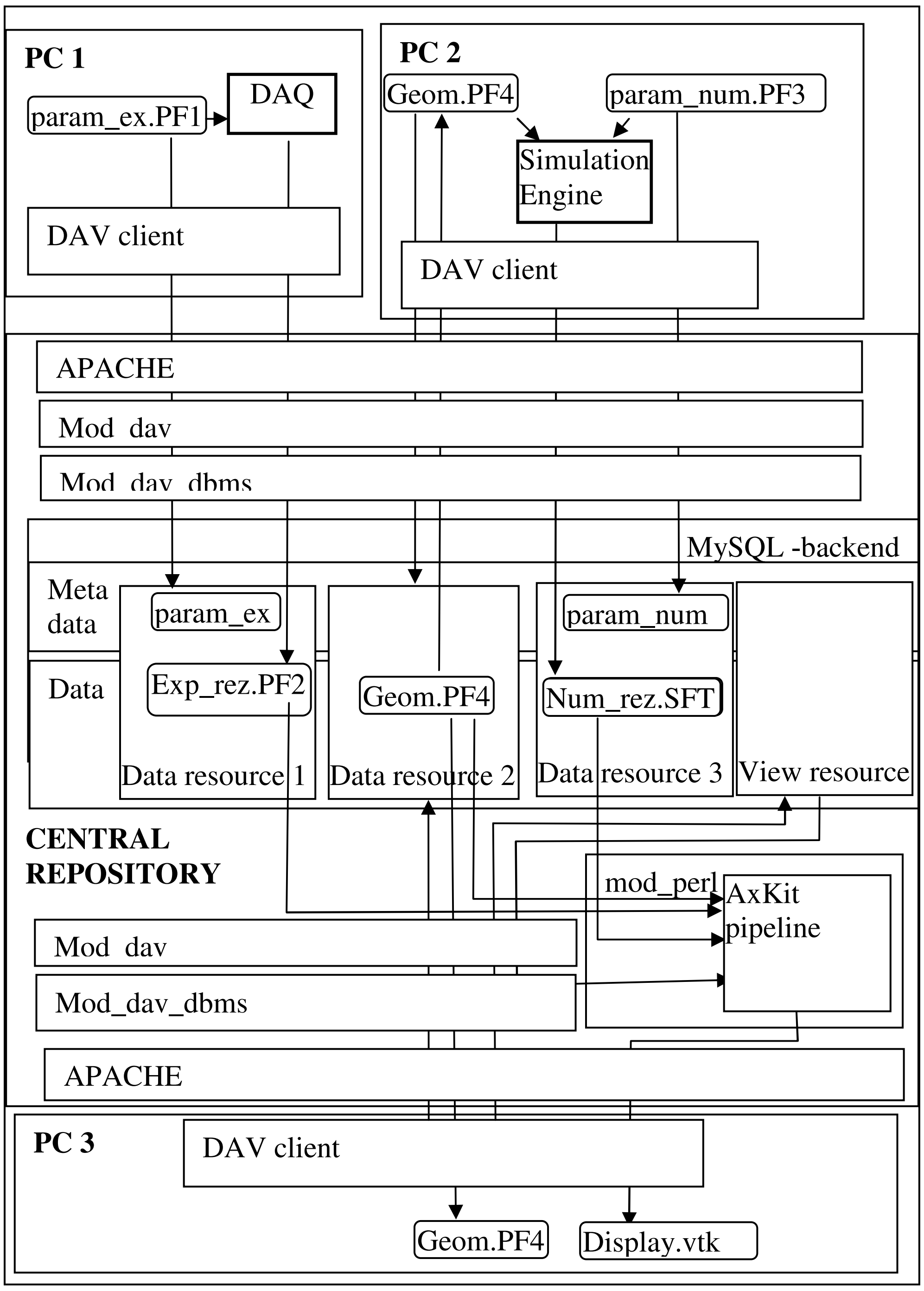

Figure 2: Structure of the data management system

- providing customisable transformation pipelines (XSLT or Perl based) from and to XML/non-XML formats (mod_perl + AxKit).(Bekman 2003) (Ray 2003)

The back-end is in this case a MySQL database.
Standard DAV enabled applications can be used for managing the data and the tools of choice (so long a open description of the used format exists) for visualising them on the client-side. The path can be shorten by either combining a visualisation library (for example the VTK based package MayaVi) with DAV functionality (for example from the PyDAV library) or using a plug-in (for example for X3D) with a standard WebDAV enabled browser.

The use of the system implies two activities: publishing and accessing data
<u>Publishing data</u>
From user point of view this is as simple as moving a file to a local folder. The browser can map the central data repository as a network drive so that standard file system operation as create, copy, move and rename are possible. As result a data resource is created in the central repository.
The metadata include information on user, creation time, data source and format but can also include information on the experimental setup or the simulation case that generated the data and can be automatically generated either by the browser on basis of template system and user provided data or on the server side on basis of a registered profile.

<u>Accessing data</u>
The data can be accessed in two modes:

- raw mode – that can be done in the same mode as the publishing by simply copying back the data on the local system;
- view mode – the data is filtered server-side to obtain a particular view of the data in a particular format. For this a view description resource must be created in the repository. This resource contains two category of information:
    - the reference to the data resource or resources (when the view is based on multiple data sources;
    - the information needed to configure and drive a AxKit pipeline of transformations that include SAX and XSLT based filter;

When a copy of such a view resource is requested from the server the transformation pipe line is fired and the server returns a document that integrate the data in a new format.

The simplest transformation is one that only transforms data of a single resource from a storage format to a suitable display format (for example from CSV to a HTML table). On the opposite end of the complexity scale are views that combine data from many sources (for example time series of measured data with geometry data) in a complex format such as the VTK XML format

For each data resource a XDTM (Moreau 2004) abstract dataset description must be provided.

If the conversion is only from one binary format to another, the filter parses this description and by using mappings (in XDTM/DLDF) (Beckerle 2003) from this data set description to the physical representations from / to which the conversion take place reads the data from one source and writes it down to the other. Whenever the conversion takes place to an XML derived format the first filter produce the XML representation of the abstract data set from which a XSLT based filter produce the desired format.

CONCLUSION

The presented architecture improves the access to the data by providing:

- a central repository with distributed access, access control system and a better searching system
- support for metadata that help in retrieving relevant data
- support for generation of multiple views on the same data

The use of the WebDAV protocol reduces the data publishing and accessing to simple file transfer operation familiar to the users.

REFERENCES

1. E. R. Harold, W. S. Means, *XML in a Nutshell, 2nd Edition*. O'Reilly, New York,2002
2. S. Bekman, E. Cholet, *Practical mod_perl* O'Reilly New York,2003
3. E. T. Ray and J. McIntosh, *Perl&XML*, O'Reilly New York, 2003
4. G. Stein WebDAV:Distributed Authoring and Versioning, Adobe technical Seminar Series, May1999.
5. S. Kim, K. Pan, E. Sinderson, *Arhitecture and data model of a WebDAV based Collaborative System,* CTS conference 2004
6. L. Moreau, Y. Zhao, I. Foster, J. Voeckler, M. Wilde, 2004, *XDTM: the XML Dataset Typing and Mapping for Specifying Datasets*. Preprint submitted to European Grid Conference (EGC'05), Amsterdam
7. M. J. Beckerle, *A Proposal for a DFDL that handles Commercial Data Processing Requirements,* draft on http://forge.gridforum.org/,2003