Evaluation of XDMoD

Marco Mambelli, 11/27/2013

This document is a brief evaluation of XSEDE Metrics on Demand[1] as a frontend to display and present information collected via Gratia or via a centralized data warehouse for the Fermilab community.
XSEDE Metrics on Demand is proposed as a comprehensive auditing framework for use by high- performance computing centers, which provides metrics regarding resource utilization, resource performance, and impact on scholarship and research. It has been developed mainly by the CCR at the University of Buffalo and focused on the XSEDE community. It is a role-based framework is designed to meet the following objectives: (1) provide the user community with a tool to manage their allocations and optimize their resource utilization; (2) provide operational staff with the ability to monitor and tune resource performance; (3) provide management with a tool to monitor utilization, user base, and performance of resources; and (4) provide metrics to help measure scientific impact.

There is a XDMoD deployment (https://xdmod.ccr.buffalo.edu/) collecting and displaying accounting and performance information for the XSEDE collaboration. This one collects information about jobs running on all XSEDE resources, including the Open Science Grid that is feeding jobs information via Gratia.
And there is an open source version of the project, Open XDMoD [3], more suited for the monitoring of a cluster or a computing center. It can acquire information only about jobs running on PBS (OpenPBS, Torque/Maui) or SLURM clusters and its code is available on sourceforge.net.
This evaluation refers to Open XDMoD, the only version I could have access to.

User interface

Anonymous users can see a summary page (Summary tab, Fig. 1) with plots and few key numbers and a second page (Usage tab, Fig. 2) with plots documenting resource usage, selectable via a tree menu. All plots allow to customize the time interval both with a calendar and some preset selections (e.g. previous and current month, previous week, ...). The plots in the "Usage" page can also be filtered for data, they provide different display options (e.g. bars, lines, area, table with the data) and they can be exported as images (in various formats) or data (CSV).
Once you login as registered user, then you can access two more tabs: the *Usage Explorer*, Fig. 3, to customize the plots in the summary and the *Report Generator*, Fig. 4, to create collections of plots that can be e-mailed periodically or exported as PDF or Word document.
It is nice to be able to customize the plots in the summary page and XDMoD remembers the settings of the user even if she/he logs in from a different computer.
I was able to perform most functions, like browsing the data, customizing the plots,

saving images and CSV files. I was able to compose and save reports but the reports generation failed, both for Word and PDF files.

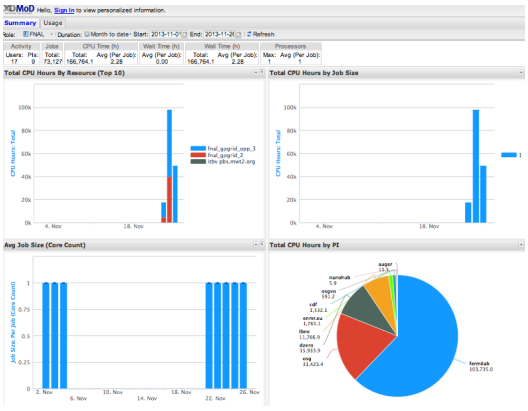

Fig.1 – Summary tab

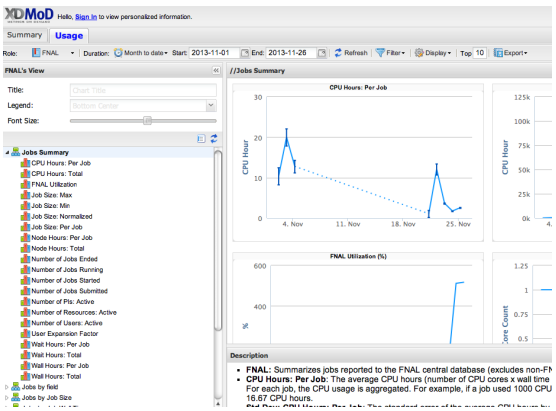

Fig.2 – Usage tab

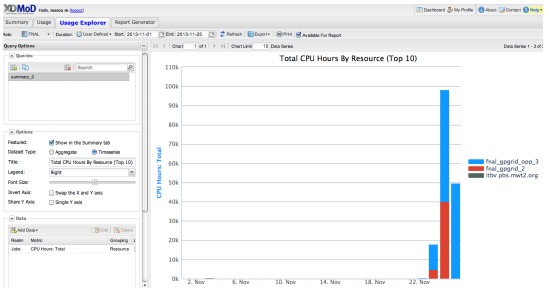

Fig.3 Usage Explorer tab

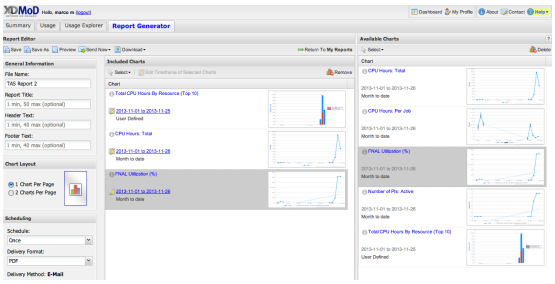

Fig.4 – Report Generator tab

There are some helper utilities for administration tasks like feeding data into XDMoD: *xdmod-setup* to help with the initial setup and add resource information; *xdmod-import-csv* to import some data (e.g. user data) from CSV files; *xdmod-shredder* to import job information from PBS or SLURM log files; *xdmod-ingestor* to populate and optimize the data warehouse.

Code

Most of the code has been developed since 2010 with some older code. Open XDMoD has been officially released as open source project in November 2013. The main developers are Jeffrey T. Palmer and Amin Ghadersohi, under the project lead of Steven M. Gallo and the direction of Tom Furlani at the University of Buffalo's CCR (xdmod-support@ccr.buffalo.edu). I corresponded mainly with Jeff (jtpalmer@ccr.buffalo.edu) and Steve.

Most of the code is in PHP 5, using the Zend framework[4]. The frontend uses dynamic HTML5 pages using the javascript library ExtJS by Sencha[5]. The report generation uses Jasper Reports [6], in Java.

XDMoD uses a Model-View-Controller pattern with many classes used to provide objects around Database elements (tables, groupings, queries, …). There is support for MySQL but should be easy to change to other databases. All the plots are coded, so adding a new plot or different data would require additional coding.

Few classes, *shredders*, take care of importing data from the log files of the PBS (OpenPBS/Torque) or SLURM job managers. And few others, *ingestors*, take care of importing the shredded data into the main data warehouse, checking the consistency and optimizing the data.
The project counts around 120 thousand lines of code (excluding the external libraries Zend, ExtJS and Jasper Reports) as summarized in Table 1.

Language	Files	Blank lines	Comment l.	Code lines
PHP	523	15268	19770	86992
Javascript	158	12746	4030	28657
CSS	28	481	140	2220
SQL	6	49	22	1753
HTML	2	31	2	94
TOTALS	717	28575	23964	119716

Table 1 – XDMoD Lines of Code

There is some documentation about installing, configuring and using the system. It is a total of 25 html files (2420 lines). The installation process was easy following the documentation.
There is limited design documentation [1], mentioning the MVC structure, the data warehouse, the REST API and the portal. There is no documentation of the REST API.

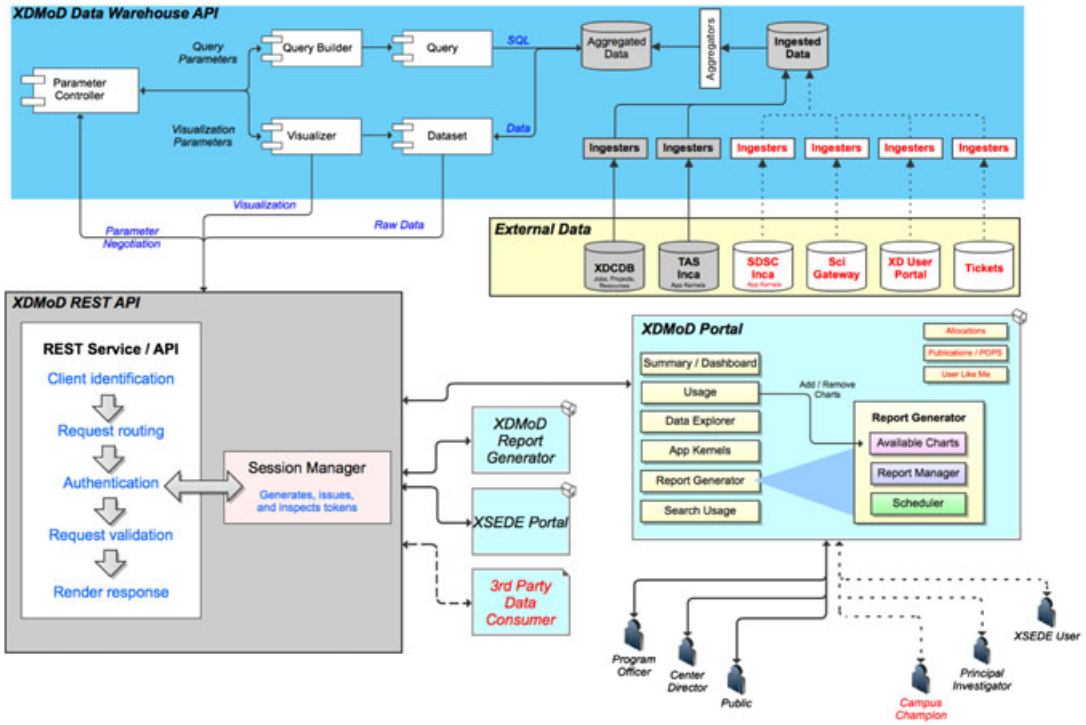

Fig.5 – XDMoD architecture schematic

The project is hosted on Sourceforge.com and the software is released as RPM for RHEL6 and at tar file bundle.

Final considerations

I installed XDMoD via RPM and TAR file bundle. The installation is simple and the instructions cover all the prerequisites. Either way XDMoD must be installed as root, at the system level (at least if you follow the documentation). Developers have been responding to emails and made available a Beta release of the code.
The role-based views are an interesting interface and so is the possibility to customize the views and remember the customizations. The display uses standard technologies that reduce the load on the server and move t to the client (HTML5, JavaScript), it is responsive, the interface is not too cluttered and, as mentioned, offers many customization capabilities.
The system monitors only job metrics (when, where, how and how many jobs ran and which projects they relate to) and is not really flexible. The addition of new metrics and the addition of new queries would both require coding.
As part of the test I found some records from a PBS cluster. The portal is not functional if there is no data. It returns an error and nothing is visible. I wanted to feed additional data extracting it from Gratia. The easier way was to emulate the E records of a PBS log file and I wrote a Python filter to do that. The system is not very robust and feeding data into the database could have been violating some constraint. Connecting a different source to the portal, e.g. queries to the Gratia database would require extensive coding since all the objects are built around elements of a single database.

References

[1] T.R. Furlani 1, M.D. Jones, S.M. Gallo, et al. “Performance metrics and auditing framework using application kernels for high-performance computer systems”, CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE Concurrency Computat.: Pract. Exper. 2013; 25:918–931 Published online 14 June 2012 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/cpe.2871
[2] XSEDE accouting https://xdmod.ccr.buffalo.edu/#main_tab_panel2:tg_summary
[3] Open XDMoD project https://sourceforge.net/projects/xdmod/
[4] Zend Framework http://framework.zend.com/
[5] Sencha ExtJS https://www.sencha.com/products/extjs/
[6] Jasper Reports http://community.jaspersoft.com/project/jasperreports-library