PROCEEDINGS OF SCIENCE

hBrowse - Generic framework for hierarchical data visualisation

Lukasz KOKOSZKIEWICZ*
CERN
E-mail: lukasz.kokoszkiewicz@cern.ch, lukasz@kokoszkiewicz.com

Julia ANDREEVA
CERN
E-mail: julia.andreeva@cern.ch

Ivan Antoniev DZHUNOV
CERN
E-mail: ivan.antoniev.dzhunov@cern.ch

Edward KARAVAKIS
CERN
E-mail: edward.karavakis@cern.ch

Massimo LAMANNA
CERN
E-mail: massimo.lamanna@cern.ch

Jakub MOSCICKI
CERN
E-mail: jakub.moscicki@cern.ch

Laura SARGSYAN
CERN
E-mail: laura.sargsyan@cern.ch

The hBrowse framework is a client-side JavaScript application that can be adjusted and implemented according to each specific community's needs. It utilises the latest web technologies (e.g. jQuery framework, Highcharts plotting library and DataTables jQuery plugin) and capabilities that modern browsers expose to the user. It can be combined with any kind of server as long as it can send JSON formatted data via the HTTP protocol. Each part of this software (dynamic tables overlay, user selection etc.) is in fact a separate plugin which can be used separately from the main application. The Experiment Dashboard framework utilises hBrowse to provide generic job monitoring applications for the ATLAS and CMS Large Handron Collider (LHC) Virtual Organisations (VOs). hBrowse is also used in mini-Dashboard which is part of the EGI Introductory Package and it is used to monitor the status of jobs submitted through the Ganga or Diane submission systems.

EGI Community Forum 2012 / EMI Second Technical Conference
26-30 March, 2012
Munich, Germany

*Speaker.

PoS(EGICF12-EMITC2)062

1. Introduction

hBrowse [1] is an open source framework that can be used by Virtual Organisations (VOs) to create generic job monitoring applications addressing different user needs. It is used by different user communities to monitor jobs submitted through Ganga or Diane job submission systems. Ganga is a frontend for grid job definition and management [2]. Diane extends Ganga adding automatic control and scheduling of computations [3]. The hBrowse framework is also used by the CMS and ATLAS Large Hadron Collider (LHC) VOs, since it is integrated within the Experiment Dashboard monitoring system [4] that is heavily used by these VOs. The hBrowse implementation is a UI client that communicates with the server using AJAX requests and expects JSON responses [fig. 1]. It can be easily configured via a single settings file. The hBrowse framework provides many options to visualise data including dynamic tables and charts.

2. Purpose and scope

The hBrowse framework was developed in order to provide a generic UI that can be used by different grid user communities in various monitoring web-based systems. Many monitoring web applications share the same UI elements such as tables, charts and filters in very similar layouts. The natural choice was to build a generic system that would be flexible enough to satisfy a wide-range of requirements.

3. Architecture

The hBrowse framework uses common modern technologies that makes it easy to modify by the developers who are not involved in the original project. It uses jQuery and many of its plugins. The main jQuery plugin component used by the framework is DataTables, which handles table rendering and interactivity. The hBrowse framework is, in essence, a fully configurable JavaScript client application with the following main architectural highlights:

- Different backend implementations can be used as an information source. Decoupling from the server allows the server implementation to be changed without changing anything on the client side [fig. 1]
- It is database agnostic, thus, it makes no difference if the data back-end is implemented in a specific open-source or proprietary solution
- Client and server communicate using JSON data transfer format [fig. 1]. JSON was selected because of its syntax which is, in most cases, similar to Python and JavaScript
- The format of the JSON server output is not important because it can be easily translated on the client side to the data format expected by the client application

The applications that were developed using hBrowse utilise the following three main action components [fig. 2]:

- Bookmarking URLs

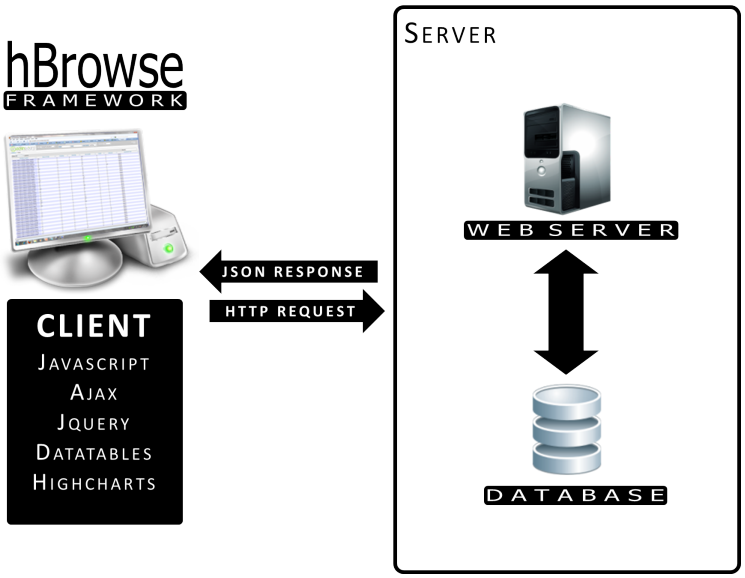

Figure 1: Usual system architecture based on hBrowse framework

- Data model
- View

When the user wants to change the content of a page by clicking on a UI element, the URL will get modified. The URL hash change will be triggered and the internal data model will be changed correspondingly. The appropriate view will be generated [fig. 2] based on the data stored in the model. The application data is stored inside the Data model. Data stored inside the Data model

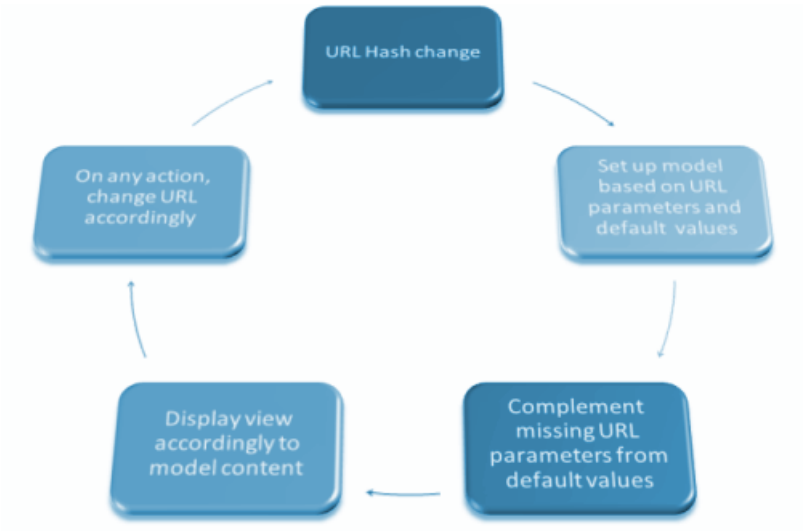

Figure 2: hBrowse working loop

can be divided into two parts, the Operating data and the Cached data. Operating data defines what

should be displayed on the screen as meta-information such as the user name, the defined time intervals relevant to the displayed data and the selected filters. The Cached data depend on the user's request and it can be visualised in various forms, such as different kinds of tables and/or plots, and it can be used as an input to additional requests to the server. An example of such data can be a list of jobs with their attributes, as job CPU and Wall-Clock time, the grid site where the job is running and the current job status.

The application consists of four main modules that are used in the presentation layer [fig. 3]:

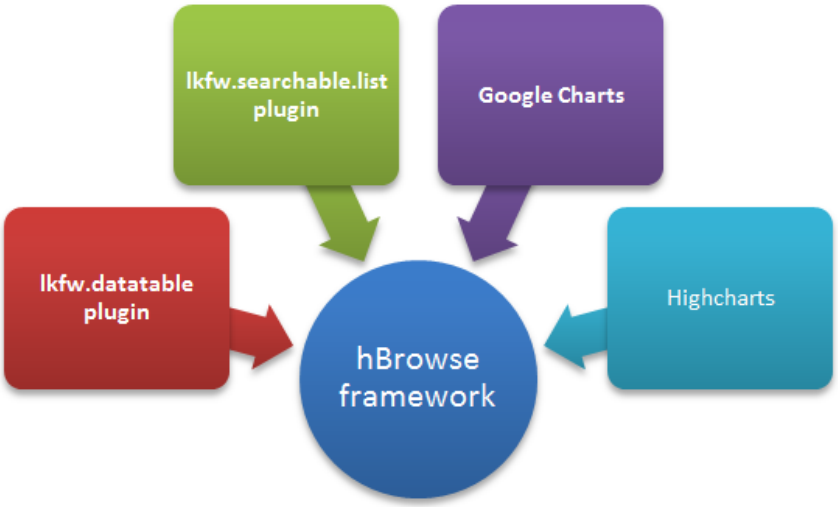

Figure 3: hBrowse main system components

- **lkfw.datatable plugin** - A custom jQuery DataTables plugin overlay that expands its original functionality.
- **lkfw.searchable.list plugin** - A custom jQuery LiveSearch plugin overlay adapted to the needs of the monitoring application.
- **Google Charts** - The framerowk uses the Google Charts service to draw data plots.
- **Highcharts** - A client-side library used by the hBrowse framework to draw charts.

4. Functionality

4.1 Easy to start with

It is very easy to initially set up the application with the hBrowse framework. In addition to required libraries, the package consists of a 'scaffold' application that can serve as a starting point for any web-based UI of a monitoring application. It is filled with many examples of all the essential functionality that the framework provides, so that the framework learning process is quick and straightforward.

4.2 Highly configurable environment

The hBrowse framework was mainly developed to visualise data that have a tree structure (i.e. tasks split in multiple jobs) and due to this, the main UI component is a table view that visualises one data level (for example a list of tasks or list of jobs). The user can access a deeper data level (for example the list of jobs in specific task) by clicking on a selected table row, that will reload the view with a new table, or by expanding the selected row that will present additional information inside the expanded space of the row, typically, in the form of a table or chart. Almost every aspect of the application can be set up inside the settings file including the structure of table rows, information that is shown inside expanded rows, charts and filters. The developer can visualise any number of data levels and create cross references between views. It is also possible to create several separate applications using the same settings file.

4.3 Bookmarking and browser history support

Typically, AJAX-based applications do not work properly with the browser's history. It is not the case in hBrowse. The jQuery.bbq plugin is used in the hBrowse framework in order to ensure that the back and forward buttons work properly in any recent browser.

4.4 Instant search

The modified datatables jQuery plugin that is used by the hBrowse framework allows users to search through the data very quickly. The user can enter a searched pattern into the "search" field and every column of every row will be searched to find a matching row.

4.5 Charts support

To visualise data in the form of charts both Google Charts and Highcharts can be used. Google Charts is an online service that can generate a chart based on the URL parameters and it produces static .jpeg files. As an alternative, the powerful Highcharts JavaScript library was included into a system. It can produce charts on the client side. Highcharts produces high-quality and interactive charts with animations, tooltips and on-click menus.

4.6 Fully configurable data filters

One of the important features of user interfaces is the ability to filter the exposed information. The hBrowse framework supports several filter types which correspond directly to the data type of input fields. The developer can use the following types: text, select, multiselect (select with a possibility to select multiple values), date and datetime. There is also a possibility to set a pre-request filters check in order to determine if the form of the filters was filled in correctly. Filters can be configured directly in the framework settings file.

4.7 Server-client separation

The framework resides on the client side and communicates with the server using HTTP requests. The server should return data in JSON format [fig. 1]. This approach has several advantages. Firstly, it facilitates an effective collaboration of developers with various skills (web design and database background). It also implies a very clear separation of responsibilities where server

and client side developers have to agree only on a communication layer such as on the request parameters and response data structure. Clear server and client side separation allows the client or server side implementation to be changed independently of each other as long as the new component is compatible with the communication layer.

4.8 Data caching

Every server response is stored on the client side for later use. For example, data that mainly serves to display a table can be re-used to draw a chart or a different table underlining a different aspect of the data.

5. Example of applications using the hBrowse framework

5.1 Datasets Distribution

The application was developed to view the dataset replication requests and operational displays for data management activities in the ATLAS VO. It clearly presents the status of every particular request and it shows whether any given data collection was properly replicated at a given site.

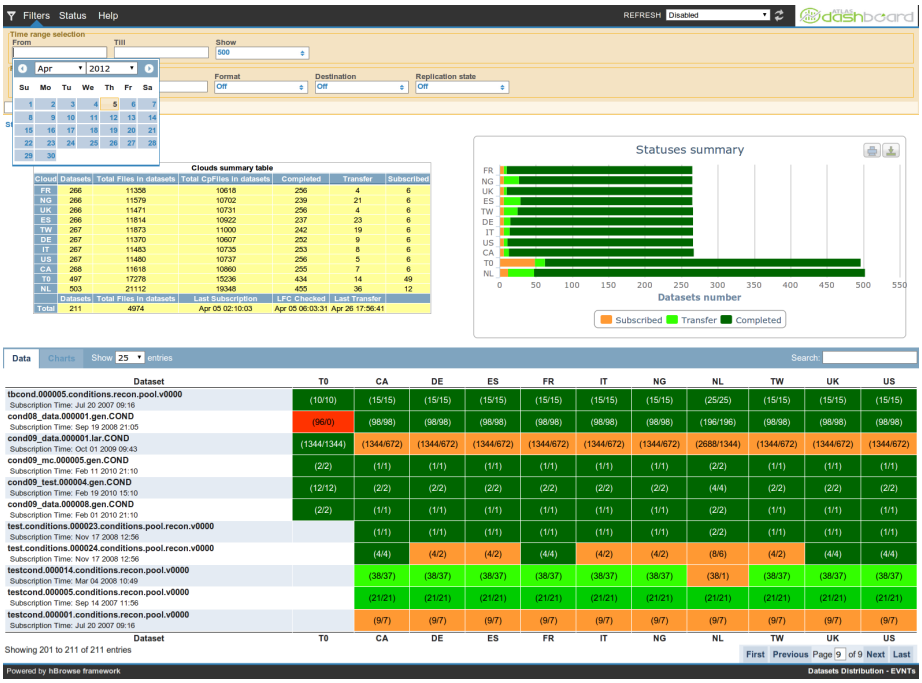

Figure 4: Dataset Distribution main page

5.2 Monitoring Tool for Analysis Users

This application allows physicists to follow the progress of their analysis tasks being processed on the distributed infrastructure [5]. The user can expand any selected task to get a graphical overall summary of execution/completion progress, and to access details of each job. Available plots show the distribution by site of successful, failed, running or pending jobs. They can help identify any

problematic site and exclude it from further resubmission in case of eventual job failure. This application is widely used by individual scientists and by the analysis support teams, who do not always have access to the job processing repositories containing data required for debugging [6].

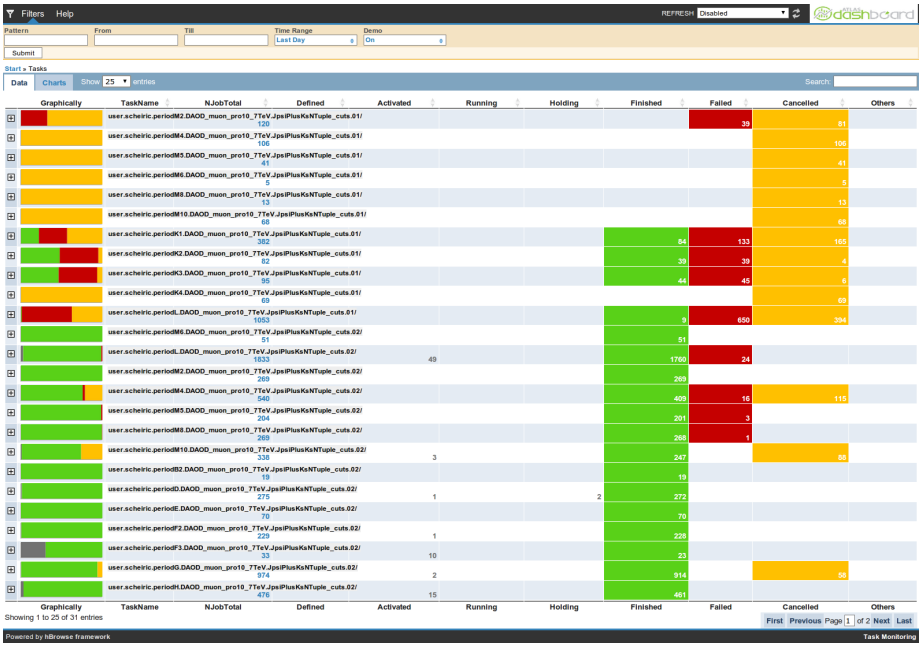

Figure 5: Monitoring tool for analysis users main page

5.3 Production Operators UI

Production teams in the LHC VOs are responsible for massive processing of data coming from the detectors and for simulating large data samples requested by the physics community. This application provides information regarding the evolution of the tasks and the associated jobs in an organised manner, starting with activity overviews and summaries of the different errors to site [fig. 6] and work-flow, with a possibility to expose more detailed information concerning individual tasks and jobs.

5.4 Interactive View

It is a job-centric view aimed at facilitating understanding and debugging of job processing in real-time [7]. The entry point is the number of jobs submitted or accomplished in a chosen time period [fig. 7] presented in a graphical or tabular form and ordered by a selected attribute, for example, by site, computing element, user, application and submission tool. The interactive interface allows the user to drill down to the detailed information of a set of jobs or a particular job. The interface exposes job processing success rate, job CPU and Wall-Clock time and efficiency in the scope of a given VO. Any problem or inefficiency can be easily identified with this application.

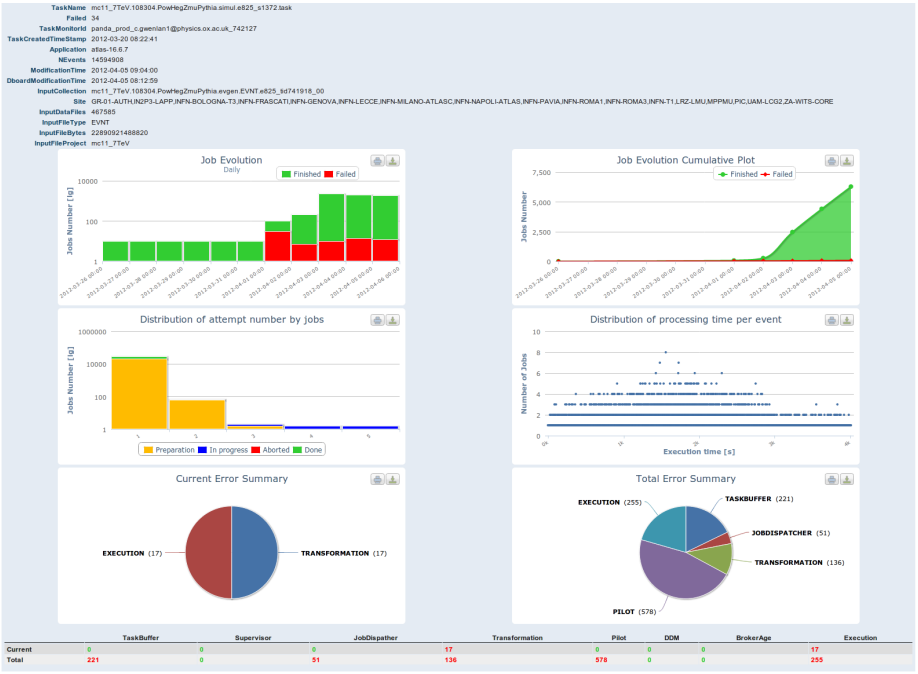

Figure 6: Production Operators UI expandable row content

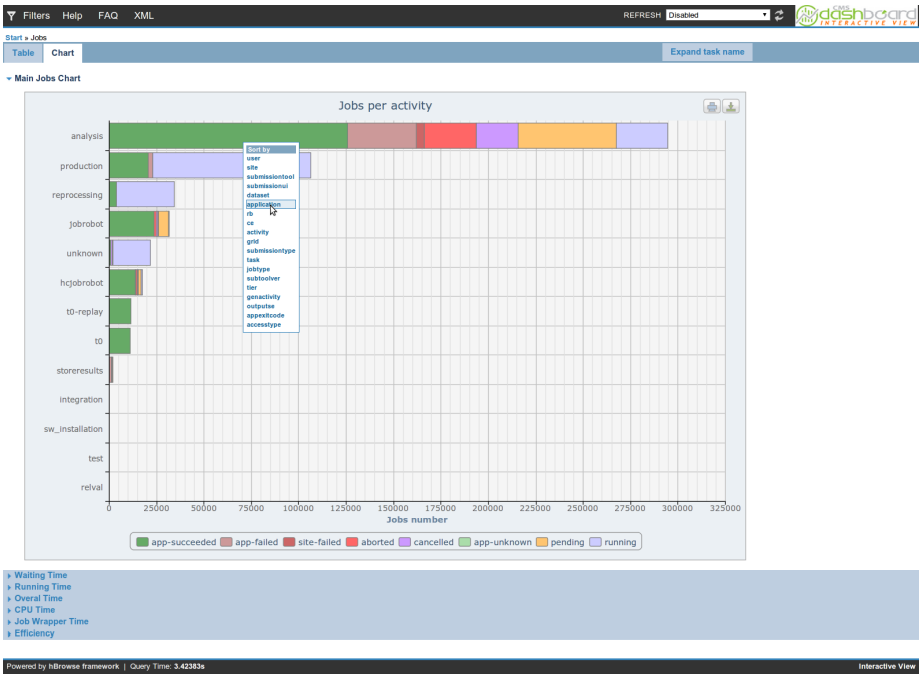

Figure 7: CMS Interactive View main page

6. Conclusions

The hBrowse framework is currently used by communities even outside the scope of the LHC such as the EnviroGRIDS [8], Geant4, VLMED. Although it was designed initially as a generic Ganga/Diane monitoring tool, hBrowse is now effectively utilised by any individual or community that submits tasks through the Ganga/Diane job submission systems. Task monitoring capabilities are exposed either by experiment-specific interfaces, or through a generic web-based front-end [9]. The hBrowse framework is also used as the user interface of several ATLAS and CMS specific applications such as:

- the Datasets Distribution which is used by ATLAS
- the Monitoring Tool for Analysis Users which is used both in ATLAS and CMS
- the Production Operators UI which is currently used in ATLAS
- the Interactive View which is currently used both in ATLAS and CMS

The framework was easily adapted for each particular use case.

The hBrowse framework proved to be a flexible tool, allowing rapid deployment of services for different user communities. Due to it's separation from the server (i.e. data handling) component it is very easy to develop new applications or upgrade existing ones to a modern user interface allowing users to visualise their data more effectively. The hBrowse framework proved to be an excellent solution for quickly implementing user interfaces visualising any sort of hierarchical data by combining the latest web2.0 technologies, JavaScript frameworks and plug-ins.

References

[1] Online, *hBrowse*, 2012, hbrowse.net

[2] J.T. Moscicki et al, *Ganga: a tool for computational-task management and easy access to Grid resources*, 2009, Computer Physics Communications, arXiv:0902.2685v2

[3] J.T. Moscicki et al, *DIANE - Distributed Analysis Environment for GRID-enabled Simulation and Analysis of Physics Data*, 2003, NSS IEEE 2004, Portland, Oregon, USA, DIANE-NSS2003

[4] J. Andreeva et al, *Experiment Dashboard for monitoring computing activities of the LHC virtual organizations*, 2010, J. Grid Comput. **8** 323-339 doi:10.1007/s10723-010-9148-x

[5] E. Karavakis et al, *CMS Dashboard Task Monitoring: A user-centric monitoring view*, 2010, J. Phys.: Conf. Ser. **219** 072038 doi:10.1088/1742-6596/219/7/072038

[6] E. Karavakis et al, *User-centric monitoring of the analysis and production activities within the ATLAS and CMS Virtual Organisations using the Experiment Dashboard system*, in proceedings of EGI Community Forum 2012 / EMI Second Technical Conference, 2012, PoS(EGICF12-EMITC2)110

[7] J. Andreeva et al, *Experiment Dashboard for monitoring of the LHC distributed computing systems*, 2011, J. Phys.: Conf. Ser. **331** 072001 doi:10.1088/1742-6596/331/7/072001

[8] L. Kokoszkiewicz et al, *SWAT Gridification*, in proceedings of EnviroGRIDS project

[9] Online, *hBrowse*, 2012, gangamon.cern.ch