Scientific Visualization, 2018, volume 10, number 1, pages 69 - 76, DOI: 10.26583/sv.10.1.05

Data visualization and representation in ATLAS BigPanDA monitoring

S.Padolski[A,1], T.Korchuganova[B,2], T.Wenaus[A,3], M.Grigorieva[B,C,4], A.Alexeev[B,5], M.Titov[C,6], A.Klimentov[A,7]

[A] Brookhaven National Laboratory, Upton, NY, USA
[B] Tomsk Polytechnic University, Tomsk, Russia
[C] National Research Center "Kurchatov Institute", Moscow, Russia
[1] ORCID: 0000-0002-6795-7670, spadolski@bnl.gov
[2] ORCID: 0000-0001-5792-8182, tatiana.korchuganova@cern.ch
[3] ORCID: 0000-0002-8678-893X, wenaus@gmail.com
[4] ORCID: 0000-0002-8851-2187, magsend@gmail.com
[5] ORCID: 0000-0001-7025-432X, frt@tpu.ru
[6] ORCID: 0000-0003-2357-7382, mikhail.titov@cern.ch
[7] ORCID: 0000-0003-2748-4829, Alexei.Klimentov@cern.ch

Abstract

A rising demand on visualization of high-volume data we observe currently is an immediate follow up to the technological ability to collect, store and handle Big Data drastically increased in the recent years. A lot of contemporary development activities both open source and proprietary are focused on indexing, plotting and navigating over historical and scattered data. In particular GitHub reports 7,449 data visualization projects registered in the 2017 while only 3,183 for 2015. Many of such developments form a technological stack, which delivers advanced data visualization functionality almost out of the box, but in the same time complex data aggregation, analysis and navigation still requires custom solutions.

In this paper we describe author's experience of building BigPanDA monitoring which provides interface to the PanDA jobs management system used in CERN Particle Physics experiments such as ATLAS at LHC and COMPASS at SPS. We present principles of design of user interfaces and data visualization which provides compact, user oriented access to the data stored in hundreds millions of rows scattered in tenths of database tables and another sources. The system operates in 24x7 mode and serves different needs of more than thousand users.

Keywords: monitoring, CERN, ATLAS, visualization, interactive visualization.

Introduction

As of middle 2017 the ATLAS Production System [1] daily executes 2M computation jobs distributed over 170 WLCG sites [2] and opportunistically used resources, such as supercomputers, academic and commercial clouds and university clusters. This number is achieved by simultaneous running on more than 350k cores. Each job registered in the system is a complex entity described by input/output data location, required software version, hardware specification, a human or system who submits it and in total has 117 attributes. Moreover they are grouped into bunches which are called tasks and campaigns and deliver bigger processing results containing millions of events.

The system internal data should be processed, analyzed and represented in a compact form which could be easily understood by several groups of users: physicists, production managers, site sysadmins, shifters and software developers. These interfaces should include many views allowing to perform operational work such as: checking status of a particular job, representing the distribution of the whole ATLAS grid payload over different physics tasks or downloading a job error log.

Visualization principles

The BigPanDA monitoring visualization pipeline has three different presentation forms: interactive interfaces, tables and plots. All displayed data is fetched from a Django [3] based scalable data processing engine. The primary initial source of the data stream is the Oracle DB [4] which is used by the PanDA workload management system as the storage backend. The following constraints determined the development choice for the custom data processing engine: relational structure of the raw data with complex aggregation of objects scattered among many tables, variety of data to be presented at every interface view, and diversity of navigation paths to be supported by the system. The architectural details including description of the different steps of data processing are described here [5].

Interactive interfaces

Most of the BigPanDA pages are designed to present objects within a hierarchy, summaries for aggregations of objects and vertical navigation in both directions. For example a jobs page [6] which is the most frequently demanded view of the monitor is presented in Figure 1. There are three following tables:

- **Job attribute summary** contains aggregates of jobs over 26 different criteria. In each cell we show a count of occurences of different values and a link to another jobs subset adding a new selection criteria. E.g. a link with a particular task identifier will lead to the jobs list belonging to a corresponding task. This approach solves two problems: delivers quick representation about structure of properties values distribution and provides navigation to a narrower jobs selection.
- **Overall error summary** provides description of errors, their popularity and corresponding codes occuring in the selected jobs.
- **Job list** provides a short description of selected jobs, one by one.

This pattern, tables with aggregates of selected objects and their description, is used on many summary views and well recognized by the user community.

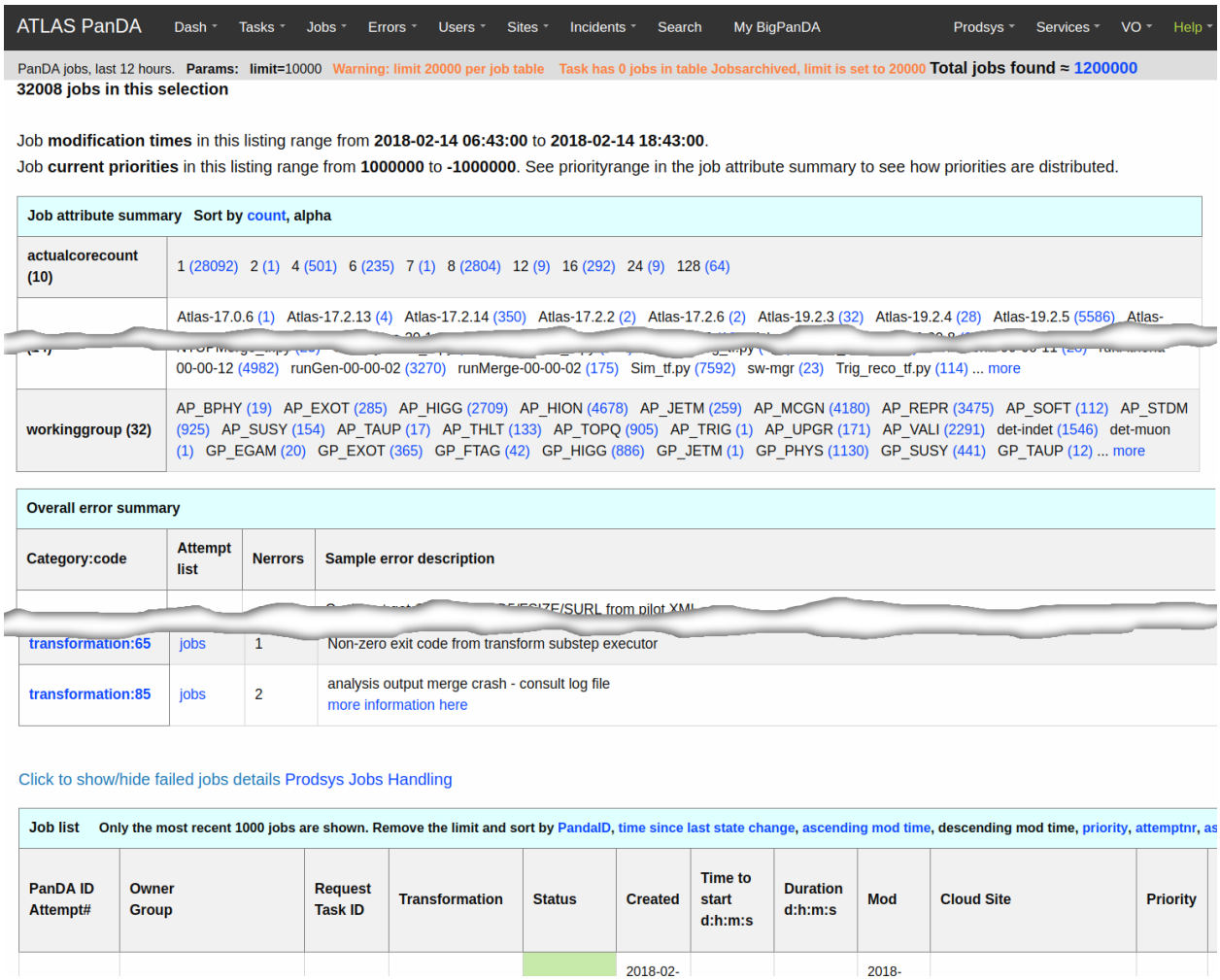


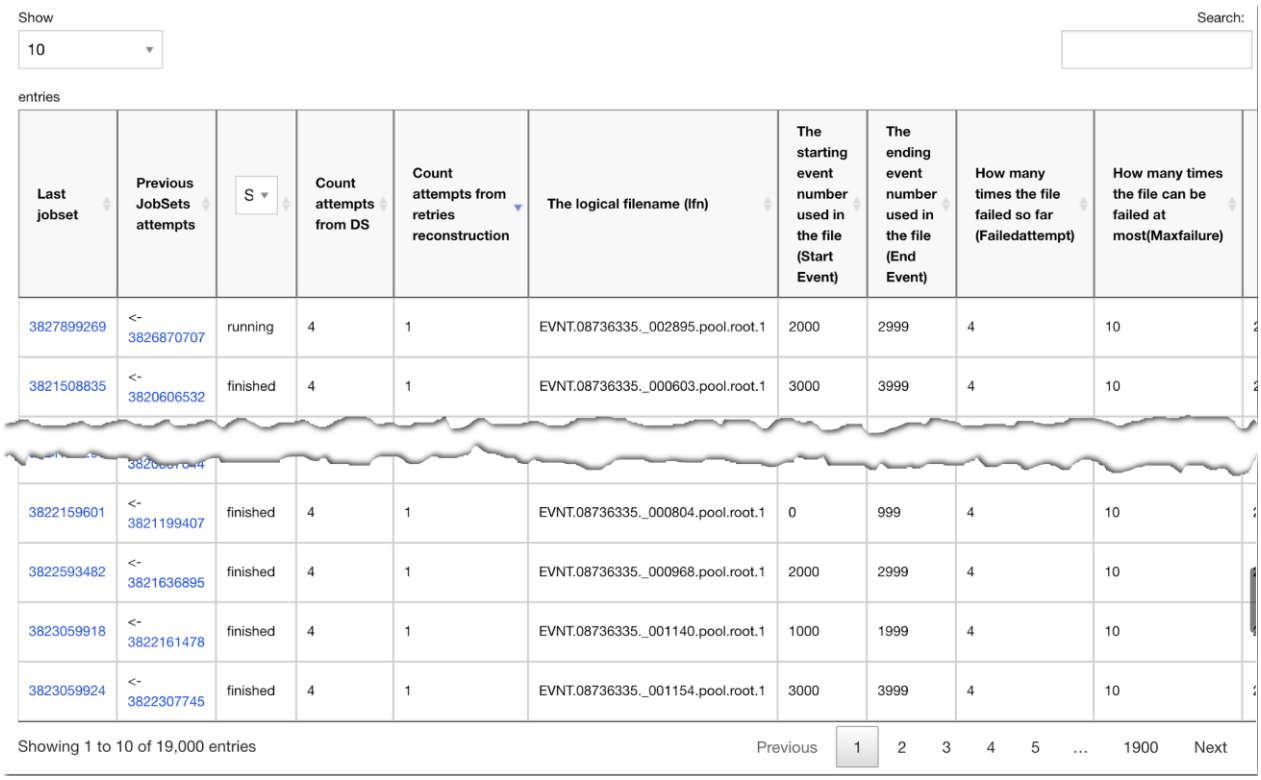
Figure 1. The jobs view

To avoid overloading of views some data is shown only on demand. As one can see in the example the failed jobs details are not shown by default.

Tables

Almost every page of BigPanDA monitoring contains one or a few tables. Most of them are filled with data provided by the processing engine and are rendered using the Django template system. But for some cases this approach does not meet requirements and we utilize a third party JavaScript component - Datatables [7].

This approach decouples table content from the rest of the information presented on a view. In this way information could be prepared and transferred to a client asynchronously. The most valuable advantage of using Datatables in BigPanDA monitoring is a client side control over representation. For example once delivered to the client a set of rows could be sorted in different orders, filtered and split into many tabs. An example of such a table is given on Figure 2.



Show 10 entries Search:

Last jobset	Previous JobSets attempts	S	Count attempts from DS	Count attempts from retries reconstruction	The logical filename (lfn)	The starting event number used in the file (Start Event)	The ending event number used in the file (End Event)	How many times the file failed so far (Failedattempt)	How many times the file can be failed at most(Maxfailure)
3827899269	<- 3826870707	running	4	1	EVNT.08736335._002895.pool.root.1	2000	2999	4	10
3821508835	<- 3820606532	finished	4	1	EVNT.08736335._000603.pool.root.1	3000	3999	4	10
3822159601	<- 3821199407	finished	4	1	EVNT.08736335._000804.pool.root.1	0	999	4	10
3822593482	<- 3821636895	finished	4	1	EVNT.08736335._000968.pool.root.1	2000	2999	4	10
3823059918	<- 3822161478	finished	4	1	EVNT.08736335._001140.pool.root.1	1000	1999	4	10
3823059924	<- 3822307745	finished	4	1	EVNT.08736335._001154.pool.root.1	3000	3999	4	10

Showing 1 to 10 of 19,000 entries Previous 1 2 3 4 5 ... 1900 Next

Figure 2. Example of table visualized by Datatable library

Figure 2 shows additional controls which provide for ordering by every column, selection of jobsets status, search over the table, selection of appropriate number of rows in each table screen and navigation over rows.

Plots

The basic plots functionality provided by BigPanDA monitoring is implemented using the JavaScript D3 library [8]. One of the advantages of using this framework is supporting the different kind of plots united by the same style of representation with additional capability to react on user actions. For example on Figure 3, the number of allocated slots for running Monte-Carlo simulation tasks is shown. When a user selects a section on the pie chart the notation in the center dynamically changes and shows advanced information about the selection.

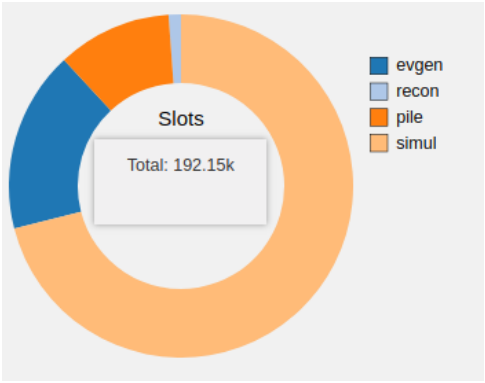


Figure 3. Distribution of number of allocated slots for running production tasks plotted with D3 library

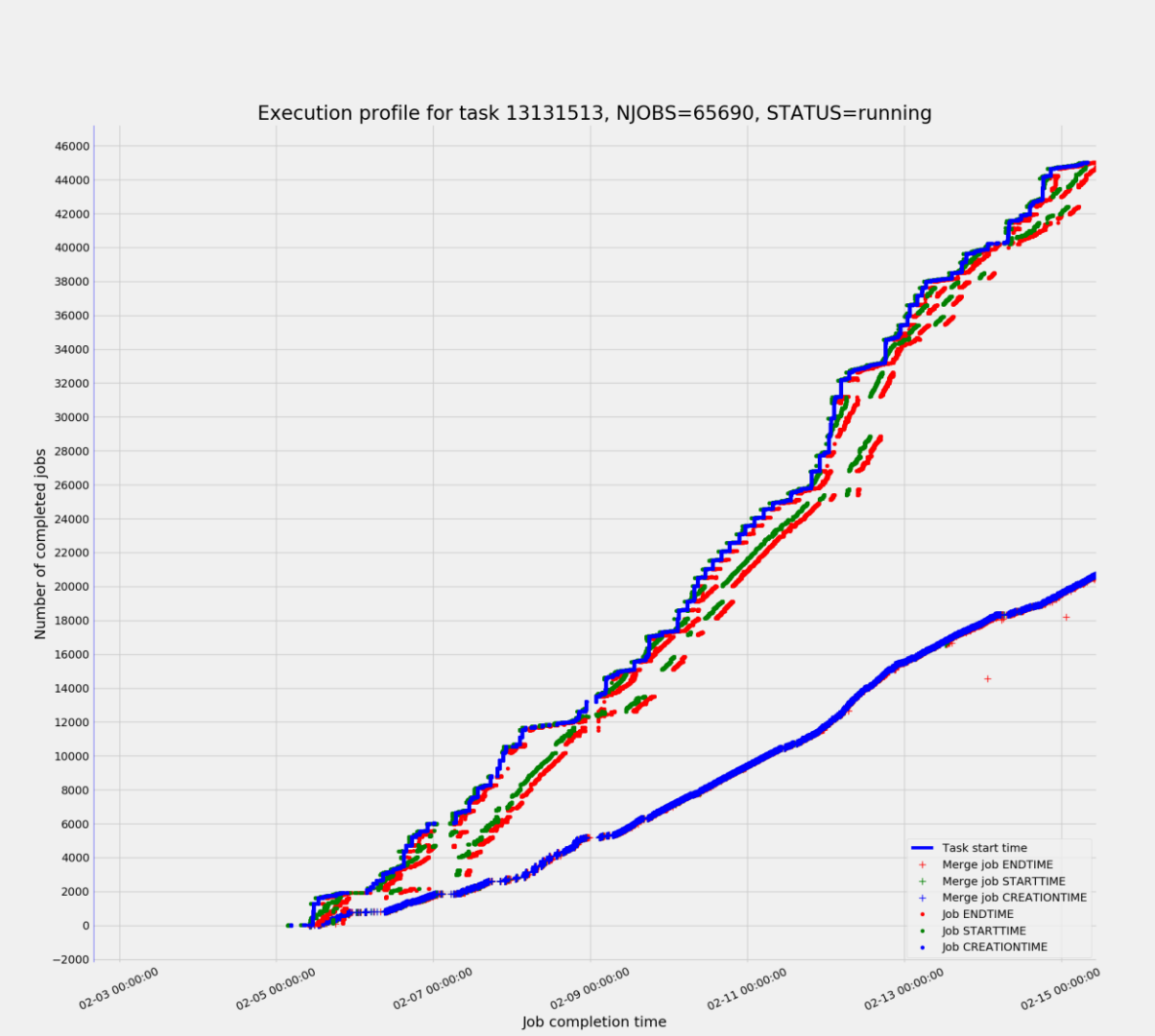


Figure 4. Task profile plotted with matplotlib library

For some accounting tasks such as event processing progress the data sample for a plot could be as high as tens of millions of entries. Making these plots in a web browser requires significant bandwidth of a client for data transfer coupled with CPU overhead for rendering. To avoid this, the high statistics plots are built on the server side using Python's matplotlib library. Once the plot is rendered inside the server process it is exported into the png format and delivered to a client as a usual raster picture (Figure 4).

Forecasted data representation (machine learning based)

One of the distinctive features of the next generation of monitoring is the possibility to integrate methods and techniques for data analysis. BigPanDA monitoring provides an experimental feature based on machine learning algorithms - prediction of the task execution duration and thus, task time-to-complete (TTC) estimation [9]. Representation of it with the connection to the real task execution process is shown on Figure 5.

Task ID:	13049347
Status:	done
Creation time:	2018-01-23 15:16:57
Start time:	2018-01-23 16:28:43
Predicted time to complete:	2018-01-27 13:21:13
Real end time:	2018-01-25 18:25:24

Show/hide task progress plot

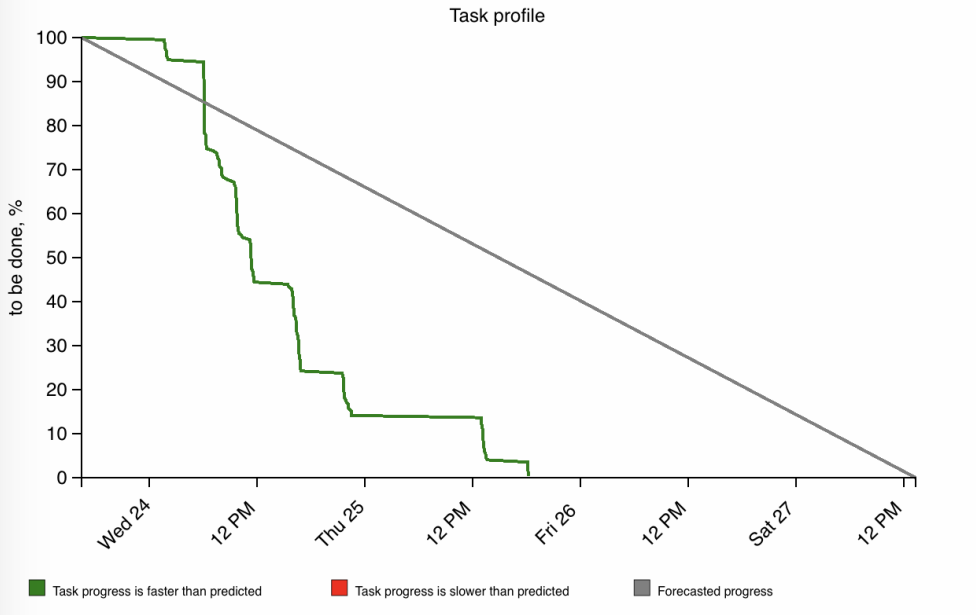


Figure 5. Representation of task Time-To-Complete in comparison with task real progress

Conclusion

BigPanDA monitoring provides various ways of information visualization including web pages, plots and tables. These forms were designed to represent information constructed by processing tens of millions of rows of internal PanDA data. In this paper we described adopted approaches and some use cases driving the implementation. The outcomes of the work could be reused in applications dealing with Big Data representation for different domains.

References

[1] M. Borodin et al., 2015 Scaling up ATLAS production system for the LHC run 2 and beyond: project ProdSys2, https://doi.org/10.1088/1742-6596/664/6/062005 J. Phys. Conf. Ser. 664 062005

[2] Bird I 2011 Computing for the Large Hadron Collider Annual Review of Nuclear and Particle Science vol 61 pp 99-118

[3] Django project, “Django” [software], version 1.11.5, 2017. Available from https://docs.djangoproject.com/en/1.11/ [accessed 2018-02-20]

[4] Oracle DB project, “Oracle” [software], version 11g, 2017. Available from http://www.oracle.com/technetwork/database/database-technologies/express-edition/downloads/index.html [accessed 2018-02-20]

[5] Wenaus T., Padolski S., Korchuganova T., Klimentov A. (2018, August). ATLAS BigPanDA Monitoring. Poster session presented at the 18th International Workshop on Advanced Computing and Analysis Techniques in Physics Research, Seattle, USA.

[6] BigPanDA Monitoring, “Jobs” view. Available from https://bigpanda.cern.ch/jobs [accessed 2018-02-20]

[7] DataTables project, “DataTables” [software], version 1.10.16, 2017. Available from https://datatables.net/download [accessed 2018-02-20]

[8] D3js project, “D3js” [software], version 3.5.17, 2016. Available from https://d3js.org/ [accessed 2018-02-20]

[9] M.Titov, M.Gubin, A.Klimentov, F.Barreiro, M.Borodin, D.Golubkov, "Predictive analytics as an essential mechanism for situational awareness at the ATLAS Production System", The 26th International Symposium on Nuclear Electronics and Computing (NEC), CEUR Workshop Proceedings, vol. 2023, pp.61--67, 2017.