Monitoring data visualization for ATLAS TDAQ transient storage

Georgios Kamaras (University of Athens) for the ATLAS Collaboration

CERN, 23 August 2018

Introduction

The transient storage of the *ATLAS*[1] data acquisition system records physics data at several GB/s. Its performance and reliability have a direct impact on the performance of the whole data acquisition system of ATLAS. Monitoring its performance is a key operational task, so a monitoring web application exists for this purpose. I was in charge of integrating interactive plots to this application. I also aimed to improve the user experience, and participate in the evolution of the application's features, robustness and performance. Through this work, many valuable lessons were learned, which I will also outline in the present document.

SFO Run Report

The transient storage software is an in-house distributed application producing a lot of raw monitoring data. These data are published to the ATLAS' full-featured monitoring system and recorded for offline analysis. *SFO*[2] *Run Report* is a dedicated monitoring web application written with a Python *back-end*[3] and a JavaScript and jQuery *front-end*[4]. It exploits the recorded monitoring data to produce complex and meaningful insights on the operations of the system, like operational summaries, aggregated data, correlations, etc. Presenting the data in an appealing, efficient and practical way is an important task of this web application.

As of July 2018, SFO Run Report was operational, but had a lot of room for improvement, since certain limitations lowered its efficiency and the depth of the insights that could be derived by its utilization. SFO Run Report was originally designed to generate pictures of plots on its back-end, store them locally into the server and then publish them to its front-end, meaning to the user's browser. As a result, the application's interface lacked capabilities of vital importance for a monitoring application, such as the ability to interact with the plots and the inclusion of important annotations regarding the operational states of the SFO.

1 A Toroidal LHC ApparatuS
2 Sub-Farm Output
3 The part of a web application that works on the server's side, not accessible for the user
4 The part of a web application the works on the user's browser side, accessible for the user

Ideas for improvement

My task was to improve the SFO Run Report application's interface, by making sure that the aforementioned limitations would stop existing. To achieve that, I had to gain an in-depth understanding about the operation of the application as well as the ATLAS experiment components with which the application interacts, like the *P-BEAST*'s API[5]. P-BEAST is "a highly scalable, highly available and durable system for archiving monitoring information of the trigger and data acquisition (*TDAQ*) system of the ATLAS experiment at CERN"[6].

The most important part of upgrading the SFO Run Report application was expected to be integrating an advanced plotting library on its interface. This needed to be a library that guarantees a high degree of interactivity, is open-source so it can be constantly customized to our needs, if necessary, and has the potential of being regularly upgraded by its original author, minimizing our group's concerns over its maintenance. Through this library, features like zooming-in/out, panning, re-sizing, selecting a specific curve, resetting, exporting as a picture needed to be enabled. Furthermore, annotations about useful statuses regarding the operation of the SFO sub-system of ATLAS was necessary be included to our plots. This inclusion is expected to advance the user's conscience and understanding concerning what is actually depicted in a plot and in which stage of the ATLAS operation it took place.

Project progress

Flot (or *Flot Charts*) was chosen as the front-end library to create the interactive plots. Flot is a JavaScript plotting library for jQuery. It focuses on being simple to use and generating attractive and highly interactive plots. Flot was chosen because it fulfills all the requirements that had been set and also has a strong community of developers supporting the project's existence by sharing their knowledge, creating useful plugins and guaranteeing its constant evolution and existence as open-source software.

In parallel to searching for the appropriate JavaScript library, I also became familiar with the application's source code and its operation, while making minor modifications to the existing interface's structure. One such modification was rearranging the plots' windows formation on their page, so that the user would be able to easily see them one at a time, while interacting with each plot independently, according to his needs. I also made small additions to the interface's capabilities, enhancing and optimizing the existing ability to toggle many items at once and delete all items at once in every page of the application.

5 Application Programming Interface, a set of subroutine definitions, communication protocols, and tools for building software.

6 A Persistent Back-End for the ATLAS TDAQ On-line Information Service (P-BEAST), Sicoe et al. 2011, 14th International Workshop On Advanced Computing And Analysis Techniques In Physics Research, Uxbridge, West London, UK, 5 - 9 Sep 2011

For the required level of interactivity to be achieved it was necessary for the plot-generation logic to be transferred from the *server-side* (back-end) to the *browser-side* (front-end). The existing Python plot-generation infrastructure needed to be replicated into JavaScript and the data needed to be transferred from the back-end to the front-end in a form that makes their processing as computationally efficient as possible, while not overloading the user's browser with unnecessary processing of the data. This process required from me to understand the underlying data structures and to distinguish which part of the data processing can remain on the server-side and which is necessary to be transferred to the browser-side. I also had to devise ways to debug the application during the update, because of problems that were naturally present while modifying certain code modules. The debugging process evolved around printing debugging messages tailored to each plot on the application's interface.

Illustration 1: A sample of a plot after the interface's update

Integrating Flot involved taking advantage of Flot's features to add the desired level of interactivity with the generated plots. Using Flot, I was able to plot many different lines in the same plot, make the plots re-sizable, capable of zooming-in and out in both axes or just in the X or Y axis. I also found ways to annotate the various operational statuses in meaningful ways and I developed a method for exporting the plots as insightful pictures, even after the user has interacted with them. For these two features I had to experiment with various plugins from the Flot community, however since none of them lead to a satisfactory result based on the standards that I had set, I implemented these two features in a simple way on my own, using the standard Flot API.

Throughout the process of upgrading the SFO Run Report interface there was one clear goal in my mind: making the application more meaningful and interactive than ever before, while

keeping a clean, highly intuitive and streamlined web interface. With this goal in mind certain design decisions were taken, like removing the legend of the plots and integrating its information in the line selection panel or disabling the toggling of processing-heavy plots, specifically the ones depicting the Stream Writing Rate and Event Size as a function of time. I also wanted to contribute to the improvement of the application's documentation, in order to eliminate unnecessary workload for anyone who will try to upgrade the application in the future. To this end, I commented the application's code wherever I found it necessary and I transferred the existing documentation (README) into an easily readable Markdown (.md) file.

Lessons learned & Ideas for the future

While working on upgrading the SFO Run Report web application's interface I obtained many useful lessons, I arrived into certain conclusions and I came up with a number of ideas for further updating the application in the future, as a continuation of the work that I did. In this section I will try to outline all these things with the hope that they will be proven useful to CERN, to my group and to whoever may be assigned to continue my work.

Perhaps the most important lesson that I learned while working on my project, is that one cannot design a successful monitoring application without implementing and testing each one of his ideas. Things that may appear useful as a concept may contribute nothing to the overall application's functionality. Also, for a monitoring application designed for *experts*, grouping the information and the features is not a priority. Since each plot is presenting a different set of information, the user should be able to interact with it independently of his interaction with any other plot on the same page. Furthermore, a monitoring application should not overwhelm its user with information in an attempt to be more insightful. Since the user is an expert, the application should strive to be meaningful, offering him only the information that he is going to need to perform his task in a clean and intuitive way. In this direction, it is often important for the developer to search for ways to combine different interface components into a single one component, that is going to condense the initial components meaning. Such an example is the current look of the line selection panel that accompanies each plot, which serves both as a legend and a check-box and was created with the goal to keep the interface as clean as possible.

Two features of the application that I believe have a lot of room for future improvement are the application's main page and the pictures exportation module. A problem that I constantly found in my way was opening tabs of monitored runs where the LHC beam was inactive, which had no meaningful information to give me when looking for some monitoring during operation. There is currently no way of distinguishing if a monitored run corresponds to a time interval when the LHC had a beam or to a time interval with no beam. The only way for the user to know is to actually select the run's tab from the application's main page and wait for the results. It would be great if at the time of loading the application's main page there was a way for the user to know whether there was a beam or not in the LHC during a certain monitored run. From a design

perspective this is easily achievable with a simple annotation, like another small badge on the run's tab. When it comes to the pictures exportation module, there is room for a lot of improvement into the generated pictures clarity and resolution, and a discrete but informative legend has to be added to the respective canvas prior to the picture's exportation.

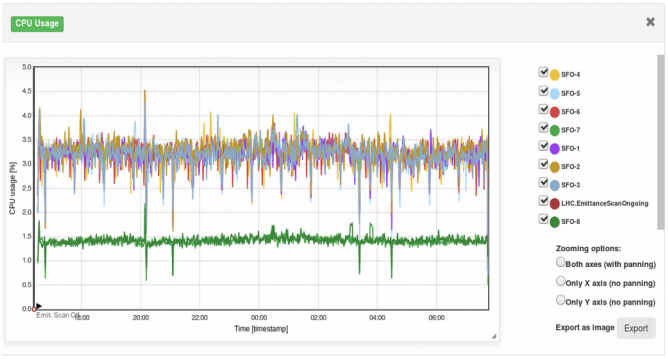

Illustration 2: CPU Usage plot tab

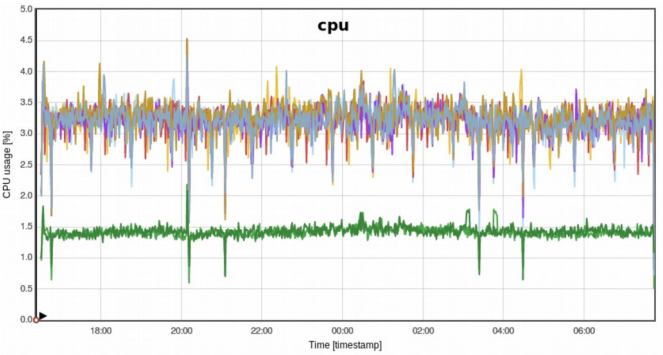

Illustration 3: CPU Usage plot as exported image

Conclusion

The SFO Run Report web application provides a great example of how cutting-edge data visualization tools and techniques can be employed to upgrade a monitoring application for an advanced Physics experiment sub-system. By simply updating an interface, in terms of usability, interactivity and range of the presented information the smooth operation and the overall success of monitoring such a sub-system can be ensured.

In a more personal note, working as a Data Visualization Developer in the SFO Run Report application gave me an amazing chance to see what "data visualization" is about. It also challenged me to discover a way to apply my web development skills and my Human-Computer Interaction knowledge in an area that was largely new to me. The overall experience definitely helped me take my Computer Science, Engineering and even Physics knowledge and skills into the next level.

Acknowledgments: I would like to thank my supervisor Fabrice Le Goff for the excellent cooperation that we had during my eight week presence at CERN, for giving me a lot of freedom and responsibility over my work, for sharing his knowledge and expertise and for providing me with valuable feedback whenever I needed it. I would also like to thank all the members of my group for their cooperation, support and for doing everything they could to ensure that I will get the most our of my presence at CERN, since the moment I arrived.