USAGE OF TIME SERIES DATABASES IN THE GRAFANA PLATFORM FOR THE NETIS SERVICE

E.I. Alexandrov[1,a], M. E. Pozo Astigarraga[2], G. Avolio[2]
on behalf of the ATLAS Software and Computing Activity

[1] *Joint Institute for Nuclear Research, Joliot-Curie 6, RU-141980 Dubna, Russia*

[2] *European Organization for Nuclear Research, CERN*

E-mail: [a] aleksand@jinr.ru

NetIs is a service used to monitor the Data Acquisition network of the ATLAS experiment. The first version was developed at CERN in 2010. Over the years, the need to replace NetIs with an improved service emerged. Indeed, the effort to maintain NetIs has considerably increased together with the size and complexity of the network system; additionally, the Round Robin Database used to store the data results in a loss of granularity over time that makes the tool unsuitable for retrieving accurate values from the past. The graphs produced by NetIs are generated by the backend server and they are quite static, though the GUI is familiar to many users. The main idea was to exploit the recent advancements in time series databases and visualization tools like Grafana in order to present data to the users in a more dynamic way. The Persistent Back-End for the ATLAS Information System, developed in ATLAS for permanent storage of operational data, was already integrated with Grafana and successfully collecting network monitoring statistics. Grafana, despite being a very popular visualization web application, does not support some GUI elements that are used in NetIs such as a tree or position of drop-down. Javascript code integrated with Grafana was used to overcome these limitations.

Keywords: ATLAS, network, monitoring, Grafana

Evgeny Alexandrov, Mikel Eukeni Pozo Astigarraga, Giuseppe Avolio

1. First version of the ATLAS network monitoring tool and motivation for upgrade

The ATLAS experiment is one of four LHC accelerator experiments at CERN [1]. The computing and network infrastructure of the experiment consists of 4,020 computing nodes, 285 network switches and 14,778 switch interfaces (ports). The NetIs system was designed to monitor this network [2]. The first version of NetIs was developed in 2010 based on the cyclic time series Round Robin Database (RRD) [3]. Over the years, the need to replace NetIs with an improved service emerged. Indeed, the effort to maintain NetIs has considerably increased together with the size and complexity of the network system; additionally, the RRD used to store the data results in a loss of granularity over time that makes the tool unsuitable for retrieving accurate values from the past. Another inconvenience of using the first version is that graphs displayed by NetIs are generated by the server backend and they are quite static. This interface has a dynamic tree for selecting nodes, switches or interfaces and information panels as depicted in the image (Figure 1). A new Grafana-based monitoring version using the Persistent Back-End for the ATLAS information system (P-BEAST) [4] database has been developed in recent years.

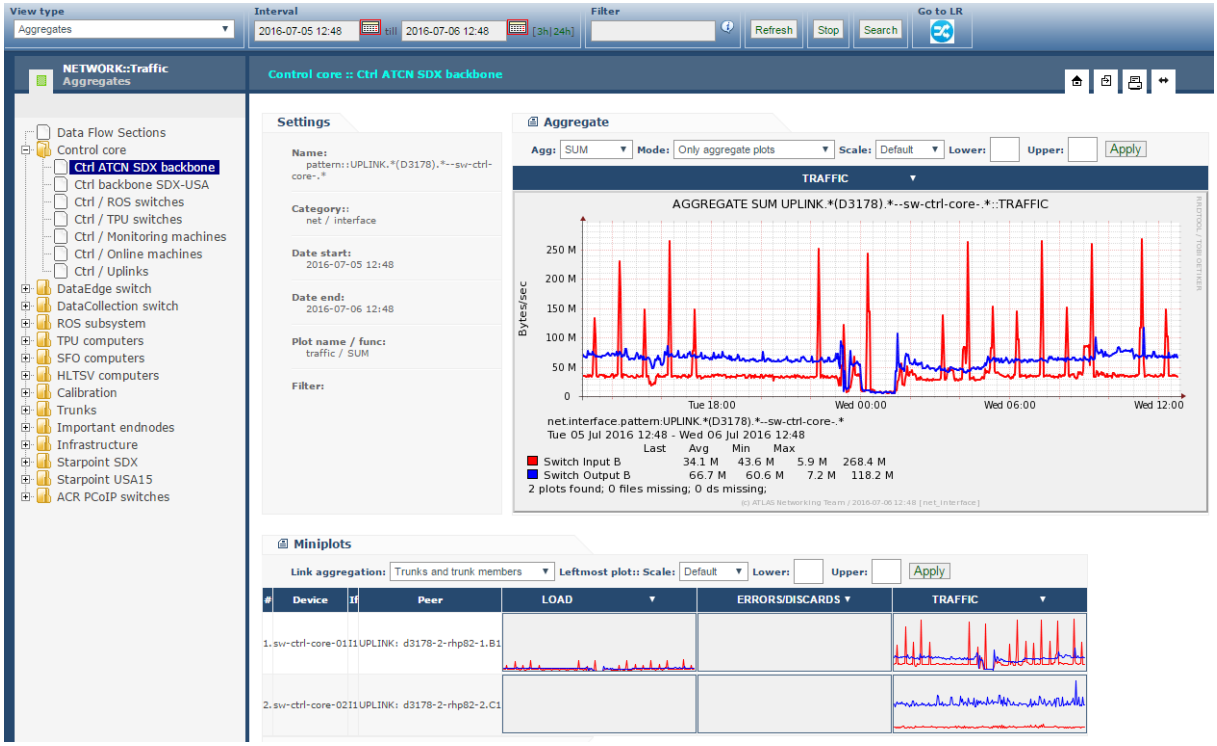

Figure 1. A screenshot of the legacy NetIs [2]

2. The common structure of the NetIs monitoring

The common structure of the legacy NetIs monitoring had two data sources: the first for creating dashboards and the second for obtaining the network topology (Figure 2). In this version, the monitoring data was retrieved from switches using the Simple Network Management Protocol (SNMP) and was stored in a MySQL database to be later converted into RRD files by a separate service. The Matplotlib [5] library generated images from the RRD data sources and the Django framework [6] passed these images to the client. The second data source was used for navigation and it was implemented with a JavaScript tree object from the dhtmlxTree library [7]. The data source of the tree was the Central DB (CDB) that contains a description of the whole system. The new NetIS

version has similar structure, but the monitoring data fetched with SNMP are placed in the P-BEAST. P-BEAST was developed in ATLAS for permanent storage of operational data, and it has already been integrated with Grafana [8]. Grafana was chosen as a new data visualization service, but it has some limitations in the GUI implementation that makes the navigation less intuitive than in the legacy version. The main problem is that Grafana doesn't support the visualization of hierarchical data structures.

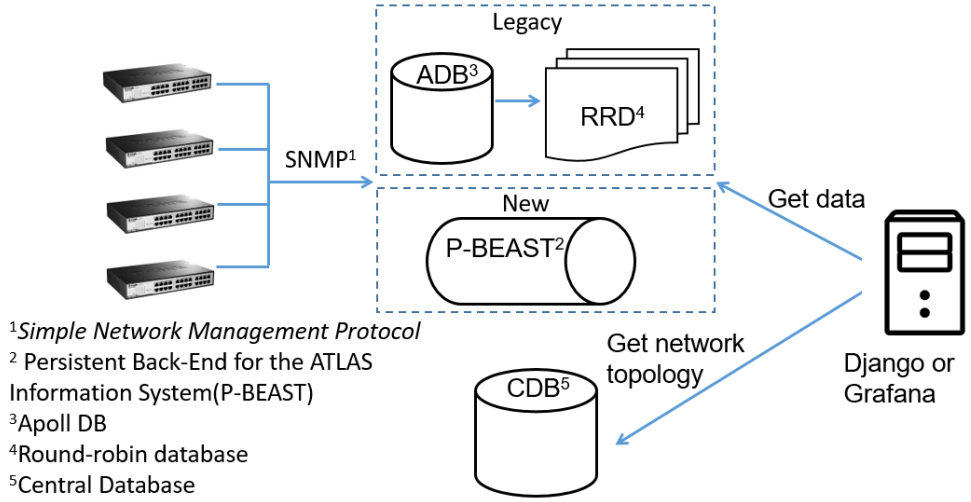

Figure 2. General structure of the NetIs Monitoring

3. The Graphics interface implementation of the Tree

The tree is the main GUI navigator in NetIs, but as mentioned Grafana does not support the Tree element. It is possible to add a tree to Grafana in the following ways: create a custom plugin or inject JavaScript code dashboards using the Grafana text panel widget. Both approaches have their pros and cons.

Creating a new panel is the main way to add new functionality to Grafana. This path requires the presence or acquisition of skills to improve Grafana itself. The plugin must be compiled as part of the Grafana developer workflow and installed on the server side. Within this path, to add any new function, for example, in our case we need to set the position for the dropdown menu, a new plugin is required. Unfortunately, this path does not guarantee compatibility with new versions. Sometimes some plugin does not work with new version.

The most effective way to add a tree into Grafana is to add JavaScript code to the text panel. This approach does not require a Grafana developer workflow because all the code will be placed in the Grafana text panel. The tree is not an element of Grafana itself and cannot use the data retrieval methods. It should directly use the JavaScript objects of the Grafana library to interact with other Grafana elements. This method can be easily adopted for any HTML/JavaScript object, including a dropdown menu.

The new version of NetIs uses JavaScript code in a text panel to implement the tree (Figure 3) and some other elements. The text panel is a basic element of Grafana. It supports HTML code and JavaScript inside it. This external JavaScript tree interacts with the just using the Grafana JavaScript code downloaded by the browser, which is included in the main HTML page. Using this library, the user can receive data from other panels and update dashboards. The NetIs uses tree as the navigation panel.

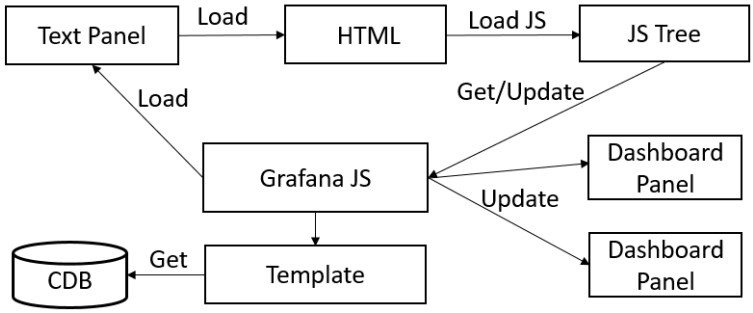

Figure 3. The scheme of interaction of the external JavaScript tree with Grafana

The initial data to build the tree for NetIs is stored in the CDB MySQL database. Grafana has a plugin for MySQL, but as already mentioned, the tree is not an element of Grafana so it cannot directly interact with a Grafana data source. Instead, the tree receives data using the Grafana template element. The template stores the data as a list of strings that needs to be created first out of the data retrieved from several tables in the MySQL database. The strings have the following structure: *Function*:*Device*::*Linecard*::*Interface*. Some devices do not have a linecard component, so in this case when converting data for such devices a special token, PORTS or LAG, needs to be generated.

The NetIs uses standard edition of the dhtmlxTree library for the tree implementation in HTML. The Initial method gets all strings of the template, parses them and generates the tree. NetIs uses special ID of tree node for detection the level of tree and getting the data required for the dashboards. The ID has the following format:

- Root level (function in CDB): _RR_*FunctionName*
- Level 1 (device in CDB): *DeviceId*
- Level 2 (linecard in CDB): _LL_*DeviceId*:*LinecardId*
- Level 3 (interface in CDB): _II_*DeviceId*:*LinecardId*:*InterfaceId*

Different levels of the NetIs tree have different context structures, with the exception of the tree area, which is always present in the upper left area. The tree uses the ID of selected node to generate content of the page using Grafana‘s JavaScript library. This approach allows the developers to make deeper changes in the structure of the monitoring page than those that can be done using standard Grafana tools.

4. View of NetIs monitoring

Figure 4 presents the NetIs view for a device (level 1). It has two dashboards with aggregated data for all the interfaces belonging to the device and two mini dashboards with aggregated data for each linecard of selected device. The first dashboard can change the network metrics displayed (e.g. from Packet/s to Octets/s) using the dropdown button on top of it. The following types of metrics are available: Discards/s, Errors/s, Link load, Octets/s and Packets/s. The second dashboard show “packets detailed” data, which includes a breakdown of the traffic into Unicast, Broadcast and Multicast packets. For the mini dashboards the same set of metrics as for the uppermost dashboard is available with its own dropdown button. The second mini dashboard displays links speed and status. To the left of the miniplots there is a table summarizing the host and linecard information.

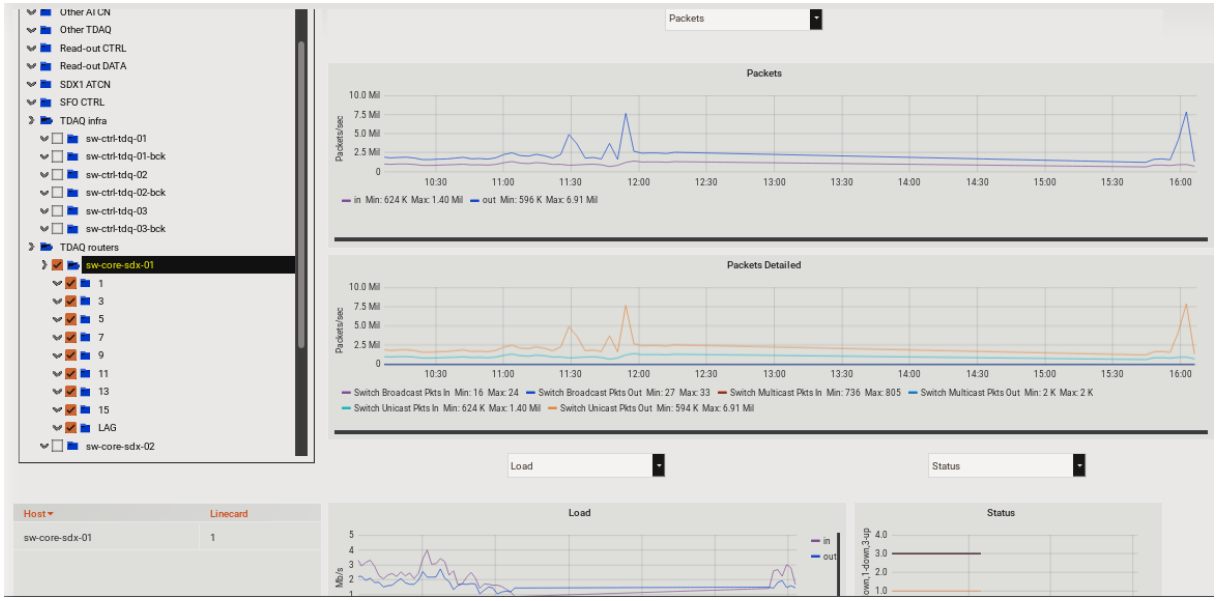

Figure 4. NetIs view for linecard (level 2)

The linecard view (level 2) has the same structure but uses the metrics gathered for the selected linecard and its interfaces. The interface view (level 3) does not display aggregated data (Figure 5), but instead it has a text area with additional information about the interface itself.

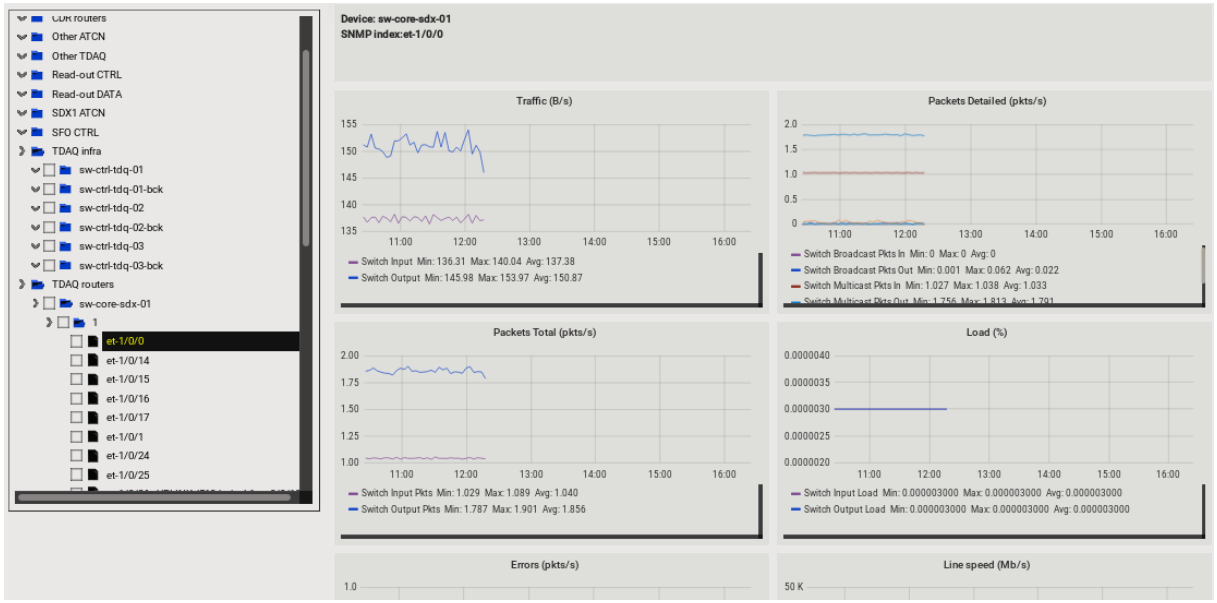

Figure 5. The interface view

5. Conclusions

The new version of NetIs was successfully implemented, tested and put into production. These services are based on Grafana and P-BEAST. The use the P-BEAST time-series database avoided the loss of data over time of the stored samples and does not degrade the resolution of the dashboards. Grafana makes the NetIs page navigation more dynamic and flexible. Maintaining the new system should be easier than before because only knowledge of Web and JavaScript technologies is required to support the NetIs service. The monitoring system will be evolved and updated following operational experience.

References

[1] ATLAS Collaboration 2008 The ATLAS Experiment at the CERN Large Hadron Collider, JINST 3 S08003 doi:10.1088/1748-0221/3/08/S08003

[2] *D. Savu, A. Al-Shabibi, B. Martin, R. Sjoen, S. Batraneanu and S. Stancu 2010 Integrated System for Performance Monitoring of the ATLAS TDAQ Network*, Journal of Physics: Conference Series. 331 052031 doi:10.1088/1742-6596/331/5/052031

[3] RRDtool: https://oss.oetiker.ch/rrdtool/index.en.html

[4] A. Sicoe, G. Lehmann, Luca Magnoni, S. Kolos, I. Soloviev 2012 A persistent back-end for the ATLAS TDAQ online information service (P-BEAST), Journal of Physics: Conference Series 368 (2012) 012002 doi:10.1088/1742-6596/368/1/012002

[5] Matplotlib: https://matplotlib.org/

[6] Django: https://www.djangoproject.com/

[7] JavaScript Tree: https://dhtmlx.com/docs/products/dhtmlxTree/

[8] Grafana: https://grafana.com/