Integrated System for Performance Monitoring of the
ATLAS TDAQ Network

Dan Octavian Savu, Ali Al-Shabibi, Brian Martin, Rune Sjoen, Silvia Maria Batraneanu, Stefan Stancu

1 CERN, 1211 Geneva 23, Switzerland
2 University of Heidelberg, Germany
3 University of Oslo, Norway
4 University of California, Irvine, USA
5 “Politehnica” University of Bucharest, Romania

E-mail: dan.savu@cern.ch

Abstract. The ATLAS TDAQ Network consists of three separate networks spanning four levels of
the experimental building. Over 200 edge switches and 5 multi-blade chassis routers are used to
interconnect 2000 processors, adding up to more than 7000 high speed interfaces. In order to
substantially speed-up ad-hoc and post mortem analysis, a scalable, yet flexible, integrated system for
monitoring both network statistics and environmental conditions, processor parameters and data
taking characteristics was required.

For successful up-to-the-minute monitoring, information from many SNMP compliant devices,
independent databases and custom APIs was gathered, stored and displayed in an optimal way. Easy
navigation and compact aggregation of multiple data sources were the main requirements;
characteristics not found in any of the tested products, either open-source or commercial.

This paper describes how performance, scalability and display issues were addressed and what
challenges the project faced during development and deployment. A full set of modules, including a
fast polling SNMP engine, user interfaces using latest web technologies and caching mechanisms, has
been designed and developed from scratch.

Over the last year the system proved to be stable and reliable, replacing the previous performance
monitoring system and extending its capabilities. Currently it is operated using a precision interval of
25 seconds (the industry standard is 300 seconds).

Although it was developed in order to address the needs for integrated performance monitoring
of the ATLAS TDAQ network, the package can be used for monitoring any network with rigid
demands of precision and scalability, exceeding normal industry standards.

1. Introduction

The Integrated System for Performance Monitoring of the ATLAS TDAQ Network (Net-IS) is a
scalable data retrieval and graphing system used for real-time and historical statistics visualization. Its
primary purpose is to gather, store and display numeric time-series plots using an efficient and user
friendly web interface.

The system consists of a data gathering and storage backend, an intermediate cache layer for speed
improvement, and a scalable, database driven, web-based user interface. Using a bottom-up approach
we describe the general architecture followed by a detailed description of each module. Finally we
address deployment options and user interface topics, such as choice of technology, ease of navigation and consistency.

2. ATLAS network monitoring

The network reliability in large physics experiments, such as ATLAS, is considered to be critical. However, malfunctions are inevitable on any network. To shorten the unavailability time, the most important goal is to detect them at an early stage and perform efficient troubleshooting. Running network performance monitoring and having an integrated visualisation application are critical to rapidly understand the situation.

SNMP [7] is a protocol for managing devices on IP networks. Traditional performance monitoring is based on polling SNMP counters at the industry standard interval of 5 minutes. Although precise enough for consumer networks, this polling interval fails to meet the needs for performance monitoring on mission critical networks, where the highest achievable rate is needed. Because for most network devices the SNMP counters are refreshed every 1 to 15 seconds, we consider an interval of 30 seconds to be small enough for our purpose.

While SNMP is sufficient for an overview of the network performance, it fails to provide detailed information about specific network flows. In addition SNMP is not usually supported by devices controlling the environment or data acquisition software infrastructure.

sFlow [3] is a standard which enables an Ethernet switch to take samples of the packets traversing it and sends them to a collector for permanent storage. The packet samples are analyzed in software and network flows can be individually traced.

Network monitoring of the ATLAS network is accomplished by gathering SNMP counters, sFlow samples and statistics from in-house developed tools. Currently there are 5 applications who gather network related and complementary statistics: an SNMP polling engine (apoll), one sFlow collector, environmental conditions collectors, data acquisition related statistics tools and NAGIOS [13] for computer farm monitoring.

The RRD [5] format is an open source industry standard for high performance logging and storage of time series data. As most of the gathered statistics are time-series values, we store them using a hierarchical structure of multiple RRD file sets. The metadata, such as network topology or device grouping, is stored using a MySQL relational database.

With so many independent monitoring applications it is difficult to cross-correlate statistics and get an overall system performance view. Finding, building or extending one interface to replace all the collector frontend interfaces, while accommodating the existing data sets, became a necessity. We tested several of the commercially available solutions [8], including CA Spectrum [6] or HP OpenView [11], and non-commercial open source products such as CACTI [9] or NAGIOS. While some of the solutions were simply not flexible or scalable enough, others lacked the ability to collect all the monitored parameters or to provide tight integration with our current tools.

Developing our own solution was the only way to have a flexible central monitoring panel without dropping any of the available statistics sets. Also it was much easier to integrate existing metadata information, such as network topology or logical grouping, in a user friendly manner.

3. System architecture

The developed system was designed on the following principles:

- Gather as much data as possible about the system, including complementary statistics such as environmental conditions and data acquisition infrastructure status;
- Incorporate time-series data from any other applications;
- Save statistics using a scalable and flexible storage structure while providing fast access;
- Provide one single application to display all the plots regardless of the data source;
- Use the same look and feel through the entire interface.
The architecture, depicted in Figure 1, comprises 3 main parts: the backend, containing all the collector tools and data repositories, an intermediate cache layer, to reduce the response time, and the frontend, implemented as a web application. While the frontend is fully developed in python [5] for flexibility, the backend is a combination of high performance multithreaded C programs and various script based programs to ensure compatibility with external applications.

Figure 1. System architecture

3.1. Collectors and storage (backend)
Any open source or commercial application which exports statistics as time-series values can be used as a data provider and can become a Net-IS collector. If its storage format is already RRD, then it is fully compatible with the system. Otherwise a plug-in is required to convert the provided statistics into RRD format.

Although the system was primarily designed to make use of time-series statistics provided by various external monitoring applications, we opted for developing an in-house SNMP polling engine and sFlow sampling collector. The decision of building these important collectors in-house was taken because none of the tested solutions, such as Spectrum, Spine, sFlowTrend or nTop, provided the required scalability, accuracy or flexibility.

3.2. Fast access cache layer
Studies on the overall system usage showed that ~95% of the plot requests are being made for statistics gathered in the last 72 hours. The statistics, packed in RRD files, are stored on normal hard drives or using a NAS, due to their large volume. With an average of a few hundreds IO operations for these media types, the bottleneck soon becomes fetching data from RRD for plot generation. Having all the statistics for the last 72 hours also stored on a fast access medium, such as RAM, greatly improves the application response time while reducing the server overload.

This mechanism is implemented in Net-IS using a filesystem stored in memory (ramfs) and acts as a cache between the storage and the web application. The cache contains a complete set of RRD files synchronized with permanent RRD files in real-time. To increase the cache size, while maintaining a low cost, single-level cell Solid State Drives can be used instead of RAM.
3.3. Storage Abstraction, Aggregation and Navigation Layer

In a distributed storage configuration the data is fetched from various RRD sets stored at different network locations. To support such a deployment, the system uses a variable locator identifier which is translated by the storage abstraction layer into a path, a file name and a RRD data source. The variable locator is similar to the URL used for accessing content over internet. Its structure is built from a domain name (i.e.: net for networking, sys for system administration), a resource (i.e.: device, interface) and a parameter name (i.e.: ifInOctets, cpuLoad). Examples of variable locators include net.interface.ifInOctets, for network interface input traffic, and sys.device.cpuLoad, for computer CPU load. By using variable locators the upper layers are agnostic to storage format, location or variable name inside a RRD file.

The aggregation layer extends the variable locator concept by using regular expressions to group a set of variables. This group can be used to generate an individual overlay plot or to generate an aggregated plot by using an aggregation function such as sum or average.

Accessing the same data using different views is one of the features provided by the navigation layer. Sometimes, the plots need to be displayed or grouped according to the physical location of devices (i.e.: same rack), other times the logical grouping should be taken into consideration (i.e.: file servers regardless of their location), while user defined grouping is the best choice when looking for a specific problem (i.e.: switch uplinks). The navigation layer takes care of these manipulations and also provides good scalability when many collectors are integrated with the system.

Web applications are state-less by default. To keep state between successive requests, one has to dump the current session state and restore it once a new request is received, with time and CPU penalties. To keep state between database requests an XML-RPC [14] server is implemented in Net-IS. This approach reduces the load on the web server, simplifies the web code and improves the overall response time.

3.4. User interface

All the collected data is presented to the user through a cross-browser web application, designed to offer a simple and intuitive interface to visualize monitored parameters (illustrated in Figure 2). The interface provides the same look and feel for all data sources and views.
The web application is implemented using the latest client-side technologies, such as xhtml, css, js, dhtml and ajax [16]. On the server side it is written in python [12] and makes use of the Django [4] framework for template-based data presentation. The content delivery is done using an Apache [10] application server through the WSGI [15] interface.

From a design perspective, the interface is structured in 3 panels:
- Top panel, used for time interval, data source and regular expression filter settings;
- Left panel, used for hierarchical navigation;
- Central Panel, where the plots are displayed and further custom settings become available.

While the top panel is static, the navigation panel uses ajax to adapt its content based on the data source and view type. For a specific data source and view type, the navigation tree fetches its content dynamically from the server while the user navigates deeper in the hierarchical structure. This feature makes the navigation scalable, since just a small amount of information has to be loaded during the initialization process.

All the time-series plots are generated through rrdgraph, a tool included in the RRD package. Having a separate module to handle plot generation means that the rrdgraph technology can be changed at any time, without restructuring the whole application. For non-time-series plots there is already support for matplotlib, since not all the sFlow statistics are time-series compatible.

The aggregation layer, together with the plot generation engine, allows the interface to display an aggregated plot, along with individual plots that make up the aggregated one, on a single page. This feature reduces the time spent analysing traffic patterns on various network segments and computer farms.

4. System deployment

Each of the system’s subcomponents - collector, storage and frontend – can be deployed either as a single instance or in a distributed configuration. This flexibility gives many deployment options from which the most important ones are depicted in figures 3, 4, 5 and 6.

We started with a multi collector partially distributed configuration (Figure 4) and gradually evolved into a system using multiple collectors, multiple storage locations and one web application instance (Figure 5).

![Figure 3. Non distributed configuration](image)

![Figure 4. Partially distributed configuration (multi collector)](image)

![Figure 5. Partially distributed configuration (multi collector and multi storage)](image)

![Figure 6. Fully distributed configuration (multi collector, storage and application)](image)
The Net-IS architecture scales horizontally on both the frontend and the backend. If deployed in a fully distributed configuration (Figure 6) the system will also be fault tolerant. In such a configuration losing a backend machine will cause a minimal amount or no data loss, while losing a frontend machine will cause no disruption of the service.

5. Conclusion
Net-IS has been developed to significantly reduce the time needed by an expert to investigate network-related problems, either ad-hoc or post-mortem. The time needed to obtain a cross-correlated report between different statistics (i.e. network traffic vs. CPU load) has been reduced from minutes to seconds. Any desired plot can be fetched with at most 4 clicks. It succeeded to be an all-in-one replacement for the various visual tools previously used for tracing network and system anomalies.

By reducing the set of tools needed for troubleshooting problems, the application allowed non-experts to do a first level check before passing a problem to an expert and thus reduced unnecessary load on experts.

Future work consists in developing an expert system, on top of Net-IS, to flag network related problems. By performing automated online and offline statistics analysis, the expert system will provide feedback to assist a network expert in identifying failure causes.

References
[1] M. Ciobotaru, L. Leahu, B. Martin, C. Meirosu and S. Stancu
“Networks for ATLAS trigger and data acquisition”, in Proc. Computing in High Energy Physics (CHEP 06), Mumbai, India, Feb. 2006.
[2] S.M. Batraneanu, A. Al-Shabibi, M. Ciobotaru, M. Ivanovici, L. Leahu, B. Martin and S. Stancu, "Operational Model of the ATLAS TDAQ Network, Proc. IEEE Real Time 2007 Conference, Chicago, USA, May 2007.
[3] R. Sjoen, S. Stancu, M. Ciobotaru, S.M. Batraneanu, L. Leahu, B. Martin and A. Al-Shabibi "Monitoring Individual Traffic Flows within the ATLAS TDAQ Network", CHEP, Prague, Czech Republic, 2009
[4] Django, The Web framework for perfectionists with deadlines. [Online]. Available: http://www.mrtg.org/rrdtool/
[5] RRDTool, OpenSource industry standard, high performance logging and graphing system. [Online]. Available: http://www.djangoproject.org
[6] CA Spectrum Infrastructure Manager [Online]. Available: http://www.ca.com/us/root-cause-analysis.aspx
[7] SNMP, Simple Network Management Protocol [Online]. Available: http://en.wikipedia.org/wiki/Simple_Network_Management_Protocol
[8] Wiki Comparison of network monitoring systems [Online]. Available: en.wikipedia.org/wiki/Comparison_of_network_monitoring_systems
[9] CACTI [Online]. Available: http://www.cacti.net/
[10] Apache HTTP Server Project [Online]. Available: http://httpd.apache.org/
[11] HP OpenView [Online]. Available: http://www.managementsoftware.hp.com/
[12] Python [Online]. Available: http://www.python.org
[13] NAGIOS [Online]. Available: http://www.nagios.org/
[14] XML-RPC Protocol [Online]. Available: http://en.wikipedia.org/wiki/XML-RPC
[15] WSGI, Web Server Gateway Interface [Online]. Available: http://www.wsgi.org
[16] WWW Technologies [Online]. Available: http://en.wikipedia.org/wiki/World_Wide_Web

International Conference on Computing in High Energy and Nuclear Physics (CHEP 2010) IOP Publishing
Journal of Physics: Conference Series 331 (2011) 052031 doi:10.1088/1742-6596/331/5/052031