New data visualization of the LHC Era Monitoring (Lemon) system

FEDORKO Ivan¹, LEFEBURE Veronique, LENKES Daniel, PERA MIRA Omar

CERN - European Organization for Nuclear Research CH-1211 Genève 23, Switzerland

E-mail: Ivan.Fedorko@cern.ch

Abstract. In the last few years, new requirements have been received for visualization of monitoring data: advanced graphics, flexibility in configuration and decoupling of the presentation layer from the monitoring repository. Lemonweb is the data visualization component of the LHC Era Monitoring (Lemon) system. Lemonweb consists of two sub-components: a data collector and a web visualization interface. The data collector is a daemon, implemented in Python, responsible for data gathering from the central monitoring repository and storing into time series data structures. Data is stored on disk in Round Robin Database (RRD) files: one file per monitored entity, with set of entity related metrics. Entities may be grouped into a hierarchical structure, called “clusters” and supporting mathematical operations over entities and clusters (e.g. cluster A + cluster B / clusters C – entity XY). Using the configuration information, a cluster definition is evaluated in the collector engine and, at runtime, a sequence of data selects is built, to optimize access to the central monitoring repository. In this article, an overview of the design and architecture as well as highlights of some implemented features will be presented.

1. Overview

Lemon [1] is a server/client based monitoring system. On every monitored entity, a monitoring agent with sensors is responsible for retrieving the monitoring information. The monitored samples are stored in a local cache and also forwarded to the application server. The application server stores the received samples in a central monitoring repository which can interface to either a relational database or a flat-file backend. A web based interface, called Lemonweb, is provided for the visualization of the data.

Lemon is part of the Extremely Large Fabric management system (ELFms) tool suite [2] but has no functional dependencies on other ELFms components and can be used separately [3].

¹ To whom any correspondence should be addressed.
2. Lemonweb concept
The Lemon front end application, called Lemonweb, is used to retrieve metric information from the monitoring repository, to store it into time serialized data structures and to provide data visualization. Lemonweb consists of two sub-components: a data collector and a web visualization interface.

The data collector, called lemonmrd (LEMON Monitoring Repository Daemon), is a daemon, implemented in Python, responsible for data gathering from the central monitoring repository and storing into time series data structures. Data are stored on disk in Round Robin Database (RRD) files: one file per monitored entity, with a set of entity related metrics. A data collector supports entity data aggregation (e.g. hosts) into groups (by cluster, by rack, by hardware model, etc.) and provides a summary or average view of each group, even if certain entities may be part of multiple groups. For more details about aggregation option see section 4.

Data from the RRD files is then passed to the web interface for visualization, mainly visualized as time trends in the RRD graphs. Lemon-web supports the federation of Lemon instances for web interface functionality such as search, as well as for data aggregation from multiple monitoring repositories.

Identical configuration sources are used by the visualization part as well as by a data collector producing RRD data files. The current implementation supports a monitoring repository based on an Oracle backend, nevertheless support for other implementations such as flat files or MySQL may be added in the future.

3. Lemonweb interface workflow
The new Lemonweb is a PHP application to visualize monitoring information, stored in RRD files or accessed directly from the monitoring repository. The Lemonweb provides visualization for various predefined types (e.g. hosts, clusters of hosts, services) depending on the specific configuration parameters (e.g. list of metrics for which RRD graphs shall be displayed).

Recently the front-end framework was redesigned and implemented using the Model-View-Controller pattern to separate the workflow logic from the user interface. The core of the modular design is the simple template engine which is using predefined templates for various types (see Figure 2). Every template (e.g. for host, cluster, sensors, overview metrics list, etc.) is combined with the customized configuration, defined by the user (e.g. user may limit or extend the list of visualized metrics graphs), and content gathered from the monitoring repository or the RRD files. To speed up the final target page generation process, all the available configuration information and available data graphs are cached by Memcache [5]. User access is done through a single entry point and integrated with the CERN Single-Sign-On authentication. Depending on the user request, the access rights are
checked against an authorization plugin (currently CERN group management system is supported) before generating the final page.

**Figure 2** The new Lemon front-end framework workflow is designed with a single entry point, optimized caching and using a template engine to separate the workflow logic from the user interface visualization.

4. **LEMON Monitoring Repository Daemon**

The Lemon Monitoring Repository Daemon is a data collector implemented in Python which collects monitoring data from the central Monitoring Repository.

Firstly the entity hierarchy, defined in the configuration and called “clustering”, is evaluated and afterwards sequences of database select queries are built at runtime. The main aim is to minimize database access and optimize mathematical operations for aggregation structures like cluster of hosts (i.e. avoid database select for entity form cluster during aggregating operation). For example: for cluster D defined as “cluster A + cluster B /clusters C – entity XY”, a collector prepares the list of all entities which belong to cluster A, B and C. After preparing the select query and gathering the entity metrics from the monitoring repository, the clusters A, B and C aggregation is calculated. Afterwards “cluster A + cluster B” and “clusters C – entity XY” fractions are evaluated with final cluster D aggregation. The monitoring metrics as well as the aggregation results are stored in the entity’s RRD files, where any hierarchy object (e.g. cluster A) is considered as a standalone entity with its own RRD file. Currently supported aggregation operations are sum, subtraction, multiplication and division.

The data collector provides monitoring of the aggregation process by storing in the aggregated entities RRD files the number of expected and currently reported monitored entities which are used in the actual aggregation process and visualized in the reliability graphs (see section 5).

The current CERN Lemonweb instance is handling ~17k RRD files. The new collector daemon has improved the start-up time by a factor 30, with respect to the previous collector implementation, and a single calculation loop over all entities sped up by a factor 100.
Figure 3 Lemon Monitoring Repository Daemon collects all metrics for the given entity and stores them in the single RRD file per entity. Entities defined by multi-level aggregation are supported as well.

5. Lemonweb visualization

The new functionality provided by the Lemonweb visualization is based on the improvements of the underlying framework as described in the sections 3 and 4. The new data collector provides improved RRD data time precision and setting the lowest time step down to the level of seconds. A typical metric taking period of CERN’s Lemon instance varies from 0.5 to 10 min. The previous implementation suffered from an increasing time step (over 10 min) when the amount of monitored entities was too large (over 15k). The visualization is provided for the predefined types of entities like hosts, clusters but also services and usually consists of text base metrics (e.g. “operating system”) and graphs, trends and pie charts for numerical metrics (see an example on Figure 4). A user may define customized RRD graphs as well.

All visualized RRD graphs provide a link to a graph widget which can be embedded in 3rd party dashboards. At the same time an HTML5 implementation of RRD graphs visualization (see Figure 5), based on an adapted JsRRDGraph library [6], is provided and allows dynamic scaling and trends time browsing.

As discussed in section 4, aggregation calculation is storing information about the number of expected and currently reporting entities. This information is visualized in reliability graph, as shown in Figure 6, and provides additional knowledge for analysis of data trends.

One of the Round Robin Database storage features is losing precision of historical data, because of smoothing over the time period. While this is not harming trends graphs, affects data mining and root cause analysis of detected problems. Lemonweb provides visualization time windows for given metrics and entity based on a high granularity data read out directly from the monitoring repository. Nevertheless in the monitoring repository only information for the monitored entity (e.g. hosts) is stored and the aggregated information (e.g. for cluster) must be computed.
The data collector was enhanced to provide export of metric data, from the central monitoring repository, with runtime aggregation based on Lemonweb configuration and storing the result as CSV files on disk. At the same time an interface for a dynamic visualization of such export is provided.
Figure 5 An advanced graph offers the possibility to look at the graph interactively, by scaling and modifying the time window.

Figure 6 Reliability graph example. In the left graph is an example of power measurement aggregated across a set of the intelligent power distributed units. The drop in the aggregated graph could be interpreted as a decrease of the measured values. In the right graph the reliability graph which shows the number of expected entities contributing to the aggregated entities (green line) and currently reporting entities (red line) is visualized. The drop in the measured power data can be correlated with the missing reporting entities.

6. Conclusion

The new Lemonweb is based on an enhanced framework which separates user visualization from workflow logic. The data collector offers an effective processing of mathematical operations for aggregated entities. Visualization profits from all enhancements and offers customization of user views, embedding and dynamic scanning of trends graphs. Data analysis functionality is supported with the possibility as exporting, with high granularity, also aggregated entities at runtime.

References

[1] LEMON – LHC Era Monitoring for Large-Scale Infrastructures, J. Phys.: Conf. Ser. 331 052025, 2011
[2] ELFms project, http://elfms.web.cern.ch/elfms/
[3] Lemon project, http://lemon-monitoring.web.cern.ch/
[4] RRDtool, http://www.mrtg.org/rrdtool/
[5] Memcache caching daemon http://www.php.net/manual/en/book.memcache.php
[6] JsRRDGraph library https://github.com/manuelluis/jsrrdgraph