Journal of Physics: Conference Series **331** (2011) 022037 doi:10.1088/1742-6596/331/2/022037

A Universal Logging System for LHCb Online

Fotis Nikolaidis[1], Loic Brarda[2], Jean-Christophe Garnier[3] and Niko Neufeld[4]

[1 2 3 4] European Organization for Nuclear Research (CERN), CH-1211 Geneva 23 Switzerland

E-mail: fotis.nikolaidis@cern.ch, loic.brarda@cern.ch

Abstract. A log is recording of system's activity, aimed to help system administrator to traceback an attack, find the causes of a malfunction and generally with troubleshooting. The fact that logs are the only information an administrator may have for an incident, makes logging system a crucial part of an IT infrastructure.
In large scale infrastructures, such as LHCb Online, where quite a few GB of logs are produced daily, it is impossible for a human to review all of these logs. Moreover, a great percentage of them as just "noise". That makes clear that a more automated and sophisticated approach is needed.
In this paper, we present a low-cost centralized logging system which allow us to do in-depth analysis of every log.

1. Introduction - Requirements

Sources - Transfer

Due to heterogeneity of LHCb Online network, many different types of logs are involved. They differ not only in format, but also by the method they are extracted. The can be divided in four categories: Syslog, FMC, PVSS and Windows logs.

FMC (Farm Monitoring and Control System) is an integral part of the ECS (Experiment Control System), which is in charge of monitoring and controlling every on-line component. PVSS II is a SCADA (Supervisory Control and Data Acquisition) system responsible for initializing, configuring and operating devices. Totally we have around 30000 sources of logs. Thus if we want to have low overhead we should avoid complex protocols. For this reason we decided to use UDP and be tolerant on some log losses. Moreover, if we used a reliable protocol such as TCP, we may had problems with resources to the central server, because every connection requires a dedicated socket. It worth pointing out that since the LHCb Online network is secure and private, no authentication or encryption is needed.

Storing

Regarding storage schema, was clear from the beginning that a centralized approach was the only way. This approach would make analysis and correlation easier since we have all the necessary information together. Moreover, it provides security because only certain people have access to these logs.

One of the first dilemmas we had, was to decide if it is better to save logs in textfiles or in a database. The solution to this problem came from analysis part. The software we decided to deploy is based on file indexing. That forced us to save them in textfiles.

Because of this choice, we inherited some "problems" that do not exist in databases. These problems are related to classification, rotation, compression.

Classification criteria could be the source, the application, and the facility of an incoming log. Regular rotation is needed because if a file grows it will be time-consuming to search for something. Compression is needed for reducing the used space. Because of log repetition, compression rate is very high and thus we save up to 70% of space.

Analysis

Real Time Detection

In some cases, we want a system that is able to react as soon as possible when an event occurs. Such a system could be useful in detecting an intrusion, a malfunction, a DOS attack or any other inappropriate activity in real time. Ideally, based on the urgency of the event the system should raise an alert or in more trivial cases a custom script should be executed. Moreover, real time correlation would be an extra bonus. For example, if an event occurs 5 times per second instead of 2 times per second which is the usual, then raise an alert.

Back Tracing

Even though the above system may alert us on an event, it is not capable to help us find the security hole that was exploited or what caused the malfunction. It is obvious that another system is needed. A system that is able to trace back a "chain" of events until we find the hole / cause. Features such as event correlation, flexibility in search patterns (regular expressions, time periods), statistical plots and a nice representation of the results are considered mandatory. And of course, we need this system to be able to handle all of our logs traffic as fast as possible.

Additional Requirements

Some other secondary requirements are related to SECURITY, AVAILABILITY, REDUNDANCY. Because we do not want an intruder to be able to manipulate logs in any way, we must be sure that proper ACL (Access Control List) have been deployed. Moreover, we need a way to ensure that the services are running and that data will not be lost in case of a crash.

2. Available Software

Transfer - Storing

As mentioned earlier we need software that is able to handle any kind of log format. In this chapter we present a comparison of available software both for Unix and Windows.

(Unix) Syslog Implementations

- Syslogd : Was the standard logging application for Unix systems and its derivatives (Linux, *BSD, Solaris, ...). Although it served Unix community for many decades, it is considered obsolete and being replaced by other applications such as Syslog-ng and Rsyslog.
- Syslog-ng : has several features surpassing Syslogd, including reliable message transferring using the TCP protocol, secure transfer with TLS, SQL database support (MySQL, PostgreSQL) and flow control for handling minor server outages.
- Rsyslog : is an enhanced multi-threaded Syslogd. Among others, it offers support for on-demand disk buffering, authentication, encryption, scalability, RELP, redirection to databases (MySQL, PostgreSQL, Oracle), email alerting, flexible output formats (including high-precision time-stamps), flexible filters, on-the-wire message compression, and text-to-syslog convert. It is now the default syslog daemon for many Linux distributions.

Windows Implementations

- Snare : is a Windows NT, Windows 2000, Windows XP, and Windows 2003 compatible service that interacts with the underlying Windows Eventlog subsystem to facilitate remote, real-time transfer of event log information.
- SyslogAgent : is a Windows add-on, allowing Windows EventLog events as well as other Windows applications logs to be sent to a syslog server. SyslogAgent is installed as a transparent service on Windows.
- Kiwi : is easy to set up and configure, it offers a feature-rich solution for receiving, logging, displaying, alerting, and forwarding syslog and SNMP trap messages

Analysis

For the analysis software we tested Sawmill, Eventlog Manager, OSSIM, Prelude, OSSEC, Splunk.

Sawmill works with profiles. Each profiles is used to describe a different log format. Due to heterogeneity of LHCb Online network it is not very efficient for us.

Eventlog Manager is a web based real time event log manager and is able to react on certain events. It is written in Java and uses MySQL, which decreasing its performance for large scale infrastructures.

OSSIM is a compilation of many well known administrating tools such as arpwatch, Nessus, Snort, Nagios etc. Although it is a great combination, it is not what we are looking for.

Prelude is a real time log analyzer. It can interact with other frameworks such as Snort via IDMEF messages (Intrusion Detection Message Exchange Format). Event correlation rules are written in Python.

OSSEC stands Open Source Host-based Intrusion Detection System. It performs log analysis, file integrity checking, policy monitoring, rootkit detection, real-time alerting and active response. For this project, we are mostly interested in its log analysis machine. There are reports showing that it can be used as LIDS (Log Intrusion Detection System).

Splunk supports searches on logs either by regular expressions or by time periods. It is based on file indexing and log normalization. Normalization is used for semantics searches, because ERROR and FAIL are considered to be the same. Except for its Console interface, it also provides us with an exceptional web interface. Statistical analysis, saved searches, zoom in/out are some features of it. Because of its general architecture, it can be used for any kind of logs.

3. Setup

Retrieval / Transfer

Syslog (Unix)

Logs that are produced from syslog syscalls are handled by the local syslog daemon. In this case the only thing we have to do was to adjust its configuration file appropriate in order to forward logs to the log server. An exception to this, are the logs of web servers, where change of httpd.conf is necessary. We use Rsyslog as default syslog daemon, except for hosts that are running SLC4. Massive deployment of configurations is made by Quattor. In figure 1 there is a graphical representation of LHCb Online logging topology.

FMC

FMC uses a proprietary application, to handle its logs, which is called LogViewer. Similar to Syslog daemons, it is distributed and every instance of it can be used either as final (storing) or intermediate (forwarding) node. The only thing we can do for integration is to save LogViewer under the same paths as Syslog logs. More details are following.

PVSS

PVSS produces a huge amount of logs that are saved on files instead of being forwarded on a daemon such as Syslog or LogViewer. Because most of these logs are "noise", in order to simplify the analysis part, we have to filter out logs that contain certain keywords. For the filtering, an intermediate node is deployed.

Every host that is running PVSS send its logs to the intermediate node. Because they may originate from different sources, filename must be contained within log. After being received, filtering is taking place. If logs match a certain rule they are forwarded to the log server with proper severity. If the log contains keywords *ERROR, FAIL, PROBLEM* it is forwarded with ERROR facility. If it contains *WARNING, EXCEED, ALERT* it is forwarded with WARNING facility. If it does not match any of the rules it is rejected.
This way we try to emulate filtering-by-severity behavior, which is not supported by default by PVSS. Moreover, we can use the intermediate node for log throttling and thus avoid congestion to the central server. Before being processed, logs must be transformed in Syslog compatible format. To do that, logs saved in textfiles must be forwarded to the local Syslog daemon. There are many ways to do that, but we present only two of them.

- Imfile is an Rsyslog's module that can send the contents of a file directly to Syslog's core. For every file that we want to transmit, we can use different priority and different tag. If we transfer the filename inside tag we can create an exact copy of the original file, in Syslog format.
- The other approach is a combination of "tail -f" and "logger". Tail is used to print the appended lines of a file and logger is used to forward these lines to Syslog. Priority and tag are the same for every file that is transmitted.

To test the performance, random logs were appended to a set of 100, 250 files with intervals 0s and 3s. As we can see in table 1 imfile losses in performance. This is because tail uses inotify and polling, while imfile uses only polling. Nevertheless we preferred imfile due to its flexibility with priority and tags. Moreover, it is using only one PID and is easy to check its status with Nagios. This would be more difficult "with tail | logger" because of multiple, random generated PIDs.

Windows Syslog Messages

Because Windows does not support Syslog by default, we had to use an external application. We choose to use SNARE because it can forward not only EventLogs but logs that have already been saved. The last feature is used for PVSS filter like mentioned above.

Central Server / Storing

Hardware - Software

For the logging system to work properly, it was necessary to be sure that the central logging server is capable enough to handle all the log traffic. In addition, we want to add some redundancy.
The current hardware for the log server is:

- Dual Xeon 3 GHz / CPU 8 GB Ram / 2x250 GB HDD in in mirror mode
- SLC 5 with Ext3, NFS, Rsyslog 5.4.0, Logrotate 3.7.4
- Redundant power supplies

Storing schema

The syslog daemon on the server is a stripped version of Rsyslog. As first classification, we have 5 categories based on the host purpose. FARM, HOSTS, PARTITIONS, PVSS, SERVICES. In addition, there are 2 more files, for OOM events and unmatched logs. As a second classification,

Table 1: Loss comparison between Imfile and custom Script

FILES	250	100	250
INTERVAL	3s	3s	0s
Imfile	8.86%	10.08%	95%
Script	0.93%	0.65 %	33.1%

hostname is used. As a third and last level we save them based on their content (SECURITY, MESSAGES). Regarding rotation, logs are rotated once a week and kept for three months. Summarizing the above, we end up with a general tree structure like HOSTS/host01/{messages, security, messages[1-12], security[1-12]}. More files may be included depending of the next cases. Because these logs may be used as input to some application, they are exported via NFS as read-only. For security reasons, only members of administrator group have access to SECURITY logs.

Special Cases

- Web logs : Are saved both locally and in central server. Locally saved logs are rotated by Chronolog for better management. Transmitted logs, include the filename as a tag. This way we can reconstruct the initial file in Syslog format and keep it on the server.
- PVSS : As mentioned above, the intermediate node is sending logs with severity depending on the containing keyword. This way we create a tree structure like PVSS/ERROR/{host01, host02} , PVSS/WARNING/{host01, host02}.
- FMC : In order to simplify the analysis part, we try to keep files as organized as possible. Thus, for nodes that are running FMC, one more file is added. Because FMCLOG grows too fast and it is used as input on certain applications, we had to change the rotation schema. It is rotated both by time period (daily) and by file size and we keep compressed 80 instances.

It worth to mention that regular Syslog logs are not in conflict with the special cases. For example, if a machine is running PVSS it will have entries both in HOSTS/host01 (regular Syslog messages) and PVSS/WARNING/host01 (PVSS case).

Analysis

As being said, analysis is divided in real time and traceback. For the real time we use OSSEC because it offers alerting, responding and event correlation. Moreover it comes with many predefined rules and it is easy to write new ones if is needed. Regarding traceback, we use Splunk because it conforms with all of our requirements and it can be combined with OSSEC via a plugin call Splunk-For-Ossec.

Due to high I/O, deploying analysis frameworks on a virtual machine is not viable solution. For this reason we use a real machine with Dual Xeon 3 GHz, 80 GB Raid01 mirrored HDDs, 8 GB RAM running SLC version 5.5, Ext3 and Splunk version 4.1.

4. Other topics

Availability

To ensure logging availability, services such as Syslog and LogViewer are monitored by Nagios. Further, Munin is responsible to alert us if something urgent is happening to a machine (Disk Space/RAM Space/High Temperature etc).

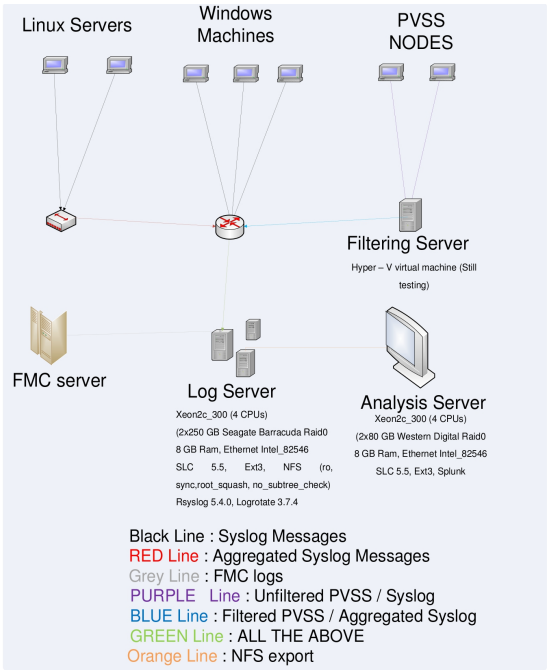
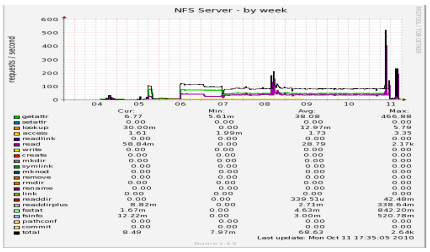
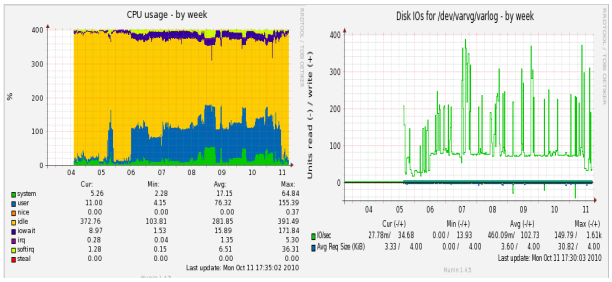


Figure 1: LHCb Online topology



(a) NFS



(b) CPU (c) DISK

Figure 2: Log Server performance measurements

Log Server Performance

In order to test the performance of our logging server, we produced an average traffic of 6 GB/day. Actual log traffic may vary from 3 GB/day to 9 GB/day depending on LHCb experiment status.

As we can see in Munin plots (2a)(2b)(2c), although the traffic was double than normally, only 1/3 of server's resources were consumed. In case of burst traffic, if the output device is too slow, Rsyslog is capable of saving logs in RAM before they are processed. If RAM and Swap space are exhausted, incoming logs are discarded.

5. Future Work

As a future work we concentrate both on improving performance and deploying other technologies. Currently, adding inotify support to imfile has high priority. Moreover, we plan to try different network protocols and filesystems. Further, we will create a second logging server to test the use of our SAN infrastructure instead of local RAID controller. This server could also be used for load balancing.

References

[1] Splunk [Online] "http://www.splunkbase.com/"
[2] Rsyslog [Online] "http://www.rsyslog.com/"
[3] OSSEC [Online] "http://www.ossec.net/"
[4] Windows Logging System [Online] "http://taosecurity.blogspot.com/2008/10/windows-syslog-agents-plus-splunk.html"
[5] FMC [Online] "https://lhcbweb.bo.infn.it/twiki/bin/view.cgi/LHCbBologna/FmcLinux"
[6] PVSS [Online] "http://www.pvss.com"
[7] Quattor [Online] "http://quattor.web.cern.ch/"
[8] E. Bonaccorsi and N. Neufeld *Monitoring the LHCb Experiment Computing Infrastructure with NAGIOS in Proc. of ICALEPCS 2009, 2009*