WEMAU001

ICALEPCS'11

13th International Conference on Accelerator and Large Experimental Physics Control Systems
October 10-14, 2011, WTC Grenoble, France

A Remote Tracing Facility For Distributed Systems

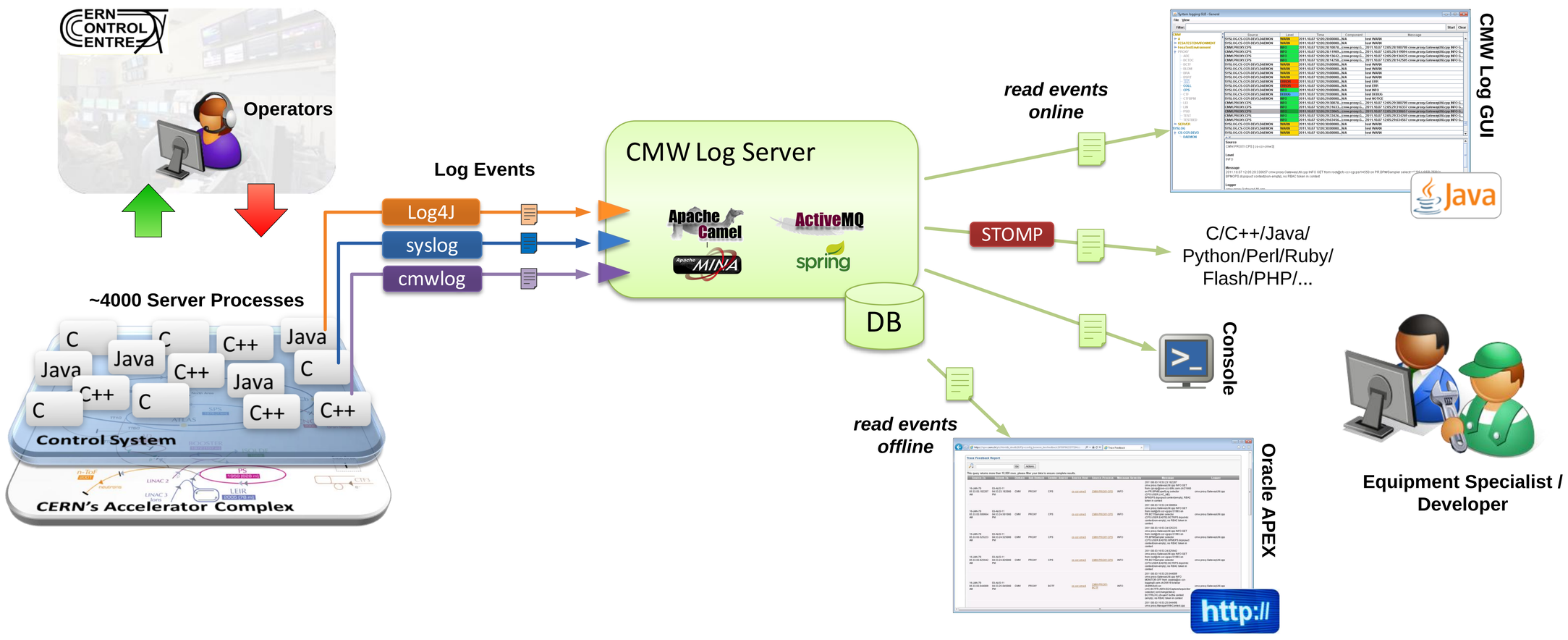
F. Ehm, A. Dworak, J. Lauener, CERN, Geneva, Switzerland

Abstract

Today CERN's Control System is built upon a large number of C++ and Java services producing log events. In such a largely distributed environment these log messages are essential for problem recognition and tracing. Tracing is vital for operation as understanding an issue in a subsystem requires analysing log events in an efficient and fast manner. At present 3'150 device servers are deployed on 1'600 diskless Front End Computers. The servers send log messages via network to an in-house developed central server which in turn saves them to file. Performance limitations and the lack of several highly desired features made the development of a new solution necessary. The new CMW Log Server fulfils these requirements by taking advantage of the Streaming Text Oriented Messaging Protocol (STOMP) and ActiveMQ as the transport layer. The system not only allows storing critical log events centrally in files or in a database but also allows other clients (e.g. graphical interfaces) to read the same events concurrently by using the provided Java API. Thanks to the ActiveMQ broker technology the system can easily be extended to clients implemented in other languages and is highly scalable in terms of performance. Long running tests have shown that the system can handle up to 10'000 messages/second.

Architecture

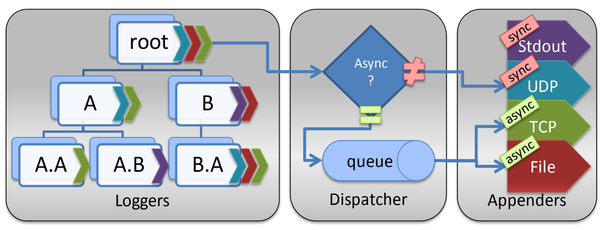
Collect and **unify** log events from **heterogeneous** sources



Log Sources

CMWLog – A Log Library for C++

- Implements Log4J concepts like **Appenders** and **Loggers**
- Support for **Windows**, **LynxOS** and **Linux**
- Thread safe and non-blocking calls
- Small memory footprint
- Depends on C++ standard libraries only
- Policies for handling I/O errors
- Logging to standard output, files, or to a remote host (TCP & UDP)
- Extendible to other output destinations



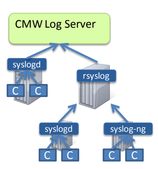
Architecture of the CMWLog C++ Library

Java

Send events using a standard log library Log4J or SLF4J.

C Sources

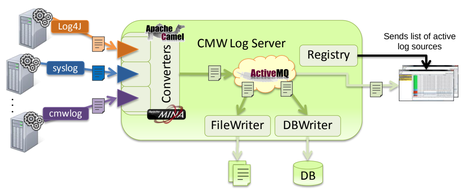
Send to local rsyslog, syslogd or syslog-ng which again forwards it to CMW Log Server.



Server Architecture

Key Features

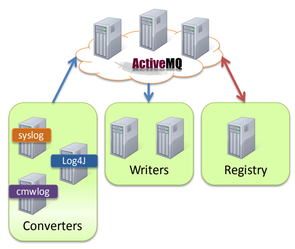
- Unification of log events from various sources
- Storage of data into a database or files
- Support for microseconds
- Automatic discovery of new log sources
- Blacklisting of log sources
- Easy to extend to new log protocols
- Allows for failover and load balancing
- Remote control and monitoring via JMX
- Large support for languages via STOMP (C/C++/Java/Python/Perl/Ruby/Flash/PHP/...)
- Designed to scale horizontally and vertically



Detailed Architecture of the CMW Log Server

Scaling the System

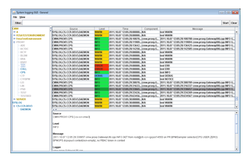
All components may run as one fully integrated process or as distributed standalone programs for scalability reasons.



Example for deploying server modules as separate processes

Presentation Layer

CMW Log Event Viewer



- Read log events **online**
- Filtering of events by keyword
- Shows **available log sources** in tree structure
- Severity level can be set **per log source**
- Enables recording of events into a local file

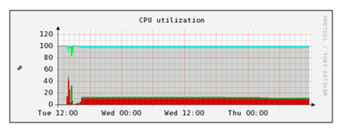
Other Readers

- Linux Console client subscribes via JMS API
- Oracle APEX web interface allows filtering with SQL
- Easy to add readers in other languages using STOMP

Performance

Performance Test

- 1'000 C clients publishing 500 Byte at 10'000 msg/sec
- 15 Java Clients subscribed to subset of data
- Resulting distribution load: **30'000 msg/sec**



CPU load over test run (48h) on CMW Log Server

Future

Feedback System

Investigations are ongoing to collect and store centrally server process data like:

- **deployment** information sent at installation time
  e.g. used libraries, version, host, path, etc.
- **configuration** information sent at startup time
  e.g. service type, port, monitoring capabilities, etc.

Supports the idea of automatic detection of newly deployed processes and central storage of their characteristics. This could for example simplify maintenance of a large infrastructure.

Conclusions

The new system fulfils all requirements and has been tested for performance and stability. It is designed to serve multiple reading applications for many equipment or service experts and enables the storage of data in a database or in size-based rotating files. Users have the possibility to read events online via a graphical interface which is integrated into existing operation and diagnostic tools. Through the Oracle APEX web interface data can be filtered and selected in a customized manner. Further, the flexible architecture allows to adapt to new log protocols and deployment models. First investigations have shown that the system is suitable for other fields of activity such as the configuration feedback for kernel modules running on Front End Computers.