

A Remote Tracing Facility for Distributed Systems

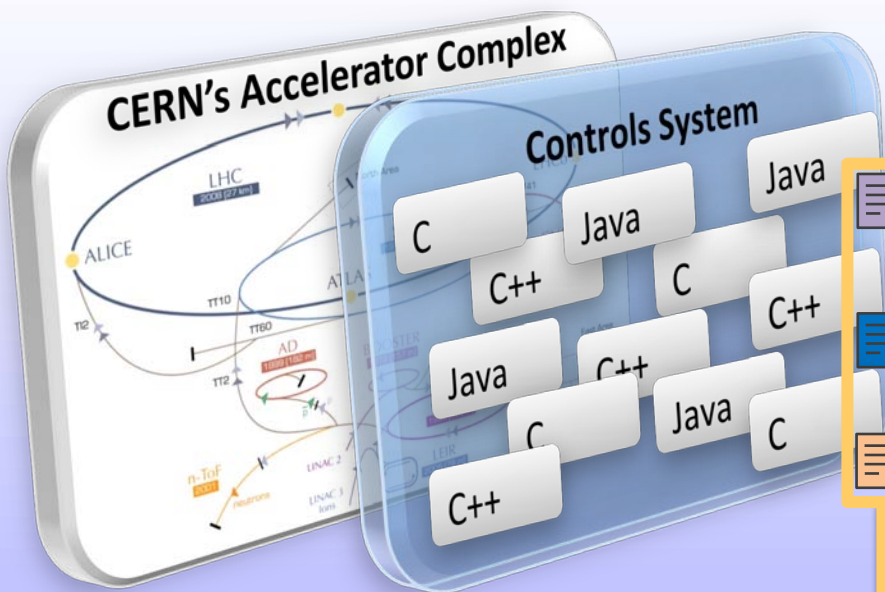
F. Ehm, A. Dworak, J. Lauener
CERN, Geneva, Switzerland

WEMAU001



Project Goal

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

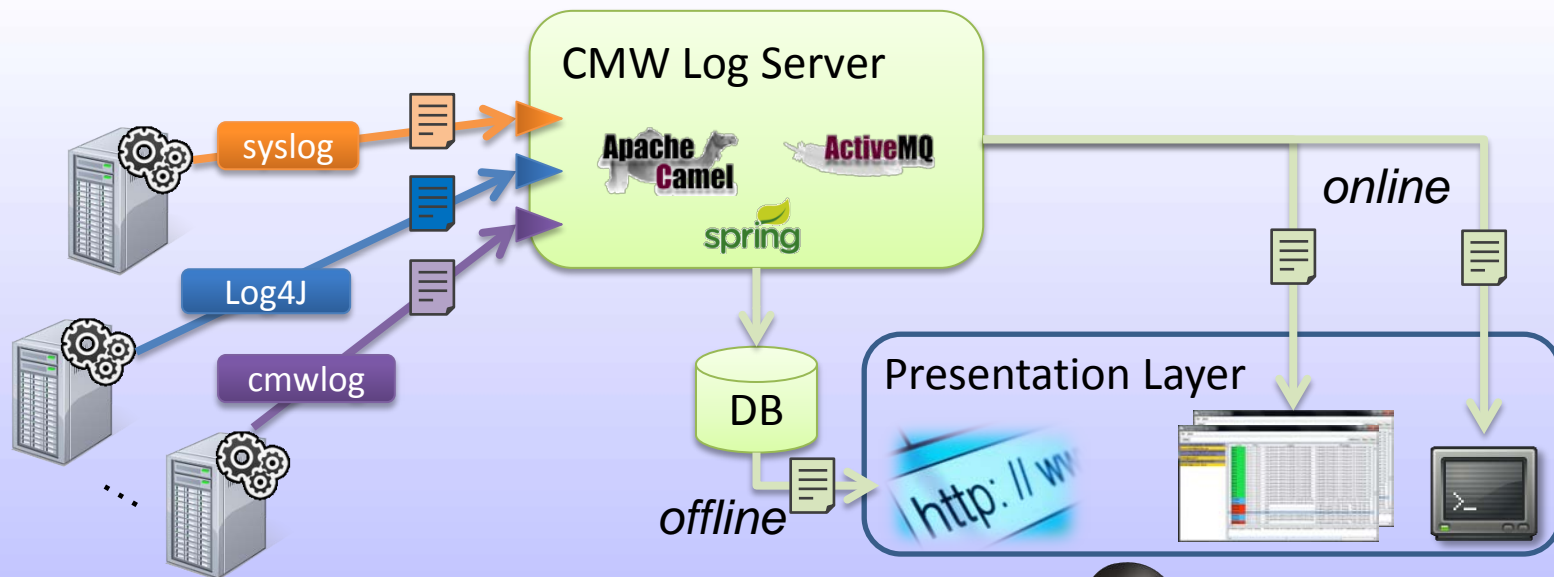


80.000 Hardware Devices

**4000 Programs
1700 Machines**



Solution Outline



4000 Server Processes
1700 Machines



Service Managers
Equipment Specialists

A Remote Tracing Facility for Distributed Systems

Thank you for your attention.

ICALLEPCS'11

32nd 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. Lauenner, 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 diagnosis and tracing. Tracing is vital for operations as understanding an issue or a subsystem requires analyzing log events in an efficient and fast manner. At present 3150 remote servers are deployed on 15000 desktop Front End Computers. The servers send log messages via network to an in-house developed central server which is built upon IBM's file. The performance limitations and the lack of several highly desired features made the development of a new solution necessary. The new CMW Log Server fulfills these requirements by using advantage of the Streaming Text Oriented Messaging Protocol (STOMP) and ApacheMQ as the transport layer. The system not only allows sending critical log events centrally in text or in a structured and precise manner, other clients (in graphical interfaces) to read the same events concurrently by using the provided Java API. Thanks to the ApacheMQ 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 10000 messages/seconds.

Architecture

Collect and unify log events from heterogeneous sources

WEMAU001

1. Floor
Kilimanjaro

<p>Log Sources</p> <p>CMWLog – A Log Library for C++</p> <ul style="list-style-type: none"> • Implements Log4j concepts like Appenders • Support for Windows, Linux/OS and LFS • Thread safe and non-blocking calls • Small memory footprint • Depends on C++ standard libraries only • Policies for handling ID errors • Logging to standard output, files, or to a remote host (TCP & UDP) • Extensible to other output destinations <p>Java</p> <p>Send events using a standard log library Log4j or SLF4J.</p> <p>C Sources</p> <p>Send to local syslog, syslogd or syslogng which again forwards it to CMW Log Server.</p>	<p>Server Architecture</p> <p>Key Features</p> <ul style="list-style-type: none"> • Flexible: log events from various sources • Supports log events from a wide range of languages • Supports log events from a wide range of protocols • Flexible control and monitoring via JMX • Large support for languages via STOMP (C/C++/Java/Python/Perl/PHP/...) <ul style="list-style-type: none"> -> Designed to scale horizontally and vertically <p>Scaling the System</p> <p>All components may run as one fully integrated process or as distributed standalone programs for scalability reasons.</p>	<p>Presentation Layer</p> <p>CMW Log Event Viewer</p> <ul style="list-style-type: none"> • Based on log events editor • Filtering events by keyword • Search in log events in real time • Can set per log source • Can filter log events into a local file <p>Other Readers</p> <ul style="list-style-type: none"> • Links Console client subscribers via JMS API • Checks APEX web interface allows filtering with SQL • Easy to add readers in other languages using STOMP <p>Performance Test</p> <ul style="list-style-type: none"> • 1500 C clients publishing 500 Bytes at 10000 msg/sec • 10 Java Clients subscribed to subset of data • Resulting distribution load: 30000 msg/sec <p>Future</p> <p>Feedback System</p> <p>Investigations are ongoing to collect and store centrally server process data like:</p> <ul style="list-style-type: none"> • deployment information sent at installation time • configuration information sent at startup time • e.g. loaded libraries, version, host, path, etc. <p>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.</p>
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Conclusions

The new system fulfills 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 distributed reading 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 extend by 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.