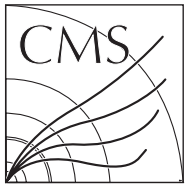
Available on CMS information server

CMS CR -2012/119



The Compact Muon Solenoid Experiment

Conference Report

Mailing address: CMS CERN, CH-1211 GENEVA 23, Switzerland

29 May 2012 (v2, 13 June 2012)

Alert Messaging in the CMS Distributed Workflow System

Zdenek Maxa for the CMS Collaboration

Abstract

WMAgent is the core component of the CMS workload management system. One of the features of this job managing platform is a configurable messaging system aimed at generating, distributing and processing alerts: short messages describing a given alert-worthy informational or pathological condition. Apart from the framework's sub-components running within the WMAgent instances, there is a stand-alone application collecting alerts from all WMAgent instances running across the CMS distributed computing environment. The alert framework has a versatile design that allows for receiving alert messages also from other CMS production applications, such as PhEDEx data transfer manager. We present implementation details of the system, including its python implementation using ZeroMQ, CouchDB message storage and future visions as well as operational experiences. Inter-operation with monitoring platforms such as Dashboard or Lemon is described.

Presented at *CHEP 2012: International Conference on Computing in High Energy and Nuclear Physics*

Alert Messaging in the CMS Distributed Workflow System

Zdenek Maxa[1]

[1] California Institute of Technology, Pasadena, California, USA

E-mail: zdenek.maxa@hep.caltech.edu

Abstract. WMAgent is the core component of the CMS workload management system. One of the features of this job managing platform is a configurable messaging system aimed at generating, distributing and processing alerts: short messages describing a given alert-worthy information or pathological condition. Apart from the framework's sub-components running within the WMAgent instances, there is a stand-alone application collecting alerts from all WMAgent instances running across the CMS distributed computing environment. The alert framework has a versatile design that allows for receiving alert messages also from other CMS production applications, such as PhEDEx data transfer manager. We present implementation details of the system, including its Python implementation using ZeroMQ, CouchDB message storage and future visions as well as operational experiences. Inter-operation with monitoring platforms such as Dashboard or Lemon is described.

1. Motivation

The efficiency of any large-scale distributed computing platform relies heavily on its ability to either respond to monitoring requests or to propagate information about its internal status to monitoring facilities. The presented Alerts framework is a messaging feature designed for the CERN LHC/CMS (Compact Muon Solenoid) [1] distributed Workload Manager - WMAgent [2], [13]. The main goal is to provide a messaging infrastructure for Alerts originating from so called alert-worthy situations or conditions. These situations are considered to have some adverse effect on an overall WMAgent job throughput and work efficiency needed to be known to Operators in a timely fashion and shall be instrumental during troubleshooting.

2. Introduction

The presented messaging framework consists from (going from the transport network layer upward):

- ZMQ [3] messaging library - TCP socket transport library with tens of language-specific bindings[1],
- Alert framework specific components of WMAgent,
- AlertCollector - central store and visualisation application for the Alerts framework[2].

[1] The implementation language of WMAgent is Python which determines the ZMQ library binding.

[2] The AlertCollector has been deployed within the CMS grid environment already. However, it only runs within the WMAgent instances and its migration behind CMS web, in order to become a central Alert collecting component, is the subject of the current development.

Like other projects within the scope of the CMS DMWM (Data Management Workload Management) [4] set of projects, of which WMAgent is one, there is a significant utilisation of the NoSQL CouchDB [5] database system. An implementation approach called Couchapp is used in the AlertCollector application which is discussed later. Couchapp [6] is a concept of executing an application, usually implemented in JavaScript, directly by the database server.

3. Alert framework design and implementation

Figure 1 displays the main blocks of the system hosted under a common WMAgent instance. The **AlertGenerator** and **AlertProcessor** components are central in the Alert framework implementation. Both of them are components[3] of WMAgent like other blocks in the two upper rows of the diagram.

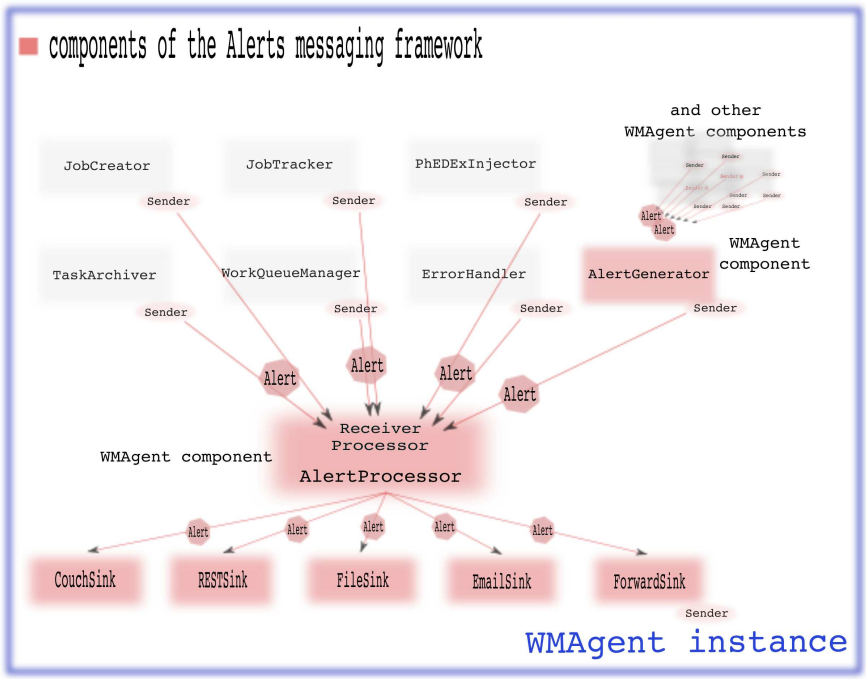


Figure 1. Overview of the Alert framework design within the WMAgent.

An **Alert** instance is derived from a Python dictionary containing information on the cause(s) of the alert or a JSON message [7] into which an Alert is converted before dispatching to network and later storing into various **Sinks**. In other words, an Alert instance can be present in the system in either format depending on the stage of the distribution chain.

3.1. Alert message example

As shown in the later example, an Alert defines **Level** parameter which carries the alert severity information. Generally, there are two types of alert severity - **soft** and **critical**[4]. Just like anything else about the Alerts system including Alert **Sinks** or **Pollers** (which are both discussed later), soft, as well as critical Alert severity association, can be configured in the WMAgent config file.

[3] WMAgent component means a daemon service out of which WMAgent consists.
[4] The philosophy is analogous to Unix soft, hard disk space quota.

An example of an Alert (JSON) encoded message[5]:


```
{... "Workload":"n/a", "Level":5, "TeamName":"mc",
    "Source":"ComponentsCPUPoller",
    "Details":
        {"numChildrenProcesses":0,"average":"76.27%",
        "component":"WorkQueueManager",
        "period":30, "threshold":"60%", "numMeasurements":3, "pid":18009},
    "Type":"WMAgent", "Timestamp":1328715746.41,
    "HostName":"vocms888.cern.ch",
    "Component":"AlertGenerator", "AgentName":"WMAgentCommissioning"}
```


Figure 2. Example of an Alert message.

Interpretation:

- Alert was generated by the **ComponentsCPUPoller**, a poller which periodically checks CPU utilisation of all daemon components of WMAgent,
- Alert severity 5 (depending on the particular WMAgent configuration, this can be a soft or critical alert message),
- WMAgent's component **WorkQueueManager** exceeded threshold **of 60% on CPU utilisation over a period of 30s**, during these 30s period, there were **3 measurements** resulting into **76.27% average value**,
- **WorkQueueManager** component runs under PID 18009 and this process has no child processes. Otherwise, they would also be polled for their CPU utilisation and evaluation would be done against their total sum.

The remaining items of the Alert message are self-explanatory.

3.2. AlertGenerator

A daemon component of WMAgent whose main duty is to maintain a set of configurable **Pollers**. A poller is an independent unit checking a particular metric of interest. From the implementation point of view, a poller is a **thread** which performs its check with a configurable frequency. The evaluation of a particular metric may be based on an one-off evaluation (i.e. an alert is generated immediately if the soft or the critical threshold is exceeded). Or the evaluation can be made over a given period of time - as in the above example. In this case an average value resulting from a configurable number of measurements over a configurable period of time is compared with soft and critical thresholds. The thresholds are usually defined as a percentage of some total (e.g. percentage of 100% CPU utilisation) or certain particular value (e.g. N GB of disk space for MySQL database).

3.3. Pollers

The list of currently implemented pollers and explanation of their function:

- CPUPoller - checks overall CPU utilisation of the machine *[percentage]*,
- MemoryPoller - checks overall memory utilisation of the machine, the aim is to trigger notification when the machine starts swapping *[percentage]*,

[5] Real Alert messages include also details like purpose and name of the WMAgent instance and operator's contact details.

- DiskSpacePoller - checks free disks space, if any of the partitions listed by the Unix command **df** exceeds a threshold, an Alert is generated *[percentage]*,
- ComponentsCPUPoller - checks per WMAgent's component CPU utilisation (uses ProcessCPUPoller) *[percentage]*,
- ComponentsMemoryPoller - checks per WMAgent's component memory utilisation (uses ProcessMemoryPoller) *[percentage]*,
- MySQLCPUPoller - checks MySQL database processes CPU utilisation (uses ProcessCPUPoller) *[percentage]*,
- MySQLMemoryPoller - checks MySQL database processes memory utilisation (uses ProcessMemoryPoller) *[percentage]*,
- MySQLDbSizePoller - checks size of the MySQL data directory *[gigabytes]*,
- CouchDbSizePoller - checks size of the CouchDB data directory *[gigabytes]*,
- CouchCPUPoller - checks CouchDB database processes CPU utilisation (uses ProcessCPUPoller) *[percentage]*,
- CouchMemoryPoller - checks CouchDB database proceses memory utilisation (uses ProcessMemoryPoller) *[percentage]*,
- CouchErrorsPoller - checks ocurrences of the CouchDB database HTTP code responses [8][6] *number of occurrences.*

The aforementioned ProcessCPUPoller and ProcessMemoryPoller take as input the base process PID number and perform checks also over all subprocesses so the resulting value of CPU utilisation and memory utilisation, respectively, is the relevant sum.

The modular design of the poller harness implementation allows for an easy implementation of other interesting metrics.

3.4. Other Alert sources

Besides AlertGenerator-maintained set of pollers, the other sources of Alert messages are various conditions in other components (usually exception handlers and alike).

For a WMAgent developer, it is very trivial to initialise **Alert Sender** by a single statement and then use the created **sendAlert()** method whenever there is a need to propagate information on an alert-worthy situation in the code flow.

3.5. AlertProcessor

This is another major block of the Alert framework which is also a daemon component of WMAgent. It consists of the following subcomponents:

- Receiver - Zero MQ [3] message receiver,
- Processor - maintains a set of configurable **Sinks**.

3.5.1. Sinks A sink is a target to which Alert messages received by the Receiver instance are eventually stored by the Processor instance. The logic of Processor is very simple: it checks whether an Alert according to its **Level** parameter is of soft or critical kind. Finally, an Alert is sent to all sinks which the AlertProcessor has been configured[7] to maintain. Practically, there is a pair for each kind of sink for soft and critical Alerts. The only difference is that soft sinks

[6] In a typical configuration this poller would watch over number of 500 status responses (internal application crash) or 404 status responses (document not found, non-existing URL). This poller takes an advantage of CouchDB exposing this kind of statistics on its **_stats** URL handler.

[7] In the WMAgent configuration file.

are buffered whereas critical Alerts are directed to critical sinks and sent off straight away.

The currently implemented sinks include:

- CouchSink - Alerts are stored into the WMAgent's local CouchDB instance[8],
- RESTSink - this has been intended as a generic REST client interface for a REST server. The CouchDB server features a REST interface as well and this sink basically is just another database within the local CouchDB instance. Its main point is later replication to a central CouchDB store - the AlertCollector,
- FileSink - a pair (soft, critical) of flat text files into which JSON-formatted Alert instances are stored,
- EmailSink - email account to which plain-text JSON-formatted Alert messages are sent,
- ForwardSink - implemented for the envisioned possibility to chain AlertProcessor components - to forward Alerts to another AlertProcessor. The feature is not used in the current production.

3.6. AlertCollector

This the last major component of the Alert framework. AlertCollector is a CouchDB database instance equipped with applications (couchapps) [6] for visualisation. It is foreseen to generate various statistics-based Alerts in the AlertCollector which will be forwarded to higher-level CERN/CMS computing monitoring facilities such as Lemon [9], SLS [10] or Dashboard [15].

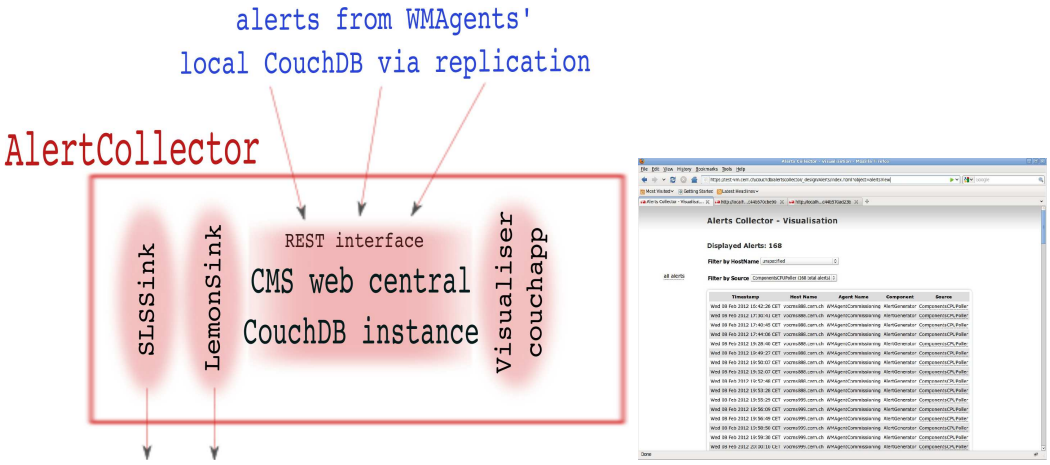


Figure 3. AlertCollector and its main sub-blocks including a screenshot of the Alert visualisation application.

Figure 3 shows a schema of the AlertCollector component. The visualisation application allows for browsing and filtering the content of the AlertCollector database in the web browser in user-friendly way.

The implementation of the AlertCollector (excluding the statistics-based Alert generation) is finished, but the deployment to the central CMS CouchDB database instance running on the CMS web server [11] is the subject of the current development effort.

The AlertCollector has been developed on the local WMAgent CouchDB instance and the main step in the CMS web migration requires to set up databases with continuous replication between the local and central instance.

[8] Each instance of WMAgent runs with both MySQL and CouchDB databases.

4. Conclusion and outlook

The described system is currently deployed on parts of the CMS data operations production in the grid environment running WMAgent workload managers. The grid site efficiency translates directly into WMAgents' efficiency and their job throughput. The Alerts system is expected to present significant contribution in terms of informing the Operators about efficiency degrading conditions. During the time of submitting the abstract for the CHEP 2012 conference, it was envisaged that the Alerts system would become an integral part of Operators' set of tools and sensors by the time of the conference. This level of adoption is yet to come.

Implementation-wise, the Alert framework is mostly independent on the WMAgent itself, although the two are very well integrated. As the framework is very versatile and configurable, it depends on the WMAgent Configuration store and both AlertGenerator and AlertProcessor components take advantage of a so called Harness which is a general system daemon wrapper. This means that the Alert system can be easily integrated into any larger-scale software project requiring internal Alert messaging facility. The Alerts framework has been developed with over 90% unittest coverage and is subject to builds on the CMS/DMWM Continuous Integration Jenkins [12] server.

The current development effort has been minimized into AlertCollector CMS web deployment and further implementation will be dependent on the degree of adoption of the Alerts framework by the CMS data operations team.

Future integration with other CMS/DMWM projects counts upon mainly RequestManager [13] and PhEDEx [14].

Acknowledgments

This work was supported by the US CMS Operations Program funded by the US Department of Energy.

References

[1] CERN LHC/CMS - Compact Muon Solenoid experiment `http://cms.cern.ch`
[2] CMS Workload Management platform - WMAgent `https://twiki.cern.ch/twiki/bin/view/CMS/WMAgent`
[3] ZMQ - Zero Message Queue messaging library `http://www.zeromq.org`
[4] CMS DMWM - Data Management and Workload Management `https://svnweb.cern.ch/trac/CMSDMWM`
[5] CouchDB - NoSQL document-based database `http://couchdb.apache.org`
[6] Couchapp - application within CouchDB database `http://couchapp.org/page/what-is-couchapp`
[7] JSON - JavaScript Object Notation `http://www.json.org`
[8] HTTP status response codes `http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html`
[9] Lemon - LHC Era Monitoring - CERN-IT monitoring system `http://lemon.cern.ch`
[10] SLS - Service Level Status - CERN-IT monitoring system `http://sls.cern.ch`
[11] CMS web, central CMS CouchDB instance `https://cmsweb.cern.ch/couchdb`
[12] Jenkins - open source Continuous Integration server `http://jenkins-ci.org`
[13] Stuart Wakefield - The CMS workload management system `CHEP 2012, New York, USA`
[14] From toolkit to framework the past and future evolution of PhEDEx, poster 188 `CHEP 2012, New York, USA`
[15] CERN Dashboard monitoring `http://dashboard.cern.ch`