Autonomous System Management for the ALICE High-Level-Trigger Cluster using the SysMES framework

Stefan Boettger, Timo Breitner, Udo Kebschull, Camilo Lara, Jochen Ulrich, Pierre Zelnicek for the ALICE HLT collaboration
Kirchhoff Institute for Physics, University of Heidelberg, 69120 Heidelberg, Germany
E-mail: {boettger, tbreitner, kebschull, lara, julrich, zelnicek}@kip.uni-heidelberg.de

Abstract. The ALICE HLT cluster is a heterogeneous computer cluster currently consisting of 200 nodes. This cluster is used for on-line processing of data produced by the ALICE detector during the next 10 or more years of operation. A major management challenge is to reduce the number of manual interventions in case of failures. Classical approaches like monitoring tools lack mechanisms to detect situations with multiple failure conditions and to automatically react to such situations. We have therefore developed SysMES (System Management for networked Embedded Systems and Clusters), a decentralized, fault tolerant, tool-set for autonomous management. It comprises a monitoring facility for detecting the working states of the distributed resources, a central interface for visualizing and managing the cluster environment and a rule system for coupling of the monitoring and management aspects. We have developed a formal language by which an administrator can define complex spatial and temporal conditions for failure states and according reactions. For the HLT we have defined a set of rules for known and recurring problem states such that SysMES takes care of most of day-to-day administrative work.

1. Introduction
Typical modern physics experiments produce a huge amount of data which has to be analyzed by computers. Data analysis can be realized in hardware (e.g. micro controllers, embedded systems) or software. Depending on the amount of data to be analyzed it is necessary to utilize large computer clusters to distribute the workload. An example of such an environment is the ALICE High Level Trigger (HLT [1]) Cluster at CERN, which is used for operating the HLT application. This application is in charge of receiving the data from the ALICE [2] experiment and performing on-line analysis. In the current commissioning state this cluster consist of about 200 nodes of three different hardware releases and 25 servers of four hardware releases. The expected number of nodes will increase in the next years up to 1000. The final number of hardware releases is unknown (currently there are six hardware releases in use) because the nodes will be bought at the last possible point in time in order to get a better cost-performance ratio. Another point which contributes to a computer cluster becoming heterogeneous is the duration of a physics experiment of that dimension (typically 10 years and more). Computers have a life-time of 3 to 5 years and therefore it is necessary to replace or update parts of the computer farm in order to achieve the computation power requirements. In order to guarantee
the successful operation of the cluster for the whole experiment duration a very high level of
system management services is required.

Basic system management for a computer cluster consists of monitoring of hardware and
software resources, analyzing monitoring data, and reacting to detected failures or errors. A
traditional system management strategy utilizes monitoring tools for retrieving and visualizing
device information. These monitoring tools are able to detect simple states and also to react
automatically. Furthermore, system administrators (i.e. humans) analyze this information and
take a decision whether an intervention is needed, i.e. initiate a manual reaction. Autonomous
system management extends these basic management methods in two aspects:

- Self-Configuration: Autonomous configuration of the management environment in order to
  manage the complexity of heterogeneous clusters.
- Self-Healing: Autonomous failure and error recognition for simple and complex states and
  also autonomous reaction for problem solution, i.e. reduction of human interventions in the
  cluster.

In addition, an autonomous system management framework has to fulfill two more requirements:
scalability and high availability. In order to fulfill all the requirements of autonomous system
management for large heterogeneous clusters the SysMES framework (System Management for
Networked Embedded System and Clusters [3]) has been developed.

2. SysMES Framework

The SysMES Framework is a scalable, decentralized, fault tolerant, dynamic, rule based tool set
for autonomous system management of networks of target systems.

2.1. Physical Architecture

The SysMES framework has been developed as a multi-layered architecture. There are three
layers: The operator layer, the management layer and the target layer. Further information
about the physical architecture can be found in [3]. The design decision for a multi-
layered architecture has been made in order to realize a vertical distribution of the workload.
A horizontal distribution is realized through the usage of clustered services for all offered
functionalities. In case of overload in a specific layer it is possible to extend the number of
members for dealing with increased workload. Members of a layer are also able to replace
each other. These strategies for the distribution of workload builds the basis to make SysMES
scalable and high available. The communication between the several layers of SysMES is realized
by exchanging XML representations of management objects. Those are Tasks for the top-down
communication and Events for the bottom-up communication.

2.2. Flexible Monitoring

Flexible monitoring describes the SysMES capability to monitor system resources locally on
the targets or remotely using common technologies and protocols such as Intelligent Platform
Management Interface (IPMI) and Simple Network Management Protocol (SNMP). Furthermore
the behavior of monitors can be changed dynamically by the reconfiguration of their properties
and redistribution to the targets.

2.3. Task Management

Task Management is the SysMES functionality for the distribution of management objects (e.g.
rules, monitors and actions) to desired targets. There are two types of tasks, configuration and
administrative tasks. The first one is used for setting up, reconfiguring and deleting management
objects and the second one for executing actions (i.e. binaries, scripts). Task execution results
are propagated through events.
2.4. Rule Management

Rule Management combines the aforementioned monitoring and task distribution features into a loop of control. Monitoring data is constantly evaluated against a set of rules; successful matching of a rule leads to the deployment of tasks. The specification of a rule follows the grammar given in [3]. It is, roughly speaking, an IF-THEN clause composed of a left-side describing a condition and a right-side describing tasks to be executed. A Rete [4] based algorithm, extended with temporal features, is used for evaluating specified rules. Rules can be evaluated both on server and on target layer. The choice for their locality depends on the syntactical complexity of the rules and the thereby imposed computational effort for their evaluation. Target-side (simple) rules do not correlate monitoring data originated by different monitors. This makes them the right choice for fast, local error recognition and resolution. Server-side (complex) rules offer the whole range of specification semantics and allow correlation of monitoring data in a spatial and temporal manner. Such global states and complex error conditions can be specified and reacted to. More information can be found in [5].

3. Autonomous Management

Concerning the SysMES framework, the term autonomous management means that SysMES assumes the role of a system administrator in order to reduce error recognition time and manual interventions in the managed environment. This is realized by the automatic configuration of the management targets with the necessary management resources, i.e. monitors and rules (self-configuration), as well as the automatic recognition of errors and their solution (self-healing).

3.1. Self-Configuration

The SysMES self-configuration feature allows handling the complexity of deploying management resources in a heterogeneous environment. The heterogeneity of the environment raises the need to deploy different management resources to different physical nodes. Such a heterogeneity can be heterogeneity of hardware as well as heterogeneity of purpose: for example different models of hard discs may need different monitors or different actions on different conditions. But also for the same hard disc models a different purpose (such as serving as storage for a production system or storing temporary and log files on a development machine) induces the need to have different management resources deployed. Because manual configuration in such a heterogeneous environment would cause an enormous effort of individual management actions and decisions, self-configuration is a feature of substantive value. Self-configuration of SysMES is made up of two sub-functionalities:

- The ability to decide which management resources are utilized for which system resources. Such a decision defines a target state for the deployment of management resources.
- The ability to remedy differences between this intended state and the actual state of the deployment. This includes the creation of initial configuration tasks as well as correcting the actual state of the management resource deployment in the case a deployment has failed or a management resource ceased to work.

The self-configuration feature is based on two object-oriented CIM [6] based models: the model of system resources (also called inventory model) and the model of management resources [7]. Relationships between both models are expressed using associations. The associations define pairs of (device,management resource), e.g. (HD_type1, CheckHDHealthMonitor) or (afsd, CheckAFSServerDaemonStatusMonitor). The SysMES client informs the server periodically about the actual system resources (hardware and software) of the node and the self-configuration uses the associations between the system resources and the management resources to determine which management objects have to be deployed.
3.2. Self-Healing
The SysMES Framework facilitates self-healing capabilities such that the time spent on manual interventions in case of failures is reduced. This is done by a) automated detection of complex error situations which otherwise would need manual investigation and correlation, b) automated execution of repair procedures, and c) reporting of errors via mail and text message as last resort. Rules capture domain-specific knowledge of administrators and allow for automated recognition of correlated error occurrences. These errors are tried to be corrected via the task execution of successfully matched rules. However, not all errors are correctable and for this purpose rules can be concatenated, leading to an automated multi-level error resolution strategy.

4. Proof-Of-Concept: ALICE High Level Trigger Cluster
This cluster is used for running the High Level Trigger (HLT) application. The HLT is a hierarchical and distributed application used for the on-line analysis of the data arising from the ALICE detectors. Processing of the collision data consist of filtering and selecting the relevant parts of the data (e.g. accept or reject it) and also compressing selected data for further processing steps. The HLT incoming data rate is about 25 - 30 GByte/s and the maximum outgoing data rate is 1.25 GByte/s. Detailed information about the HLT can be found in [1]. In order to process this very high amount of information a large distributed computer cluster is needed.

4.1. Infrastructure
In the actual commissioning state the ALICE HLT cluster is a heterogeneous cluster composed of about 25 servers and 200 nodes. Those nodes are divided into two categories: front-end processors (FEP) and computing nodes (CN). Each FEP node is equipped with two HLT ReadOut Receiver Cards (H-RORC) which are used for the injection of the ALICE detector data into the cluster and also for the realization of the first processing steps. Each computer in the cluster is equipped with a remote access and management card/chip. At the moment there are three different solutions in use: The Computer Health and Remote Management card (CHARM), the Winbond® WPCM450 BMC and the Tyan® M3291 SMDC add-on card. All three solutions offer similar remote access capabilities such as display, keyboard, and mouse exportation and also interfaces for reading out sensor values such as temperatures. The CHARM card can be accessed using the local operating system and the other solutions utilize IPMI. The nodes currently show a certain heterogeneity in terms of CPUs, using dual and quad core CPUs as well as different AMD Opteron™ and Intel® Xeon® CPUs. For other hardware components such as motherboards, hard disks, memory, etc. one can observe a similar variety. Nodes are interconnected using a structured gigabit ethernet network as well as QDR InfiniBand.

The current state of the cluster represents the required computer power for the first year of the ALICE experiment. In the next years it is planned to extend the cluster up to about 1000 nodes. It is also planned to utilize other computation devices such as Graphical Processing Unit (GPU) and Field Programmable Gate Array (FPGA) for the acceleration of the data processing.

4.2. SysMES Installation for the HLT Cluster
The SysMES framework is currently used for managing the components of the HLT cluster as well as the HLT application. The main task of SysMES is the recognition of simple or complex states and the unattended reporting and solution of recognized errors. The actual management environment is composed of two SysMES servers and two database servers. Each computer on the cluster is equipped with a SysMES client for the execution of management tasks such as monitoring, rule checking and reacting. Furthermore the SysMES framework provides interfaces for retrieving relevant information from other monitoring systems such as LHC Era Monitoring (LEMON [8]).
The prerequisite for self-healing is a monitoring strategy which provides information about the state of single nodes or applications. Monitoring the HLT cluster is divided into the following 3 aspects:

- **Hardware monitoring**: monitoring of computer components by accessing the information through the remote management card (e.g. temperature, fan RPM, postcodes, CMOS status, etc) or directly using the operating system. Monitoring of the H-RORC cards (e.g. revision ID, Detector-Data-Link state, etc.). Monitoring of other hardware components such as racks and switches via SNMP or IPMI (e.g. network traffic, errors, collisions, rack voltage, humidity, etc.).

- **Operating system (OS) monitoring**: monitoring of operating system resources (e.g. log files, configuration files, etc.). Monitoring of system resources via OS (e.g. CPU/Memory utilization, Hard Disk/Raid Health, etc.). Monitoring of server applications (e.g. AFS status, Quotas, LDAP, etc.). Monitoring of client daemons (AFS, NFS, CRON, etc.).

- **Application monitoring**: Monitoring of the HLT (e.g. event rate, buffer utilization, etc.).

### 4.3. Self-Healing Use Cases

The self-healing strategy for the ALICE HLT cluster is realized by the development of rules for unattended reaction to undesired states and also for reporting those to the system administrators. The self-healing strategy involves client simple rules, server simple rules and complex rules. Client simple rules are especially required for the definition of emergency strategies, e.g. client-side task execution without server interaction in case of network or connectivity failures. Another important usage of these rules are cases where a fast reaction has to be performed in order to prevent damages. An example would be a rule for evaluation of events for a critical hard disk temperature (\( HD_{\text{temp}} \geq 70 \)). The client reacts to this state by shutting down the node. However, if the network is available, this state will also be propagated to the server.

One typical use case of server simple rules is sending mails and text messages containing the event data. Some of the nodes do not have the rights or capabilities to send those and therefore the SysMES server hosts this reporting rules and sends messages to the administrators with the relevant information. Complex rules are used, among other things, for the correlation of multiple events. In some cases it is not necessary to react to single events immediately, e.g. in case of a temporary increase of the CPU load. The monitor of this system resource sends events periodically (e.g. each 59 seconds). Multiple occurrences of events of the same monitor and target within a specified time interval will be recognized as a major problem and a new event with a higher severity will be created.

Furthermore, complex rules can be used for escalating problems if simple rule solutions do not work out. In case of failure of a system daemon (event "daemon = down") there is a simple rule with one action for trying to restart the daemon. Additionally, there is a complex rule with two actions: one for the escalation of the problem to a higher severity, which means that the simple rule was not able to solve the problem, and one for informing an administrator via mail. To point out further usages of rules in the area of autonomous system management we present the following use case.

The use case concerns the staged and unattended shutdown of the servers in case of a power outage. At the moment the racks where the infrastructure servers and network switches are located are connected to an UPS equipment which guarantees power availability and stability for a period of 15 minutes. The strategy for avoiding damages in the server and network infrastructure consists of monitoring the voltage values of the electricity, detecting power supply interruptions, waiting a certain period and shutting down the servers in a staged way. A SysMES monitor reads out the actual voltage values of the RMS via SNMP and generates a high severity
event if the lowest measured voltage value of the three-phase electricity is less than 200 Volt. The complex rule will be activated by the first occurrence of one PowerCut event. The complex rule fires at the occurrence of three PowerCut events from two different servers within a time interval of 150 seconds. The number of required events is part of the complex rule configuration and depends on the desired grace period before the shutdown procedure has to be performed (i.e. for avoiding a shutdown due to a simple power glitch). Afterwards the staged shutdown of the servers is initialized. This is required in order to ensure that file servers go down last. Otherwise the central storage could end up with inconsistencies.

4.4. Performance
The SysMES framework has been tested extensively during the last ALICE shutdown time at the end of 2009. This test has been realized in the ALICE HLT cluster, i.e. with real servers and nodes. The first scalability tests demonstrate that the framework is able to process about 2500 monitor values per second (including their storage, simple rule matching and execution of a server action) with one server and about 4000 per second with 2 clustered server instances. Assuming that all monitors on all controlled nodes fire constantly (worst-case), and assuming 30 monitors with an average period of 10 seconds, this setup would already cover at least 1000 nodes:

$$\frac{\text{nodes} \cdot \text{monitors}}{\text{period}} = \frac{1000 \cdot 30}{10s} = 3000s^{-1}$$

However, the experienced rate of monitor values during normal operation is in the order of a few values per minute.

Concerning the fault tolerance several tests have been performed simulating a server crash. The targets connected to the failed server were able to reconnect to another server instance and to send desired events and execute tasks. More results can be found in [3].

5. Conclusion
The management of large heterogeneous clusters needs sophisticated tools to handle the complexity of such systems to reduce manpower needed for administration. We have therefore developed SysMES for the autonomous management of the HLT, featuring the core capabilities self-configuration and self-healing. SysMES implements these capabilities in form of a powerful rule system and an extensible model for management resources. A set of specific management objects and rules has been developed for the HLT and has been used extensively and successfully to manage the cluster.

References
[1] Alt T et al. (ALICE) 2004 J. Phys. G30 S1097–S1100
[2] Aamodt K et al. (ALICE) 2008 JINST 3 S08002
[3] Lara C 2011 The SysMES Framework: System Management for Networked Embedded Systems and Clusters Ph.D. thesis Ruperto Carola University of Heidelberg
[4] Forgy C 1979 On the efficient implementation of production systems Ph.D. thesis Carnegie-Mellon University
[5] Boettger S 2006 Distributed Composite Event Monitoring Master’s thesis University of Leipzig
[6] CIM: Common Information Model http://www.dmtf.org/standards/cim/
[7] Zelnicek P 2005 Object-oriented Environment Modelling for Rule Based Event Management Master’s thesis University of Leipzig
[8] LEMON: LHC Era Monitoring http://lemon.web.cern.ch/lemon/index.shtml