Automated Inventory and Monitoring of the ALICE HLT Cluster Resources with the SysMES Framework

J. Ulrich¹, C. Lara¹, Ø. Haaland², S. Böttger¹, D. Röhrich², U. Kebschull¹ for the ALICE HLT Collaboration

¹Infrastructure and Computer Systems for Data Processing (IRI), Institute for Computer Science, Goethe-University Frankfurt/Main, Germany
E-mail: {j.ulrich, lara, boettger, kebschull}@iri.uni-frankfurt.de

²University of Bergen, Department of Physics and Technology, Bergen, Norway
E-mail: {Oystein.Haaland, Dieter.Rohrich}@ift.uib.no

Abstract. The High-Level-Trigger (HLT) cluster of the ALICE experiment is a computer cluster with about 200 nodes and 20 infrastructure machines. In its current state, the cluster consists of nearly 10 different configurations of nodes in terms of installed hardware, software and network structure. In such a heterogeneous environment with a distributed application, information about the actual configuration of the nodes is needed to automatically distribute and adjust the application accordingly. An inventory database provides a unified interface to such information. To be useful, the data in the inventory has to be up to date, complete and consistent. Manual maintenance of such databases is error-prone and data tends to become outdated. The inventory module of the ALICE HLT cluster overcomes these drawbacks by automatically updating the actual state periodically and, in contrast to existing solutions, it allows the definition of a target state for each node. A target state can simply be a fully operational state, i.e. a state without malfunctions, or a dedicated configuration of the node. The target state is then compared to the actual state to detect deviations and malfunctions which could induce severe problems when running the application. The inventory module of the ALICE HLT cluster has been integrated into the monitoring and management framework SysMES in order to use existing functionality like transactionality and monitoring infrastructure. Additionally, SysMES allows to solve detected problems automatically via its rulesystem. To describe the heterogeneous environment with all its specifics, like custom hardware, the inventory module uses an object-oriented model which is based on the Common Information Model. The inventory module provides an automatically updated actual state of the cluster, detects discrepancies between the actual and the target state and is able to solve detected problems automatically. This contribution presents the current implementation state of the inventory module as well as the concept for future development.

1. Introduction

In heterogeneous computer clusters like the ALICE High-Level-Trigger (HLT) cluster [1], the configuration of the nodes differ in terms of installed hardware, software and network structure. Therefore, it is necessary to look up the configuration to perform tasks relying on that information. Examples for such tasks include placing components of a distributed application on the cluster nodes and resolving configuration related problems like hardware failures. Inventory databases are a solution to provide that information in a unified, centralized manner. However,
in manually maintained inventory databases, data tends to become outdated, incomplete and mistakes slip in easily which makes the data inconsistent. To overcome these drawbacks, the inventory solution should automatically gather and update the data in regular intervals. This ensures that the data is up to date, complete and consistent with the real world.

Most existing inventory solutions are limited in their customization capabilities and it is difficult to include custom hardware like the HLT Read-Out Receiver Card (H-RORC) which is used in the ALICE HLT cluster [1, section 6.3.2.2]. A detailed review of many existing inventory solutions can be found in [2]. The presented inventory module has been developed to provide a highly customizable, automated inventory solution that fulfills all requirements for the ALICE HLT cluster and other heterogeneous computer clusters with custom hardware. These requirements include the automatic detection of objects in the cluster, the automatic updating of the data in the inventory and the flexibility to support heterogeneity and customizations (see [2, chapter 2] for the full list of requirements).

2. Concept
The SysMES inventory module is integrated into the system monitoring and management solution SysMES [3]. This way, the inventory module is able to use existing infrastructure like the clients-server based architecture and databases. The inventory module gathers the desired configuration information using SysMES monitors. These monitors are run in configurable periods, read the information from the nodes and send it back to the SysMES server. On the server side, the inventory module retrieves the events, processes them and stores the information in a relational database. Storing the data in a relational database allows other applications to access and use the inventory information.

2.1. Model and Mapping
The basis of the inventory module is a model of the environment. This model is based on a Common Information Model (CIM) schema [4]. A CIM schema describes which types of objects (classes) exist in the environment, what their properties are and how they can be related to each other (associations). A schema is written in the Managed Object Format (MOF) as described in [5]. Existing CIM schemas already provide the most commonly used classes like ComputerSystem, Processor, etc. and it is also possible to modify and extend a schema to fit the specifics of the environment like custom hardware.

In addition to the model, the inventory module requires the definition of a mapping between the properties in the real world and the properties of the classes in the model. This mapping tells the inventory which properties from the real world should be included in the inventory, how to collect their values and how they are stored in the model. This mapping is defined by so called InventorySettings objects. Each InventorySettings object is linked to one class in the model and is responsible for detecting objects in the real world that should be mapped to that class (called instances), as well as the associations between the instances. For that purpose, the administrator specifies two programs (typically scripts) in the InventorySettings object, one used to detect the instances of the class and one used to detect the associations. Additionally, so called InventoryProperty objects are defined. Each InventoryProperty object is linked to one property of the class of an InventorySettings object. In the InventoryProperty object, the administrator defines the period in which the value of the property is updated and he provides the program that reads the value of the property from the machine. Being able to change all involved programs/scripts enables the inclusion of any custom hardware or software in the inventory.

2.2. Workflow
The workflow of the inventory module can be seperated into two parts: the first part is the configuration workflow, which is performed by the administrator, and the second part is the
Figure 1. Workflow of the SysMES Inventory Module. The upper part shows the tasks of the configuration workflow. The lower part shows the operation workflow.

operation workflow, which is performed by the inventory module automatically.

The configuration workflow is highlighted in the upper part of figure 1 and is primarily concerned with the creation of InventorySettings and InventoryProperty objects. The administrator creates these objects and then uses the objects to generate so called discovery monitors which will be explained in the next paragraph. Both steps are performed through a graphical user interface (GUI) as shown in figure 2.

The operation workflow is highlighted in the lower part of figure 1 and is concerned with the collection, processing and storage of the data. The collection is performed by two mechanisms: the discovery mechanism detects the instances and their associations using the programs defined in the InventorySettings objects. The update mechanism performs the updating of the properties of the instances using the programs defined in the InventoryProperty objects. Both mechanisms are realized through SysMES monitors which are running periodically to detect appearance/disappearance of instances and changes in the properties respectively.

The administrator initiates the operation workflow by deploying the created discovery monitors on the machines that should be included in the inventory. The discovery monitors detect the existing instances and their associations and sends that information to the inventory module on the SysMES server. Depending on that information, the inventory module automatically generates the necessary update monitors and deploys them to the nodes. The update and discovery monitors stay deployed on the nodes and the update monitors will only be removed/redeployed when (all) instances on the machine disappear/appear. The update monitors read the properties’ values of the detected instances from the nodes and send these values to the inventory module. The inventory module stores the information about the instances and their properties in the database. The SysMES inventory GUI, as well as other applications, are then able to use and display the information from the model database.
2.3. Target State

The target state represents the state the environment should be in and the definition may vary from environment to environment. For example, in one environment it might be enough if the target state includes the currently installed hardware and software of the machines. This would mean that the target state represents a fully operational state where all the installed hardware and software works properly without malfunctions. However, in another environment, the machines might have dedicated configurations like specific server daemons running on the machine, and the target state may include this dedicated configuration. Discrepancies between the actual and target state would then exist even if all installed hardware and software is working properly but the configuration of the machine is not correct, e.g. because one required daemon was not installed.

Both, the target and actual state, use the same underlying CIM model to allow comparing them to detect discrepancies. The comparison is performed by finding matching instances and associations in both states using the properties of the classes which uniquely identify instances. However, in the actual state, a property can only have one value which is the current value of the property. In the target state, on the other hand, it is reasonable to allow multiple possible values or a range of values for a property. For example, the actual state could contain a partition of a file system which is formatted using the “ext3” file system. The target state, on the hand, may allow that this partition is formatted with either “ext3” or “ext4”.

The target state is maintained by the administrator who decides which changes in the actual state are intended, i.e. should be adopted by the target state. For example, if a memory DIMM
disappears on a machine, this could mean that the DIMM has a defect and has to be replaced. In such a case, the instance of that DIMM must be kept in the target state to ensure that this defect is visible in the comparison between actual and target state as long as it exists. On the other hand, the disappearance of a DIMM could mean that it has been removed by an administrator because it is not needed any more in that machine. In such a case, the instance of the DIMM should be removed from the target state since this was an intended change which should not be marked as a discrepancy between the actual and the target state.

3. Current State of Implementation

As already mentioned, the inventory module is integrated in the SysMES framework. Therefore, it is written as a Java Enterprise Edition (Java EE) application running on a JBoss Application Server.

Currently, the actual state is fully implemented, i.e. the SysMES inventory module is able to automatically collect and update the instances and their properties. The instances are stored in a relational database using the CIM model and object-relational mapping as described in [6, 7]. The instances and their properties can be viewed through the SysMES web GUI which also allows creating and editing the InventorySettings and InventoryProperties objects. Screenshots of the GUI are shown in figures 2, 3 and 4.

In the current implementation, the monitors that perform the discovery and update send a message every time they run. To remove unnecessary network traffic and message processing, it is planned to change this behavior by saving the last value of the discovery and update on the client and send a message only when the value changed.

The implementation of the target state is still in work. The remaining tasks are the extension of the GUI to be able to view and edit the target state as well as visualizing the discrepancies between the actual and the target state, and the implementation of an algorithm to detect the discrepancies between both states.

4. Conclusions, Results & Outlook

The SysMES inventory module provides an automated, highly customizable inventory solution that fulfills the needs of heterogeneous computer clusters with custom hardware.

Functionality and performance tests in a test cluster have proven that the SysMES inventory module is able to build an inventory of the actual state of a cluster of 60 nodes with 23 properties...
Figure 4. Screenshots of the GUI showing the properties of an instance of the class CIM_Processor (on the left) and the associated objects of an instance of the class CIM_ComputerSystem (on the right) in about 4 minutes and updating 20 properties of 100 nodes, i.e. 2000 properties in total, takes about 72 s [2, section 7.4].

The plan for the future is to finish the implementation of the target state and reduce the network traffic as indicated in section 3. Then, the inventory module will be thoroughly tested to ensure that it has no negative effects on the application running in the cluster. After that, the inventory module is ready to be installed on the production partition of the HLT cluster.

5. Literature
[1] ALICE Collaboration T 2008 *Journal of Instrumentation* 3 S08002 URL http://stacks.iop.org/1748-0221/3/i=08/a=S08002
[2] Ulrich J 2011 *The Inventory Module of the SysMES Framework* Diploma thesis Kirchhoff Institute for Physics, Heidelberg University, Germany
[3] Boettger S, Breitner T, Kelschull U, Lara C, Ulrich J, Zehnek P and the ALICE HLT Collaboration 2011 *Journal of Physics: Conference Series* 331 052003 URL http://stacks.iop.org/1742-6596/331/i=5/a=052003
[4] CIM Website URL http://www.dmtf.org/standards/cim
[5] Distributed Management Task Force *CIM Infrastructure Specification Version 2.5.0* URL http://www.dmtf.org/sites/default/files/standards/documents/DSP0004_2.5.0.pdf
[6] Haaland O, Hermann M, Ulrich J, Lara C, Rohrich D and Kelschull U 2011 *5th International DMTF Academic Alliance Workshop on Systems and Virtualization Management (SVM)* pp 1–8 DOI 10.1109/SVM.2011.6086466
[7] Hermann M 2010 *Object-Relational Mapping for the Common Information Model* Diploma thesis Kirchhoff Institute for Physics, Heidelberg University, Germany