Monitoring VMware-based Virtual Infrastructures with OpenNMS

Christian Pape[1]*, Ronny Trommer*[2]

Abstract

Today's IT infrastructure is more and more being virtualized. Almost every kind of component like servers, storage and network devices can be virtualized to provide a high available, scalable and sustainable environment. Server consolidation - replacing many small servers by one large physical machine - leads to an increased utilization of hardware resources, decreased power consumption for operation and cooling. Virtualization also changed the way of day-to-day operations of IT administrators. The deployment of virtual machines is an easier and simpler process than installing new physical hardware and the possibilities of moving virtual machines between host systems reduces maintenance windows and downtimes. The virtualization of hardware also introduced new ways of monitoring and measuring performance of virtualized IT services. The integration of these features in network management and monitoring systems is needed for a secure and proactive operation of virtualized infrastructures. This paper presents the idea and development process of integrating *VMware vSphere* virtual infrastructures in an open source enterprise-grade network management system called *OpenNMS*.

Keywords: network monitoring, virtualization, virtual machines, datacenter management

1 Introduction

Virtualization offers a lot of new possibilities and features for modern IT infrastructures. Scalability and high availability of services can be achieved with less but higher utilized hardware. Less power is needed for operation and cooling of data centers and management procedures can be highly centralized. But these technologies also introduce challenges for traditional monitoring systems[1, 5, 6]. To fully benefit of virtualization technologies it is required that network management systems are able to retrieve management and monitoring data such as performance indicators from virtual infrastructures. In this paper we discuss the development process of integrating *VMware*-based infrastructures in the open source network management system *OpenNMS*. Following this introduction we will give a brief architectural overview of *OpenNMS* and *VMware vSphere*. In section 2 we introduce the basic frameworks and technologies we used for our implementation and in section 3 more details on our implemented modules. The following section 4 describes the requirements for the *VMware vSphere vCenter Server* and the basic steps for configuring our modules. Section 5 summarizes the results of our work and finally, we discuss the results and relate our conclusions and possibilities for additional work in section 6.

1.1 OpenNMS architecture

Integration of more complex monitoring solution in open source software requires a modular architecture and a solid stack of API's. *OpenNMS* provides a Java-based open source platform. The architecture is a modular and event-

[1] University Of Applied Sciences Fulda, Marquardstr. 35, 36039 Fulda, Germany

E-Mail: Christian.Pape@informatik.hs-fulda.de

[2] University Of Applied Sciences Fulda, Marquardstr. 35, 36039 Fulda, Germany

E-Mail: Ronny.Trommer@informatik.hs-fulda.de

driven approach. Development and design of *OpenNMS* follows the *FCAPS* principle. It defines the standard functionalities of network management applications and covers several administrative entities:

Fault management

Configuration management

Accounting management

Performance management

Security management

The current set of daemons supports fault-, configuration- and performance management. For realizing different use cases in network management, *OpenNMS* contains a set of daemons for specific use-cases in network management.

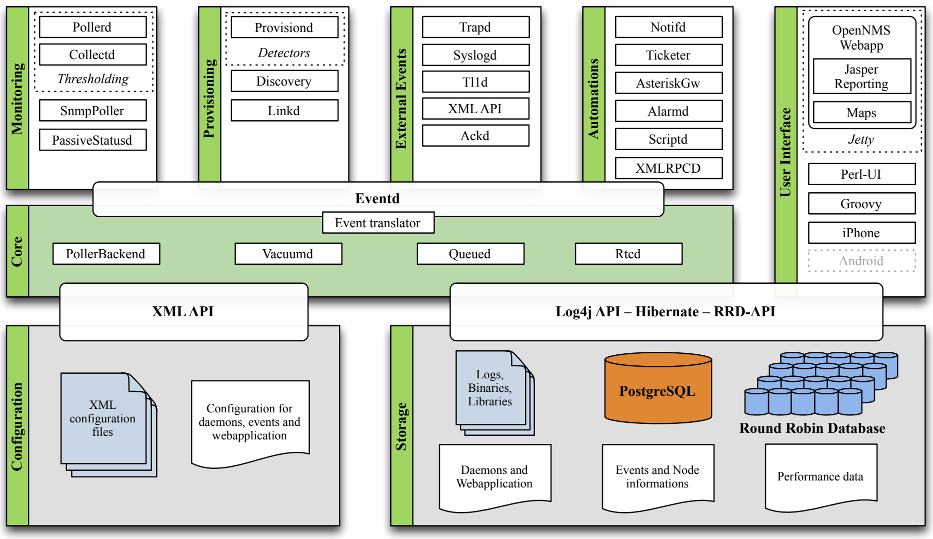

Figure 1: The OpenNMS architecture

To implement modules for monitoring *VMware vSphere* environments we have to extend the following parts of *OpenNMS*.

1.1.1 Pollerd – Service assurance

Pollerd is a Java daemon, which executes monitors to test service availability and provides also failure management functionalities. Each monitor is executed in a thread pool. The monitor provides the necessary metrics to calculate and measure the availability of a service. For this purpose, a monitor represents just two states – up and down. It is possible to extend *Pollerd* with customized monitors. It provides a Java API we can use for *VMware vSphere* specific use cases. We can also use this service for monitoring the hardware health status of a host system.

1.1.2 Collectd – Collecting performance metrics

Collectd is a Java daemon, which collects performance metrics. These metrics are used to measure utilization or latency of devices and services and provides abilities for performance management. To be able to collect performance metrics from virtual environment infrastructures we have to extend *Collectd's* Java API with *VMware* specific features. The performance time series data is stored in the open source *Round Robin Database* provided by *RRD-Tool*.

1.1.3 Provisiond – Provisioning of nodes in network management

It is important to define a workflow, which allows network administrators to bring physical and virtual network nodes into a network management system. With *Provisiond OpenNMS* provides a daemon to synchronize external data sources containing network node information with the *OpenNMS'* internal network management database. Extending *Provisiond* allows to automatically synchronizing node information between *VMware vSphere vCenter* and *OpenNMS'* internal network management database. In this paper the *VMware vSphere vCenter* is used as the data source for the node information. *OpenNMS* will pull this node information and updates its internal network management database. This process follows a preconfigured schedule.

1.2 VMware vSphere architecture

The *VMware vSphere* architecture focuses on enabling IT organizations to deliver flexible, scalable and reliable IT services. All kinds of components like CPUs, storage and network devices are virtualized to provide a highly dynamic operating environment. In this section we give an overview of the architectural entities important for the integration into network management systems. First of all physical server hardware needs an abstraction layer in form of a hypervisor operating system - the *VMware ESX* and *VMware ESXi* products are providing this. When installed on a physical server it abstracts physical resources like processors, memory and storage. This provides the foundation for running virtual machines on this hardware.

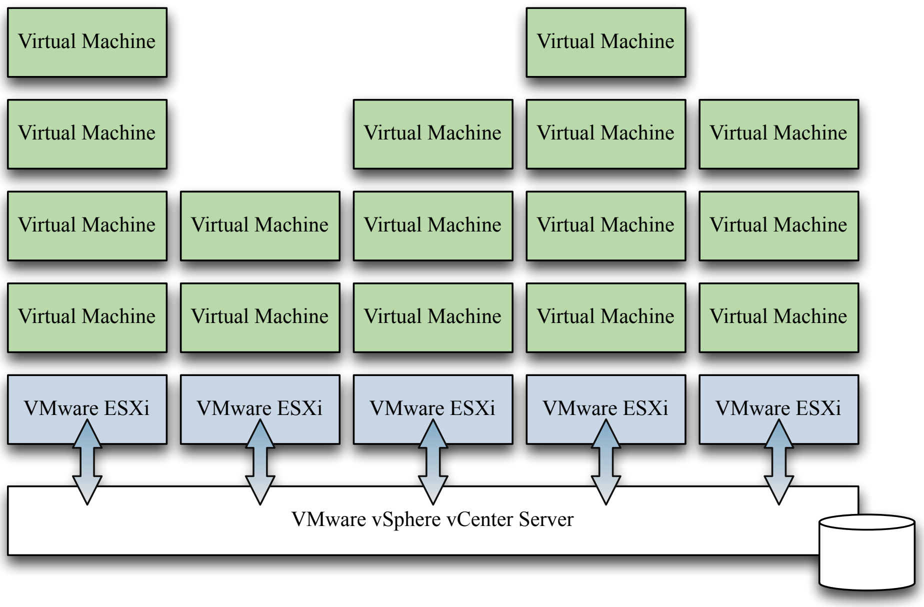

Figure 2: The VMware vSphere architecture

For managing hypervisors *VMware* introduced the *VMware vSphere vCenter Server*, the central point for configuring, provisioning and managing the virtual infrastructure. All data centers including host systems, storage devices, network interfaces and virtual machines are managed by this server. The *VMware vSphere vCenter Server* can be installed on physical hardware, but it is also possible to run this software on a virtual machine.

For accessing the *VMware vSphere vCenter Server* a client software called *VMware vSphere Client* is available. This client connects to the server and all kind of administrative tasks are possible like connecting to virtual machines consoles[3, 4, 7].

2 Libraries and frameworks

In order to query the *VMware vSphere vCenter Server* for data related to the virtualization infrastructure we need an appropriate Java-based framework. We found an open source project called *Virtual Infrastructure (vSphere) Java API (VI Java SDK)*, initially founded by *Steve Jin* who used to work at *VMware* R&D. This framework retrieves all kind of data from a *VMware*-based infrastructures including information about data centers, host systems, virtual machines and all corresponding performance indicators. Requirement for retrieving data is a HTTP(S) connection to the *VMware vSphere vCenter Server* and an appropriate user-account with the needed privileges. For all implemented queries read-only permissions are sufficient.

The *VMware vCenter Server* stores information about its so-called managed entities, like host systems, virtual machines and data centers. Each managed entity is identified by an unique managed object id. Thus the *VI Java SDK* framework enables us to query all kind of data related to the *VMware* infrastructure itself. But for a complete overview we also need information about the physical systems like the hypervisors, their health and sensors states. We retrieve them using a *CIM/WBEM* approach.

The *Web-based Enterprise Management (WBEM)*[3] open standard developed by the *Distributed Management Taskforce (DMTF)* consists of technologies and procedures targeting to unify the management of distributed computing environments. The *Common Information Model (CIM)*[4] also published by the same organization is an open standard for representing managed objects in an IT infrastructure and the relationships between them. It also defines how management information can be exchanged between systems. The goal is a vendor-independent management of an IT environment and its elements.

The *VI Java SDK* requires a *CIM* client library for *CIM/WBEM*-based calls. This framework is part of the *SBLIM* umbrella project for a collection of system management tools to enable *WBEM* on Linux. Using this library we were able to query the *VMware vSphere vCenter Server* for health and sensor data of the physical host systems[3].

3 Implementation

The implementation consisted of three different goals to be achieved. First of all, the automatic provisioning of virtual machines and host systems has to be achieved. This means that the *OpenNMS* network management system will query the *VMware vSphere vCenter Server* for all available host systems and deployed virtual machines and will add them to the database without any human interaction. The second goal is the collection of monitoring data of the entities found by this *VMware* requisition of nodes. Finally two service monitors were used to trigger service-down-events if critical situations like failing hardware or virtual machine power operations occur.

3.1 Provisioning

OpenNMS provides a mechanism for importing nodes from different sources, e.g. *DNS servers*. So we implemented an additional requisition method for the *OpenNMS provisiond* module. Providing the *VMware vSphere vCenter Server's* hostname and credentials is sufficient to import all elements of the virtual infrastructure into the *OpenNMS'* database. First of all a so-called foreign id has to be set, identifying the entity to be imported. This is used to distinguish between already existing nodes and new nodes to be imported into the database. Also a primary interface has to be set – this is used for all operations like polling for data via SNMP and for discovering additional services on a node[2].

For the collection of data stored in the *VMware vSphere vCenter Server's* database it was also necessary to store additional asset fields for a node because this data is not queried via the node's primary interface - this data is

[3] http://dmtf.org/standards/wbem

[4] http://dmtf.org/standards/cim

collected from the management server itself, so the *VMware vSphere vCenter Server's* hostname and the unique management object id for this node are stored for further use in the *OpenNMS'* database.

The implemented Java classes (org.opennms.netmgt.provision.service.vmware) provide the functionality to read the required parameters like hostname and credentials for the *VMware vSphere vCenter Server* from the *OpenNMS'* configuration files and to query the management server for entities like host systems and virtual machines. Additional parameters are available to control whether powered-off or standby machines should also be imported. The *provisiond* configuration contains schedules for all available requisition methods, so it is possible to check for new machines (based on the node's foreign id) on a regular basis. When the so-called provisioning group is generated from the *VMware vSphere vCenter Server* results the *OpenNMS pollerd* starts to collect additional information from these imported nodes. After this process all nodes are available and visible in the *OpenNMS'* web interface.

3.2 Datacollection

The second addressed topic was the polling and collecting of performance metrics. The *OpenNMS' collectd* daemon provides the collection and storage of such metrics from different sources like *SNMP* or *JMX*. For all entities managed by the *VMware vSphere vCenter Server* performance counters are available and we wanted *OpenNMS* to poll these values to generate graphs.

3.2.1 Performance counters

The performance counters are organized in seven groups: *CPU*, *ResCpu*, *Memory*, *Network*, *Disk*, *System*, and *ClusterServices*. In each group several counters and for each counter different aggregation rollup types are available. So a single counter is identified by a group name, a counter name and a rollup type.

Six rollup types are defined - *average*, *latest*, *maximum*, *minimum*, *none*, and *summation* - but not all of them are available for each counter. A counter also defines a key called *level* (valued from 1 to 4) providing an indicator for the importance of the counter. Lower values are more important and are longer stored in the *VMware vSphere vCenter Server's* database[3].

3.2.2 Hardware sensors

Further we wanted to collect data like sensor and health states from the host systems (*VMware ESX* and *VMware ESXi*) via *WBEM/CIM*. For this purpose additional permissions must be granted to the user accessing the *VMware vSphere vCenter Server*. After this it is possible to authenticate against the management server and to generate a ticket for retrieving the sensor data from the host system.

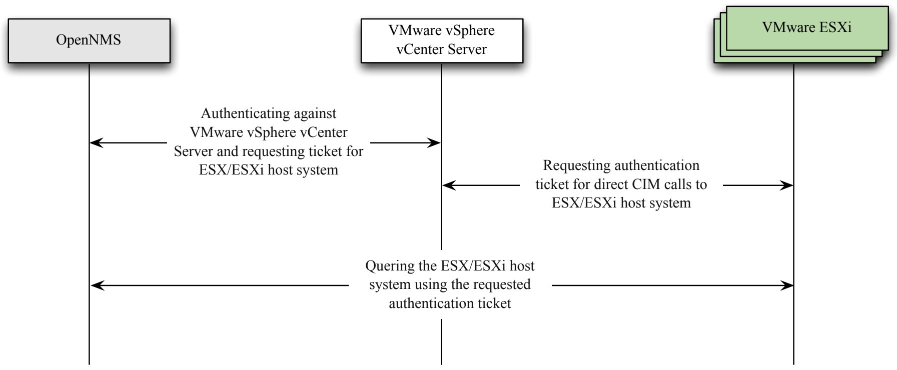
OpenNMS
VMware vSphere vCenter Server
VMware ESXi
Authenticating against VMware vSphere vCenter Server and requesting ticket for ESX/ESXi host system
Requesting authentication ticket for direct CIM calls to ESX/ESXi host system
Quering the ESX/ESXi host system using the requested authentication ticket

Figure 3: CIM query using authentication tickets

Our implementation of all required interfaces and abstract classes is split into two packages, one for querying the *VMware vSphere vCenter Server* (org.opennms.netmgt.collectd.vmware.vijava) and one package for accessing the health and sensor states of host systems via *WBEM/CIM* (org.opennms.netmgt.collectd.vmware.cim).

3.3 Service monitors

OpenNMS service monitors returning only two different states, up or down. We introduced two different service monitors, which are automatically bound to the entities discovered by the *VMware* requisition operation. The first monitor, bound to the name *VMware-ManagedEntity,* will trigger a service-down-event if the power state of a virtual machine or a host system is not equal to "powerOn". The second service monitor called *VMwareCim-HostSystem* queries the hardware sensors of physical *VMware ESX/ESXi* host systems and marks this service as down if one of these sensors is indicating a not fully operational state.

4 Basic configuration

Monitoring the different aspects of *VMware* virtual infrastructures requires different technologies and tools. For the requisition and basic performance monitoring of *VMware* host systems and virtual machines we use the *Virtual Infrastructure (vSphere) Java API (VI Java SDK)*. For the hardware-related monitoring of *VMware ESX/ESXi* host systems common *CIM* queries are used. The query of *CIM* objects requires a host system to be reachable by the *OpenNMS* host - for performance monitoring only the reachability of the *VMware vSphere vCenter Server* is required.

4.1 Preparing the VMware vSphere vCenter Server

First of all a new user is required for *OpenNMS* to access the *VMware* infrastructure. In most environments this can be accomplished by creating a new windows user on the *VMware vSphere vCenter Server* itself and assign a password to this user. After this step it is possible to connect to the *VMware vSphere vCenter Server* using the *VMware vSphere Client* and select the *Administration* view. In this view you can use the *Roles* tab to edit roles on this *VMware vSphere vCenter Server*. The access from an *OpenNMS* host requires read-only permissions and a feature called "CIM interaction". This can be achieved by cloning the "Read-only" role and assign this new role the privilege "Host→CIM→CIM interaction". Finally the new user and role can be assigned to the *VMware vSphere vCenter Server's* root node by right-clicking it and selecting "Add Permission...".

4.2 Configuring the OpenNMS host

For configuring the automatic requisition in *OpenNMS* edit the *provisiond-configuration.xml* file and add a new schedule entry for the *VMware* requisition:

```
<requisition-def import-name="ipaddress"
                 import-url-resource="vmware://username:password@ipaddress">
  <cron-schedule>SECONDS MINUTES HOURS DAY-OF-MONTH MONTH DAY-OF-WEEK YEAR</cron-schedule>
</requisition-def>
```

For *username* and *password* enter the credentials of the user created before and for the *ipaddress* the IP address of the used *VMware vSphere vCenter Server*. In the *cron-schedule* apply the schedule parameters for running this requisition. The following entry will run a requisition for the server 192.168.15.254, the user "OpenNMS-User" and password "secret" every day at 10:15 am.

```
<requisition-def import-name="192.168.15.254"
                 import-url-resource="vmware://OpenNMS-User:secret@192.168.15.254">
  <cron-schedule>0 15 10 * * ? *</cron-schedule>
</requisition-def>
```

Now the *send-event.pl* script is used to signal the *provisiond* daemon to reload the configuration:

```
send-event.pl uei.opennms.org/internal/reloadDaemonConfig --parm 'daemonName Provisiond"
```

When the specified system time matches the defined schedule, all virtual machines and host systems will be synchronized with your *OpenNMS* system. For data collection and monitoring purposes interfaces of nodes will be assigned some additional service names. After the nodes are successfully imported into the *OpenNMS*‘ database all detectors will check the presence of additional services. Also the data collection of *VMware* performance counters and hardware sensors will be started.

Type	**Service name**	**Purpose**
VMware Host System	VMware-HostSystem	VMware HostSystem specific datacollection
	VMwareCim-HostSystem	VMware HostSystem hardware monitoring
	VMware-ManagedEntity	VMware power state monitoring
VMware Virtual Machine	VMware-VirtualMachine	VMware VirtualMachine specific datcollection
	VMware-ManagedEntity	VMware power state monitoring

Figure 4: Service names automatically bound to nodes by the requisition operation

For the data collection a set of service names are used to map different configurations to *VMware* host systems and virtual machines. For both types of *VMware vSphere vCenter Server* managed object different performance counters are available and also hardware monitoring is only available on a physical host system (*VMware ESX/ESXi*).

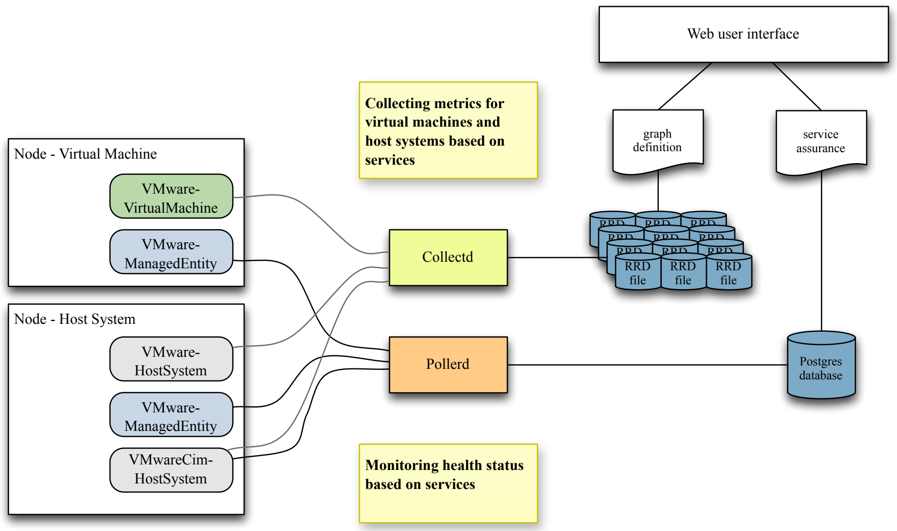

Figure 5: Used service and collection names

The service name *VMware-HostSystem* is used to map to the collection *default-HostSystem* and the service name *VMware-VirtualMachine* is used to map to the collection *default-VirtualMachine*. If you like to modify the attributes, which have to be collected, edit the file *vmware-datacollection-config.xml*. A second service called *VMwareCim-HostSystem* is bound to a *VMware* host system. It is used to collect hardware sensor data from a *VMware ESX/ESXi* host system. The configuration file *vmware-cim-datacollection-config.xml* controls the sensor data to be collected for a host system. For all *VMware*-related data collections it is required to edit the *vmware-config.xml* file and to add the user credentials for all *VMware vSphere vCenter Server* hosts to be queried. A read-only user is required for *OpenNMS* to access the *VMware vSphere vCenter Server*.

For *CIM*-based queries like hardware monitoring the user also requires the privilege “Host→CIM→CIM interaction”. For example this file should look something like this:

```
<?xml version="1.0"?>
<vmware-config>
  <vmware-server hostname="192.168.15.254" username="OpenNMS-User" password="secret"/>
</vmware-config>
```

Two different service monitors are available at present. The first one bound to the service name *VMware-ManagedEntity* checks the power state of a virtual machine or a host system. It will trigger a service-down-event when the state is not equal to “powerOn”. The second monitor is bound to the service name *VMwareCim-HostSystem* and monitors the hardware sensors of *VMware ESX/ESXi* host systems. These numeric sensors (also queried for the host system‘s data collection) also return an overall status, where a value of 5 indicates that it is fully operational. If one of the queried sensors return a inadequate value the service will be marked as down. For both monitors it is required to enter the correct user credentials in the *vmware-config.xml* file on your *OpenNMS* host.

5 Results

Monitoring a closed source *VMware* virtualization environment with open source software is possible. The *VMware vSphere vCenter* provides open and free API’s. Supporting standardized protocols and technologies like Web services, *WBEM* and *CIM* allows highly integrated virtualization and monitoring solutions. For research and further development the source code of this paper is published in *OpenNMS GNU Public License version 3+*. The *OpenNMS* source code is published on *SourceForge*. The implemented features from this paper are published in an *OpenNMS* feature branch called *origin/feature-vmware*. To get a copy of the source code the free version control system *git* is required and can be done with the following commands:

```
user@host:~$ git clone git://opennms.git.sourceforge.net/gitroot/opennms/opennms
user@host:~$ git checkout -b feature-vmware origin/feature-vmware
```

The complete source code will be copied to a local directory *opennms* in users home directory. To build and compile the application from source the minimum requirements of *OpenNMS* are necessary:

- *PostgreSQL* as database for network node and monitoring information
- *Oracle Java Development Kit 1.6* or *OpenJDK 6* as runtime and compiler
- *JICMP* and/or *JICMP6* for *ICMP* protocol implementation for IPv4 and/or IPv6
- *Maven* as dependency management tool for building the *OpenNMS* source code

The source code can now be compiled and deployed with

```
user@host:~/opennms$ ./compile.pl && ./assemble.pl -Dopennms.home=/opt/opennms -Dbuild.profile=dir
```

The *compile.pl* command compiles the java class files. The *assemble.pl* script is used to build jar-package and to deploy these to the */opt/opennms* directory. To initialize the database, configure the Java runtime and start or stop *OpenNMS* the following commands can be used:

```
user@host:/opt/opennms/bin$ ./runjava -s
user@host:/opt/opennms/bin$ ./install -dis
user@host:/opt/opennms/bin$ ./opennms start | stop | restart
```

For further information on compiling and deploying *OpenNMS,* public documentation is available at the *OpenNMS* website[5].

[5] http://www.opennms.org/wiki/Installation:Source#Building

6 Discussion

The developed modules for integrating *VMware vSphere* entities in *OpenNMS* provide an easy and reliable way for importing and monitoring the complete virtual infrastructure. Changes in the infrastructure like adding or removing virtual machines and host systems are automatically reflected by the *OpenNMS* network management system. We tested our modules against different *VMware infrastructure* releases like *VMware Virtual Infrastructure 3.5*, *VMware vSphere 4* and *VMware vSphere 5* and it seems to be reliable and stable.

Based on this functional prototype we have to improve software testing with *JUnit* and data mockup components against the *VMware* API. To visualize the performance metrics it is necessary to define *RRD* graph definitions for the collected metrics. *VMware vSphere vCenter* provides a huge number of different metrics and for this reason we created simple line graphs to display the data. For better understanding it is necessary to combine performance metrics to key performance indicators for decisions for capacity planning and capacity management. The *OpenNMS* community plans to include this feature in the roadmap for the next *OpenNMS* releases. With the current implementation it is possible to fully monitor complex *VMware* environments based on an open source network management solution.

References

[1] Ciuffoletti, A. 2010. Monitoring a virtual network infrastructure: an IaaS perspective. *ACM SIGCOMM Computer Communication Review*. 40, 5 (2010), 47–52.

[2] Finger, A., Thielking-Riechert, K. and Trommer, R. 2010. *OpenNMS*. dpunkt Verlag.

[3] Jin, S. 2009. *VMware VI and vSphere SDK: Managing the VMware Infrastructure and vSphere*. Prentice Hall.

[4] Kamoun, F. 2009. Virtualizing the Datacenter Without Compromising Server Performance. *Ubiquity*. 2009, August (Aug. 2009).

[5] Soundararajan, V. and Anderson, J.M. 2010. The impact of management operations on the virtualized datacenter. *ISCA '10: Proceedings of the 37th annual international symposium on Computer architecture* (Jun. 2010).

[6] Soundararajan, V. and Govil, K. 2010. Challenges in building scalable virtualized datacenter management. *SIGOPS Operating Systems Review*. 44, 4 (Dec. 2010).

[7] VMware, Inc. 2011. *VMware vSphere Basics*. VMware, Inc.,
URL: http://pubs.vmware.com/vsphere-50/topic/com.vmware.ICbase/PDF/vsphere-esxi-vcenter-server-50-basics-guide.pdf