DIAGNOSING GUEST OPERATING SYSTEMS OF VIRTUAL MACHINES LEVERAGING AGENT ARCHITECTURE

Kamil Szczygieł[1], Krzysztof Bielawski[2]

[1] Intratel Sp. z o.o

[2] Faculty of Computer Science, Bialystok University of Technology, Białystok, Poland

Abstract: Maintaining large amount of virtual machines requires a lot of dedication and time from administrator. Using tools provided by virtualization vendors help in daily maintenance. Additionally it is often required to predict future problems. To address this need there are solutions which include analytic mechanisms to lower the risk of possible issues. In open source world there are many competitive tools, but none of them is integrated with virtualization solution. Maintaining large infrastructure using many administrative consoles is difficult and creates a potential for human mistakes. Mentioned software miss one key functionality - it is not possible to deeply monitor guest operating system of the virtual machine while maintaining integration with virtualization software. Solution proposed in this paper was created to solve this issue with agent based diagnostic mechanisms to provide information about network connectivity, resource usage, applications state, system settings and health.

Keywords: virtualization, diagnosis, KVM, virtio

1. Introduction

Running a modern data center is very difficult task even when major part of the infrastructure is virtualized. Maintaining large amount of virtual machines requires a lot of dedication and time from administrator. Diversity of hardware, systems, vendors and technologies is not helping in daily tasks. With introduction of software defined approach in virtualization administration became more centralized, but still requires a lot of time.

Managing virtual infrastructure can be easier by leveraging software provided by virtualization vendors. There are two leading commercial hypervisors - VMware

Advances in Computer Science Research, vol. 11, pp. 137-150, 2014.

ESXi [1] and Microsoft Hyper-V [2]. Maintaining virtual machines running under control of first is done through VMware vCenter Server [3], while managing virtual machines under control of latter is done through System Center Virtual Machine Manager [4]. They allow to perform basic control actions such as creating, editing, destroying, powering on and off and suspending virtual machines. To ensure access to the virtual machines they have high availability mechanisms that in case of failure move them to another physical server. When access to the application running inside virtual infrastructure is critical both technologies provide fault tolerance mechanisms. There is an copy of a virtual machine that is synchronized in real time and will take place of the original one in event of failure. Additionally there are functionalities that allow moving virtual machines between physical servers or storage without interruption.

Managing open source hypervisors such as KVM [5] or Xen [6] can be done through many different solutions [7], starting from less popular such as PetiteCloud [8], through oVirt [9] and OpenNebula [10], to rapidly developing OpenStack [11]. However, they are not as advanced as their commercial counterpart. They provide basic functionalities to manage and control virtual infrastructure, but they lack advanced mechanisms such as high availability, fault tolerance or moving virtual machines between storage. These functionalities are often available through additional software not integrated into one solution. For instance high availability can be delivered through Pacemaker [12] - open source resource manager. Because of fact that these mechanisms are not integrated into one platform they require additional configuration and it is not possible to manage them from one management console.

Mentioned software provides management mechanisms only for virtual machines. In modern datacenters it is often required to manage all of the components such as storage, network, physical, virtual servers and guest operating systems. Administrators are expected to predict future problems and bottlenecks. Virtualization vendors are aware of these requirements and released software such as VMware vCenter Operations Manager [13] or Microsoft System Center Operations Manager [14] providing advanced analytics of virtual and physical components. They are able to predict future problems for instance not enough storage, computing power or performance bottlenecks. Additionally these tools have ability to help administrator to evaluate health of the virtual infrastructure. There is variety of available metrics such as workload (showing how high virtual machine load is), anomalies (rating behaviour of virtual machine compared to the past) or efficiency (showing how efficient your virtual infrastructure is and how you can improve it). Using these tools it is easier to manage and improve virtual platform. While using additional tools that extend functionalities of these tools such as VMware vRealize Hyperic [15] or Management

Packs for System Center Operations Manager it is possible to monitor applications running inside of the guest operating system such as database servers, web servers, SAP [16], application servers, email servers and more.

On the contrary there is much open source software able to monitor all infrastructure components for instance Zenoss [17], Nagios [18] or Zabbix [19]. These tools are able to monitor [20] both physical and virtual infrastructure which allows to see complete infrastructure health from single management platform. However, these solutions are not aware of virtualized components such as virtual switches, virtualized storage or vendor specific features. Therefore they are not able to leverage mechanisms available through virtualization software. It is possible to extend functionalities of these tools by using additional software. However, while managing large virtual infrastructure administrator does not have time to switch between consoles and compare results from many tools. It is very inconvenient and create a potential for human mistakes while evaluating management feedback.

All mentioned software miss one key functionality - it is not possible to deeply monitor guest operating system of the virtual machines while maintaining integration with virtualization software. This shortcoming inspired us to create a solution that will solve this issue. The proof of concept was done for KVM hypervisor which is rapidly developing, but is still missing important tools. By leveraging private communication channel between hypervisor and virtual machine there is no network requirement to exchange messages. By using agent architecture it is possible to diagnose high resource usage of running processes to avoid performance bottlenecks of the virtual machine. Additionally to ensure highest service availability solution is able to diagnose network failures and notify administrator about it. Often there are critical applications running inside guest operating system and ensuring their availability is a priority. Software presented in this paper is able to diagnose and notify when application is not responding. Biggest source of diagnostic data are log files, therefore published solution is also able to track system settings and logs to diagnose occurring errors.

2. Principle of operation

Presented solution leverage agent architecture introduced in paper [21].

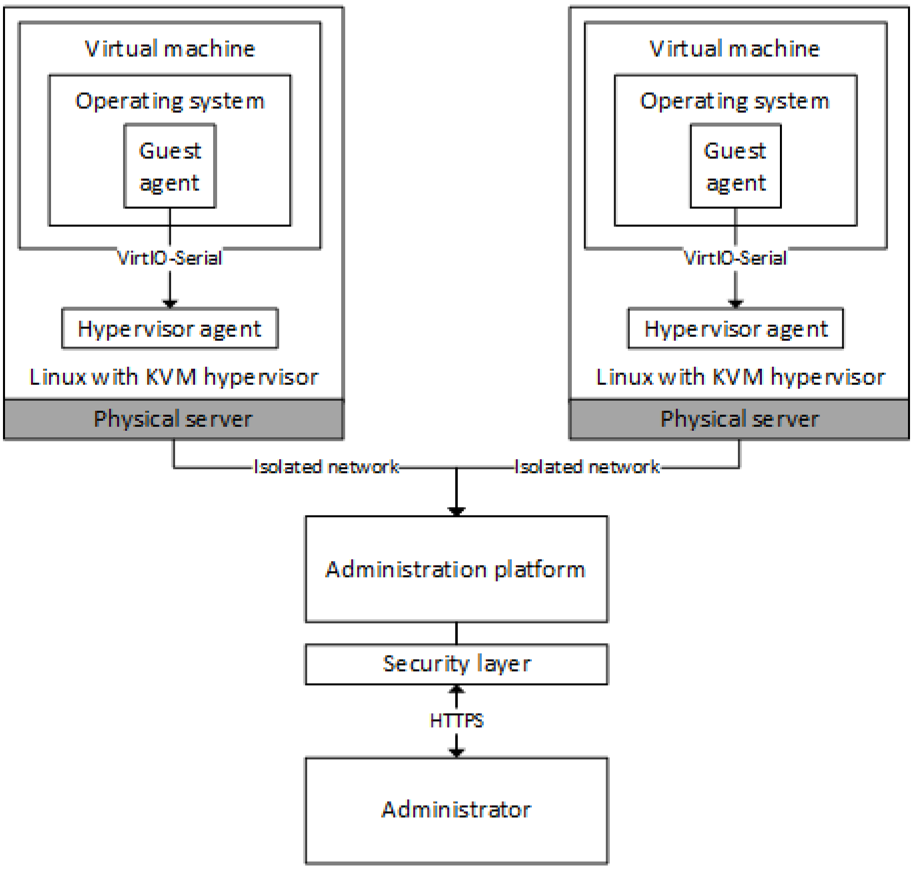

Fig. 1. Architecture of the proposed solution

Open source Linux operating system is installed on each physical server. Hypervisor of choice is KVM which is built into operating system kernel and because of that it does not require any additional modules to be installed. Administration is performed through administration platform - web application intended to maintain and run virtual infrastructure allowing for diagnosis and monitoring. It gathers all diagnostic information from virtual machines and presents them to the administrator. It has built in resolution knowledge base using Windows event IDs [22] to help administrator to easier fix occurring problems. By having all the data in one place, administrator is able to analyse and troubleshoot problems in virtual infrastructure more accurate and faster. Communication is secured by using HTTPS protocol and software security layer. Hypervisor agent is an agent installed on every physical server with hypervisor role. It relays all the diagnostic data from virtual machines to the administration platform using dedicated, secure network. Guest operating system agent is a system service with elevated rights installed inside virtual machine operating system. It periodically performs the diagnostic tasks, gathers results and sends them to the hypervisor agent through hardware communication channel.

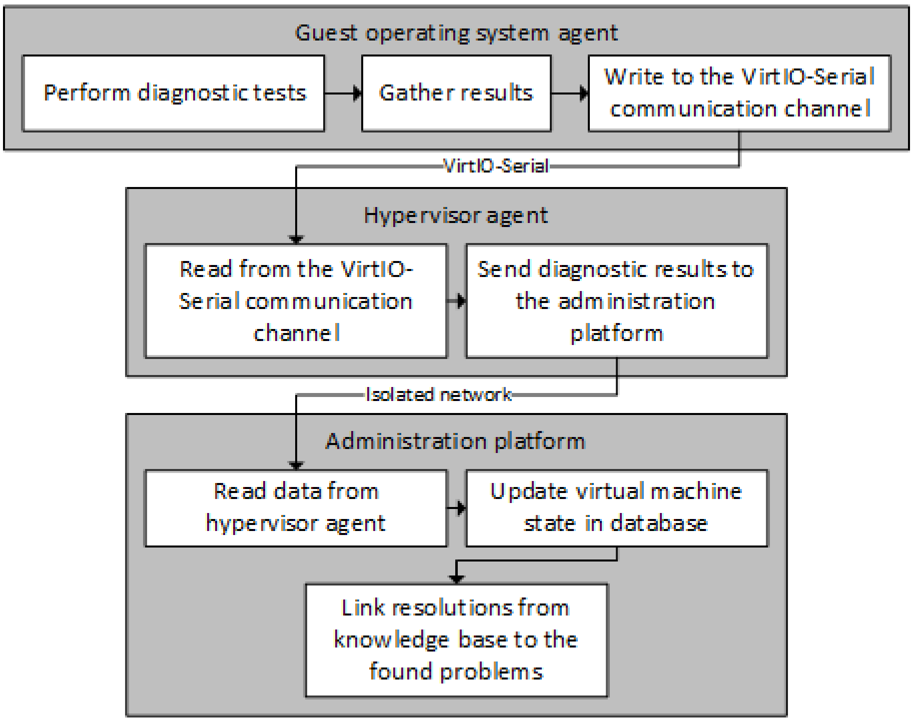
Guest operating system agent
Perform diagnostic tests
Gather results
Write to the VirtIO-Serial communication channel
VirtIO-Serial
Hypervisor agent
Read from the VirtIO-Serial communication channel
Send diagnostic results to the administration platform
Isolated network
Administration platform
Read data from hypervisor agent
Update virtual machine state in database
Link resolutions from knowledge base to the found problems

Fig. 2. Communication between components

Communication in proposed solution is one way only. The agent running inside guest operating system is performing diagnostic tasks such as network, applications, log files, system configuration and resource usage tests periodically. After all tasks are finished, results are converted to the JavaScript Object Notation (JSON) [23] format and sent to the administration platform through VirtIO-Serial [24] communication channel. VirtIO-Serial is a transparent, hardware bridge between virtual machine operating system and hypervisor. Inside virtual machine it acts as a PCI device allowing write and read operations. Then in the physical server VirtIO-Serial can act as many types of output devices such as named pipes, text files, TCP servers, UNIX sockets and more. Because of hardware nature of communication channel, proposed solution is resistant to network problems. Hypervisor agent listens for incoming messages containing diagnostic results from virtual machines and relays them to the administration platform using dedicated, isolated network. Administration platform gathers all messages incoming from hypervisor agents and updates database containing virtual machine information. When diagnostic results contained problems, knowledge base is searched for resolutions allowing for administrator to view quick fix to the occurring problem.

3. Diagnostic cases

This section presents main diagnostic cases included in proposed solution in order of importance with brief description.

One of the major problems in virtual infrastructure maintenance software is that they require network connection to the guest operating system in order to diagnose processes running inside. Therefore we focused on solution that does not have this requirement and is able to perform network connectivity diagnosis and notify administrator in event of failure.

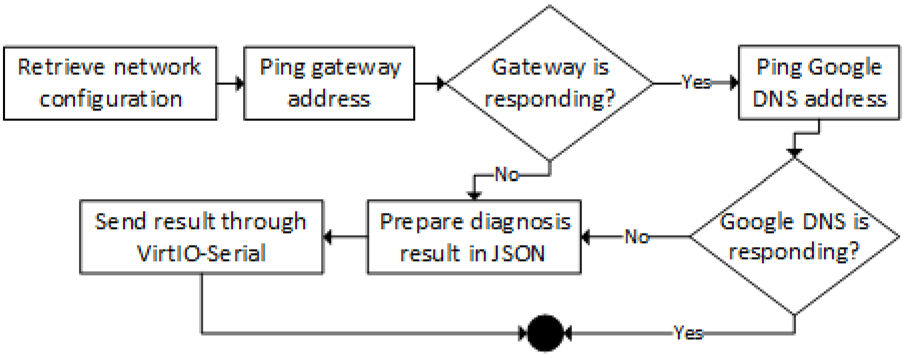
Retrieve network configuration
Ping gateway address
Gateway is responding?
Yes
Ping Google DNS address
No
Google DNS is responding?
No
Prepare diagnosis result in JSON
Send result through VirtIO-Serial
Yes

Fig. 3. Tasks performed during network diagnosis

At first agent is retrieving network configuration of network interface that has is active, is not loopback and has gateway address set. Then it perform simple ping tests to the gateway. If ping fails result of the test is sent through VirtIO-Serial communication channel to inform administrator about complete network failure. If not another ping test is performed to the Google DNS servers. If it fails result is sent to the hypervisor agent which will relay it to the administration platform indicating lack of Internet connection. Otherwise all tests passed successfully and virtual machine has properly functioning network connectivity.

Another significant aspect is that most of the virtual machines have important processes running inside guest operating system that must be maintained. Making sure that they are running properly is crucial. Therefore we developed a diagnostic test that notifies administrator when process is not responding.

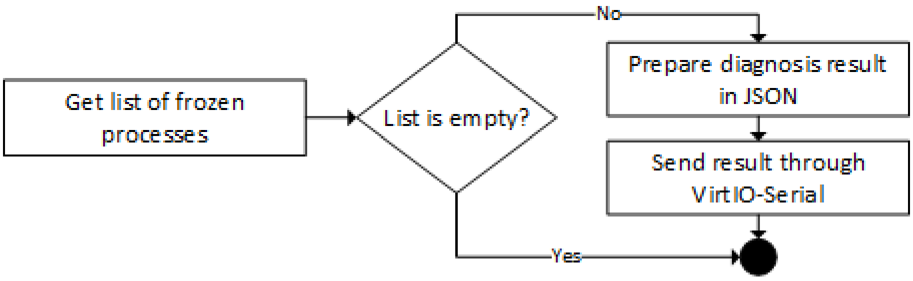
Get list of frozen processes
List is empty?
No
Prepare diagnosis result in JSON
Send result through VirtIO-Serial
Yes

Fig. 4. Tasks performed during processes diagnosis

At first guest operating system agent retrieves list of all frozen processes. If the list is not empty result of the diagnostic test is sent through VirtIO-Serial communication channel. Otherwise, test passed successfully and all processes are running properly.

In most of the enterprises there are internal politics regarding security. We have noticed that there are repeating rules in most of the rules sets. One of them is firewall policy - it should be always enabled. Because of that we implemented diagnostic mechanism that is monitoring firewall and will notify administrator when settings have been changed.

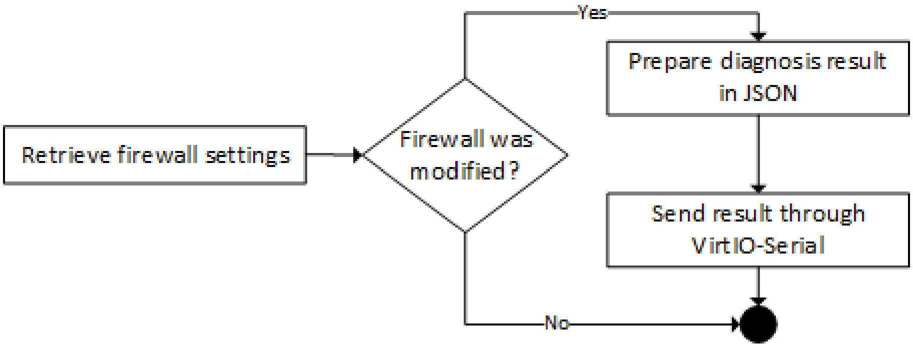
Retrieve firewall settings
Firewall was modified?
Yes
No
Prepare diagnosis result in JSON
Send result through VirtIO-Serial

Fig. 5. Tasks performed during firewall diagnosis

Guest operating system agent periodically retrieves firewall settings and compares them to the previous result. If change of settings was detected result of this diagnosis is sent through VirtIO-Serial communication channel to inform administrator of potential security breach.

Log files are heart of every system. They contain a lot of diagnostic information therefore analysing them provides large amount of feedback. In our solution we focused on error entries that administrator should be notified of. They are sent through communication channel to the administration platform. Administrator can look up resolution in built in knowledge base and decide to fix it.

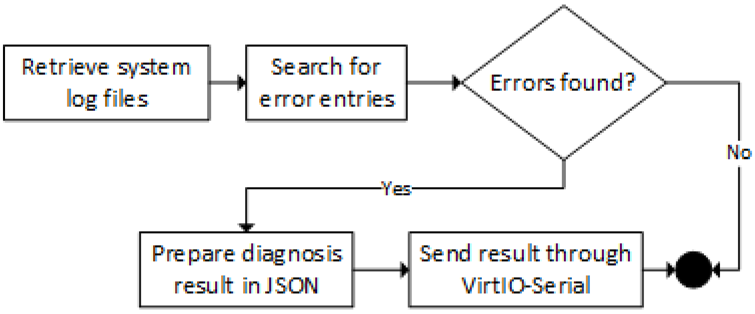
Retrieve system log files
Search for error entries
Errors found?
No
Yes
Prepare diagnosis result in JSON
Send result through VirtIO-Serial

Fig. 6. Tasks performed during logs diagnosis

At first guest operating system agent retrieves all system log files and searches for error entries. If they are found result of this diagnosis is sent through VirtIO-

Serial communication channel to inform administrator about failure. If resolution to the error has been found in knowledge base it will be linked to the particular virtual machine allowing administrator to look up resolution to the occurring errors.

Performance is very important factor in virtualization. Being aware when virtual machine is using a lot of resources may prevent application performance issues. With this knowledge administrator can either diagnose why virtual machine is under such high load or assign more resources to the virtual machine. Therefore proposed solution has diagnostic test that notifies administrator when virtual machine is using a lot of resources.

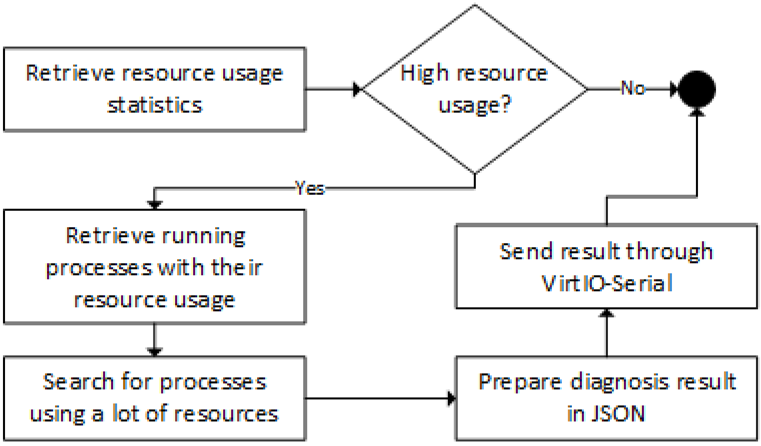
Retrieve resource usage statistics
High resource usage?
No
Yes
Retrieve running processes with their resource usage
Send result through VirtIO-Serial
Search for processes using a lot of resources
Prepare diagnosis result in JSON

Fig. 7. Tasks performed during resources diagnosis

At first agent retrieves resource usage statistics to analyze usage. When very high usage is detected it will retrieve list of all running processes with their resource usage, filter it and prepare diagnosis result containing all resource intensive applications. Such list is sent through VirtIO-Serial to the administration platform.

Important concern for the administrator are applications running inside of operating system. Installation of software may leave system vulnerable and therefore compromised. Diagnostic test proposed in our solution compares installed software and notifies administrator through administration platform when application has been installed or removed. It allows for administrator to have more detailed view on the software that is being installed or removed inside of operating system.

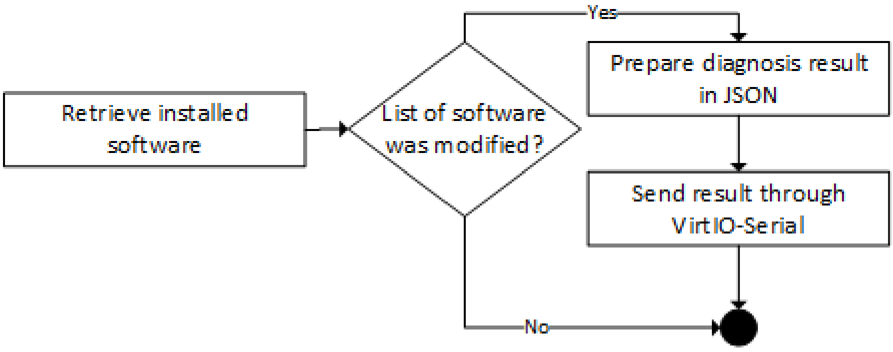
Retrieve installed software
List of software was modified?
Yes
Prepare diagnosis result in JSON
Send result through VirtIO-Serial
No

Fig. 8. Tasks performed during applications diagnosis

Guest operating system agent periodically retrieves list of installed software and compares it to the previous result. If any application was installed or removed result of this diagnosis is sent through VirtIO-Serial communication channel to inform administrator.

4. Case study

Presented solution was leveraged to create a self-service portal where customers are able to order a customized virtual machines. They are able to choose an operating system of their preference from predefined list containing Windows and Linux. Then it is possible to adjust virtual machine hardware such as amount of virtual CPUs, RAM and additional disks. To provide even more personal solution they are able to modify system hostname and administrator password. Additionally customer is able to order a virtual machine with preinstalled and preconfigured software such as databases, identity, web and mail servers. Parameters required by additional components are also customized during virtual machine creation process. User is able to control complete lifecycle of the instance with power on, off, suspend and delete actions. Additionally it is possible to preserve the current state of the virtual machine and return to it later with included snapshots mechanism. Self-service portal delivers variety of information about instance such as resource usage, health and network state.

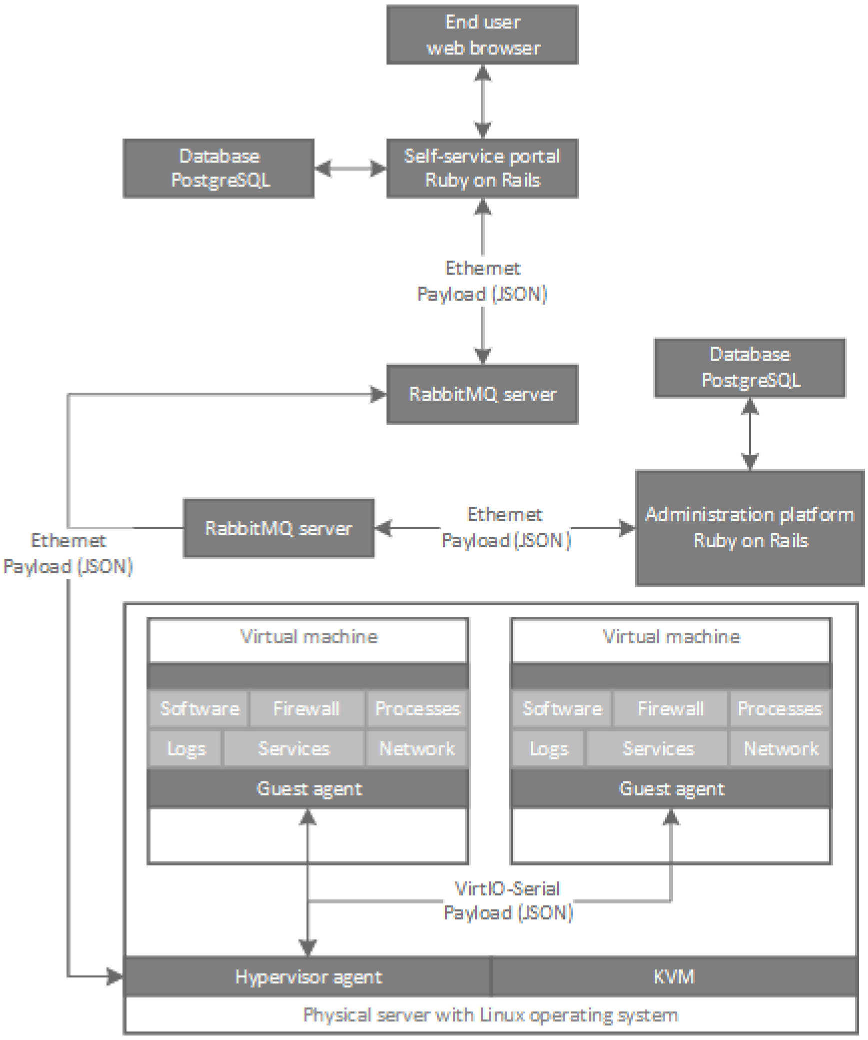

Fig. 9. Architecture of presented case study

Communication between components in this case study is depicted in figure 9. Self-service portal and administration platform are built using Ruby on Rails [25], open source web framework written in Ruby [26]. To meet requirements of modern, responsive web design technologies such as jQuery [27] and asynchronous calls are leveraged. Self-service portal is using PostgreSQL [28] open source database engine to store user and virtual machine data. To integrate with solution proposed in this paper RabbitMQ [29] was introduced. RabbitMQ is open source message broker to exchange messages between components in asynchronous manner. It implements Advanced Message Queueing Protocol (AMQP) [30] to control payload flow. In presented case there are two RabbitMQ servers – one to handle communication between self-service portal and hypervisor agent, second to handle communication between administration platform and hypervisor agent. By separating message brokers components are completely independent and can work separately. Failure of self-service RabbitMQ server does not influence availability of administrative capabilities through administration platform. Payload in messages exchanged between components is in JSON format. Presented use case leverage solution depicted in this paper to install and preconfigure software delivered to the virtual machines. When a virtual machine with database server is requested, new virtual machine is cloned from predefined template image and configured with user hostname and password. Then self-service portal requests an installation of database server software inside of the virtual machine and preconfigure configuration files. In meantime, user can observe whole process in self-service portal, because guest agent is notifying portal about currently performed operation. After whole process is completed, user is notified that a virtual machine is ready to use. Minimized results of diagnostic cases presented in this paper are visible to the user in virtual machine information window after deployment process is finished.

5. Conclusion

In this paper a complete solution is depicted that includes functionalities providing diagnostic tests of the most frequently occurring problems inside virtual machines. Agent running inside of guest operating systems performs network connection diagnosis to ensure network connectivity. Additionally it diagnoses system settings and installed applications to make sure their configuration meets internal security policies. The agent periodically diagnoses state of running processes to ensure that critical applications are running properly. To avoid performance bottlenecks, the solution is diagnosing resource usage inside a virtual machine. Additionally the agent diagnoses system log files and events to provide most detailed diagnostic information. By gathering results provided by these tests and depicting them in administration platform, administrator is able to evaluate virtual infrastructure health and react to problems as they emerge, before they influence performance or availability of the services. Because of hardware private communication channel network failure does not influence ability to inform administrator about problems. Presented mechanisms are able to integrate into one management platform without additional configuration and provide management mechanisms for virtual machines, storage, network, physical servers and guest operating systems including services and applications.

References

[1] VMware vSphere: [http://www.vmware.com/products/vsphere]

[2] Microsoft Hyper-V: [http://www.microsoft.com/en-us/server-cloud/solutions/virtualization.aspx]

[3] VMware vCenter Server: [http://www.vmware.com/products/vcenter-server]

[4] System Center Virtual Machine Manager: [http://technet.microsoft.com/en-us/library/gg610610.aspx]

[5] KVM: [http://www.linux-kvm.org/]

[6] Xen: [http://www.xenproject.org/]

[7] Sotomayor, B., Montero, Ruben S.m Llorente, I.M., Foster, I.: Virtual Infrastructure Management in Private and Hybrid Clouds, Internet Computing, IEEE, 2009.

[8] PetiteCloud: [http://petitecloud.org/]

[9] oVirt: [http://www.ovirt.org/]

[10] OpenNebula: [http://opennebula.org/]

[11] OpenStack: [http://www.openstack.org/]

[12] Pacemaker: [http://clusterlabs.org/]

[13] VMware vCenter Operations Manager: [http://www.vmware.com/products/vrealize-operations/]

[14] System Center Operations Manager: [http://technet.microsoft.com/en-us/library/hh205987.aspx]

[15] VMware vRealize Hyperic: [http://www.vmware.com/products/vrealize-hyperic/]

[16] SAP: [http://www.sap.com]

[17] Zenoss: [http://www.zenoss.com/]

[18] Nagios: [http://www.nagios.org/]

[19] Zabbix: [http://www.zabbix.com/]

[20] Marik, O., Zitta, S.: Comparative analysis of monitoring system for data networks, Multimedia Computing and Systems (ICMCS), Marrakech, 2014.

[21] Szczygieł, K., Bielawski, K.: Controlling KVM virtual machine guest with VirtIO, Design, development and implementation of real-time systems, Warsaw, 2013.

[22] Windows Events: [http://technet.microsoft.com/en-us/library/dd299434(v=ws.10).aspx]

[23] JavaScript Object Notation: [http://json.org/]
[24] VirtIO-Serial: [http://fedoraproject.org/wiki/Features/VirtioSerial]
[25] Ruby on Rails: [http://www.rubyonrails.org]
[26] Ruby: [http://www.ruby-lang.org]
[27] jQuery: [http://www.jquery.com]
[28] PostgreSQL: [http://www.postgresql.org]
[29] RabbitMQ: [http://www.rabbitmq.com]
[30] Advanced Message Queueing Protocol: [http://www.amqp.org]

DIAGNOZOWANIE SYSTEMÓW OPERACYJNYCH MASZYN WIRTUALNYCH PRZY WYKORZYSTANIU ARCHITEKTURY AGENTOWEJ

Streszczenie: Zarządzanie dużą ilością maszyn wirtualnych wymaga od administratora wiele czasu oraz poświęcenia. Wykorzystanie narzędzi dostarczonych wraz z oprogramowaniem wirtualizacyjnym ułatwia utrzymanie infrastruktury. Dodatkowo często wymagane jest przewidywanie problemów, które mogą wystąpić w środowisku wirtualnym. W tym celu powstało oprogramowanie zawierające mechanizmy analityczne zmniejszające ryzyko awarii. W świecie oprogramowania open source istnieje wiele narzędzi, lecz żadne z nich nie jest zintegrowane z platformą wirtualizacyjną, a w związku z tym zarządzanie infrastrukturą jest trudne. Przedstawionym rozwiązaniom brak jest jednej istotnej funkcjonalności - możliwości dokładnego monitorowania systemów operacyjnych. Zaproponowane w publikacji oprogramowanie w oparciu o architekturę agentową stara się rozwiązać ten problem poprzez wykorzystanie mechanizmów dostarczających informacji o stanie sieci, zużyciu zasobów, stanie aplikacji, ustawieniach systemu oraz jego zdrowiu.

Słowa kluczowe: wirtualizacja, diagnostyka, KVM, virtio

Artykuł zrealizowano w ramach pracy badawczej S/WI/2/13.