BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF INTELLIGENT SYSTEMS
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

SUMMARY ADMINISTRATION OF VIRTUAL MACHINES IN THE OVIRT PROJECT

PŘEHLEDNÁ SPRÁVA VIRTUÁLNÍCH STROJŮ V PROJEKTU OVIRT

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR **SAMUEL MACKO**
AUTOR PRÁCE

SUPERVISOR **Doc. Mgr. ADAM ROGALEWICZ, Ph.D.**
VEDOUCÍ PRÁCE

BRNO 2018

Abstract

This thesis tackles the idea of virtualization and virtual machines. Theoretical part covers basics of virtualization from various aspects. It introduces a concept of virtualization and various architectures used to achieve it. The thesis also examines most popular implementations of mentioned architectures as well as commercially available virtualization solutions using those implementations. The aim of its practical part is to design and implement desktop application for administrator level overview over virtual environment running on oVirt. The main goal is to solve some of its known issues regarding accessibility of its data.

Abstrakt

Táto práca sa zaoberá myšlienkou virtualizácie a virtuálnych počítačov. Teoretická časť pokrýva základy virtualizácie z rôznych aspektov. Predstavuje koncept virtualizácie a rôzne architektúry využívané na jej dosiahnutie. Práca takisto skúma populárne implementácie spomenutých architektúr rovnako ako komerčne dostupné virtualizačné riešenia vyžívajúce tieto implementácie. Cieľom praktickej časti je navrhnúť a implementovať desktopovú aplikáciu pre administrátorskú úroveň prehľadu virtuálneho prostredia bežiacom v službe oVirt. Hlavným cieľom je vyriešiť niektoré z jeho známych problémov súvisiacich s prístupnosťou dát.

Keywords

Virtualization, virtualization architectures, virtualization implemantations, hypervisor, container, oVirt, python, ovirt-python-SDK, PyQt5.

Klíčová slova

virtualizácia, architektúry virtualizácie., implementácie virtualizácie, hypervisor, kontajner, oVirt, python, ovirt-python-SDK, PyQt5

Reference

MACKO, Samuel. *Summary Administration of Virtual Machines in the OVirt Project.* Brno, 2018. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Doc. Mgr. Adam Rogalewicz, Ph.D.

Summary Administration of Virtual Machines in the OVirt Project

Declaration

I declare that I created this bachelor's thesis as an original author's work under the supervision of Doc. Mgr. Adam Rogalewicz, Ph.D. and Ing. Lukáš Svatý. I included and properly cited all sources of information I used for this thesis.

.....
Samuel Macko
May 15, 2018

Acknowledgements

I would like to thank my supervisors Doc. Mgr. Adam Rogalewicz, Ph.D. and Ing. Lukáš Svatý for their endless patience and guidance.

Contents

Chapter 1

Introduction

Although it may seem that virtualization is the invention of last few years its concept was first brought up in 1960s. It was firstly implemented by IBM to split resources of their massive mainframe machines among multiple virtual machines that could work independently from each other. The main goal was to use resources more efficiently. Today, virtualization of hardware resources is heavily used and its popularity is rapidly increasing, mainly due to its advantages and accessibility.

There are many types of architectures and products to choose from depending on what we need. We can virtualize a whole physical hardware or just part of it for a single application. For an easy deployment, we can create a container to hold all packages and everything else needed to run our application. For even more autonomy and flexibility we can use virtual machines which have their own autonomous operating system. There are even solution for whole virtual environments with multiple virtual storages, virtual network interfaces and virtual machines all running on multiple instances of physical hardware. For an easy accessibility, we can use a virtual storage which can use multiple disks connected, for example, by a network but they all act as a single instance. These are just some of the real-world use cases for the virtualization technology.

In the chapter 2, I will introduce a concept of virtualization, its advatages and disadvantages and I will explain some of the most popular architectures of virtualization and what are they used for. In the next chapter, I will go through specific implementations of mentioned architectures and briefly describe how they work. Chapter 4 will examine biggest companies offering virtualization solutions and explore their most popular products.

Last chapters 5, 6 and 7 will focus on the product of this thesis, a desktop application providing overview over virtual environment running in oVirt (see 4.5.2). oVirt is an upstream project of RHEV (see 4.5.1) which provides web based user interface targeted to admins, but it has same issues which I am trying to solve.

Chapter 2

Virtualization

2.1 Virtualization basics

In short, virtualization means emulation of hardware within a software platform. To explain basics of virtualization I would like to start by brief explanation of architecture of standard computers. First, we need hardware layer which contains processing unit, storage and other hardware devices, on top of this layer is operating system which abstracts functioning of hardware layer and offers application layer means to communicate with HW layer. Operating system runs as privileged software which means that it is generally able to perform any operation supported by hardware. Its role is to create an interface which simplifies or ignores implementation of components on lower levels of a hierarchy to make using them on application level easier. On the top of an operating system is the application layer which consists of user programs which are less privileged and can generally perform only operations that are permitted to them by operating system.

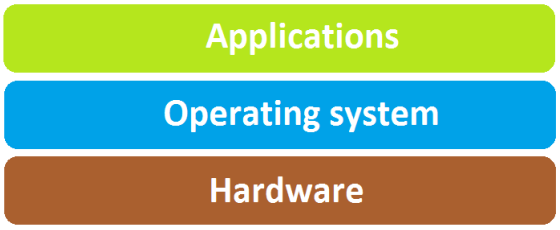

Figure 2.1: Standard architecture

Virtualization allows creating multiple entities of operating system and application layers on single HW layer. This is done by inserting additional layer of system software between operating system and lower layers. This virtualization layer is called hypervisor or VMM (virtual machine monitor) and there are two main types of it:

- Type I hypervisor (or bare-metal/native hypervispor) runs directly on host system's hardware with highest level of privilege and has full control over application layer running on top of it. Type I hypervisors have generally better performance than type II hypervisors because they don't have to communicate with hardware layer through an operating system thus can utilize full potential it. Some of them need a special privileged virtual machine called Domain-0 from which it can be managed and controlled.

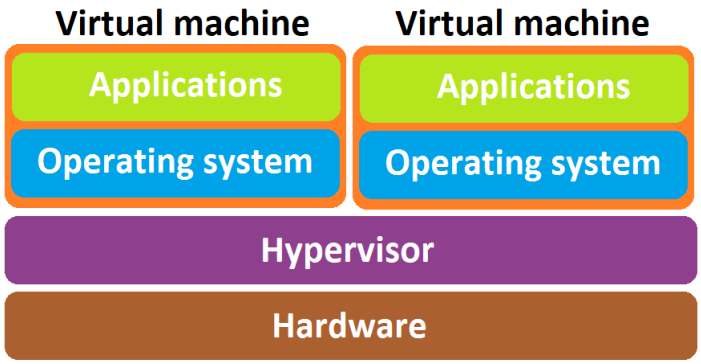

Figure 2.2: Type I hypervisor

- Type II hypervisor (or hosted hypervisor) : runs on top of the host operating system in an application layer. This means that hypervisor doesn't require specific drivers for I/O operations and allows running the virtual environment within already existing environment. It is less efficient than type I hyperisor because communication between virtual machines and hardware transitions also through the host operating system. But on the same level as hypervisor can run applications directly on host operating system.

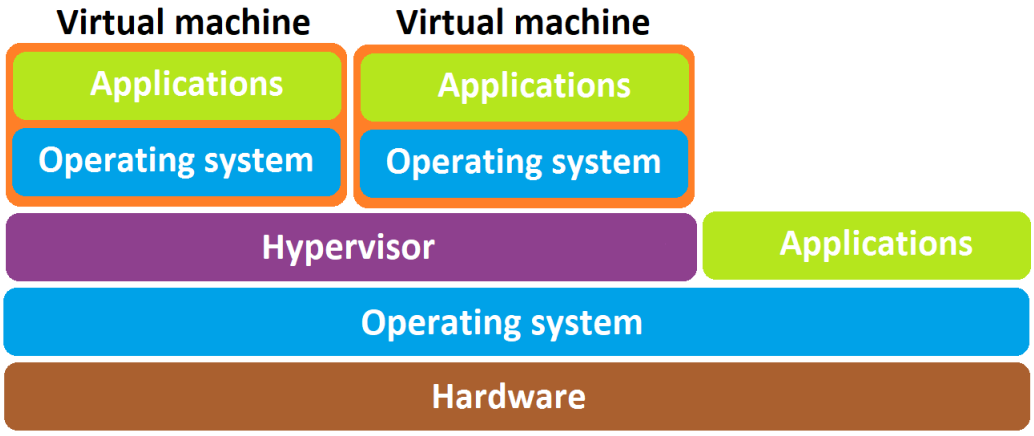

Figure 2.3: Type II hypervisor

2.2 Advantages of virtualization

As every other technology, virtualization have some advantages and disadvantages. Lets start with advantages:

1. Efficiency: Virtualization allows more efficient use of the host machine - multiple virtual machines can run different services. Running multiple services on same server can be dangerous for multiple reasons so it is considered a bad practice. Most obvious problem is with hardware overhead, if numerous very resource-demanding services run on single physical machine it can slow it down considerably or even crash it which makes business's services very unreliable. On the other hand, if we run only single

service on a whole physical host than the service can use for example only 20% of server machine's resources (this is very common especially noways as hardware is becoming more and more powerful and the average service utilizes only about 10% to 15% of all available resources). This way we waste remaining 80% of resources which, in the long run, unnecessarily raises expenses on managing physical hosts. Rather, we can run various virtual servers on one physical hardware and on each of them run a single service.

2. Cost reduction: By concentrating virtual machines on fever hosts we can reduce expenses associated with running large number of physical machines in various ways. We need less physical space for storing hosts, less energy for cooling is needed, less energy is spent on powering the hardware and managing fewer physical devices is also easier and cheaper. Companies with fewer than 1000 employees spend up to 40% of their IT budget on hardware [4], which is the cost that can be greatly reduced by virtualization. Although this benefits more larger companies using plenty of server machines, smaller companies can benefit greatly from virtualization too. Using less energy/space/etc. means that virtualization is also friendlier to environment.

3. Flexibility: Increasing the number of physical workstations or servers is very financially and time consuming process. We need new physical space, order new machines, set them up and so on. With suitable use of virtual machines, the whole process can be easier and faster. There are no more additional hardware costs and administrators can easily setup and manage virtual machines using virtual machine management software. By using templates we can make creating new virtual machines even faster by automatizating of setting up procedures. When some hardware which hosts virtual machines becomes obsolete or it just needs to be out of service for maintenance reasons, we can migrate virtual machines to another physical hardware.

4. Testing: Virtual machines are completely isolated from each other which gives us possibility for testing environments with completely different operating systems and configurations. Even extreme situations are easy to set up or changed. Compared to physical machines, virtual machines can be added and removed very fast. QA teams often have multiple virtual machines with which they speed up testing and therefore development process.

5. Security: All virtual machines are confined entities separated from other software so when one of them gets attacked, gets virus, or for any reason fails, only that one virtual machine fails. While problematic virtual machine is being diagnosed and repaired, another backup virtual machine can take it's place and continue running the affected service which greatly reduces down-time and increases reliability.

6. Isolation: Virtual machines are hardware independent which means that their current state can be captured and reproduced on another physical host in process called migration. This reduces down-time even more because we can run all our services temporarily from another host while former host is being down due to maintenance. There is also a possibility of live migration - i.e. migration without downtime. It is the ability to move running virtual machine between physical hosts with no interruption of service. The virtual machine remains powered on and user applications continue on running while the virtual machine is relocated to a new physical host.

In the background, the virtual machine's RAM is copied from the source host to the destination host. Storage and network connectivity are not altered.

2.3 Disadvantages of virtualization

Disadvantages: The disadvantages of virtualization are mostly those that are associated with any transition to a new technology and can be overcame by careful planing and professional implementation.

1. Overloading: This problem lies in wrong or incomplete estimation of amount of hardware resources needed to handle desired virtual environment. Virtualization carries along additional bandwidth in form of hypervisor and other components which is not neglectable. Other extreme is not utilizing full potential of physical host capabilities and wasting resources in long run by using more hosts that are actually needed. Basic rule of thumb is to use around 80% of physical machines resources.

2. Bandwidth: The volume of data transfered through network might be too much to handle for single network interface card (NIC), this can lead to slower network transfers and therefore offering of services. One of possible ways to solve this is to use host machine with multiple NICs.

3. Need for adjustments: In some cases, adapting a virtualization technology requires rewriting or patching some pieces of software to be compatible with virtual environment.

4. Cost: To run multiple machines on a single host machine, we need it to be sufficiently powerful. This means that additional investments into hardware may be needed. Another investments are into virtualization software and managing virtual machines.

5. Learning curve: Conversion to and managing virtual environment will require IT staff with necessary training. The beginning stages can be painful due to lack of experience with new technology. Many modern virtualization solution vendors offer trainings for their products.

6. Vulnerability: Although virtualization brings certain security benefits it also brings a big risk in form of potential damage cost by lower level layers corruption. In physical environment, if an operating system of one machine gets corrupted than only that one machine is affected. But in virtualized environment if hardware, operating system, or hypervisor of a host gets damaged than all virtual machines running on it can potentially become unavailable. This problem can be reduced by regular backups and snapshots which allow an easy transfer of virtual machines to a new host.

7. Licensing: Majority of software vendors consider virtual machine to be exactly the same as physical machine so if certain piece of software is needed on multiple virtual machines than we have to pay for a license for each one of them. We can try to solve this by using open-source software. This is becoming less of a problem nowadays because more and more software vendors are adjusting their view on virtualization (for example Windows Server 2016 uses Per Core Licensing which means that each copy of Windows Server will license 2 physical processor cores).

2.4 Virtualization architectures

Virtualization technology is spreading rapidly and today there are several architectures that implement this concept. This section provides basic concepts and the chapter 2 we will discusses some of their implementations.

The main variants are the following:

1. Full virtualization
2. Paravirtualization
3. Operating system virtualization
4. Other types of virtualization

2.4.1 Full virtualization

It provides a total virtualization of hardware which means that every virtual entity runs as it is running on a physical hardware completely unaware that its platform is being virtualized. When a virtual machine wants to access hardware it accesses virtual hardware provided by a hypervisor which finally accesses physical hardware. Hypervisor thus acts as the only bridge between virtual machine and physical hardware. This is a reason why the full virtualization is in terms of performance behind non-virtualized machines.

Software assisted virtualization

Historically, the first full virtualization solution introduced in 1969. With this approach, every part of a physical hardware is virtuallized which has advantage in fact that every operating system or application can be run completely without modifications. On the other hand every operation made by operating system of virtual machine needs to be simulated and checked if it doesn't conflict with any other virtual machine or hypervisor [11] which makes this process very resource-expensive.

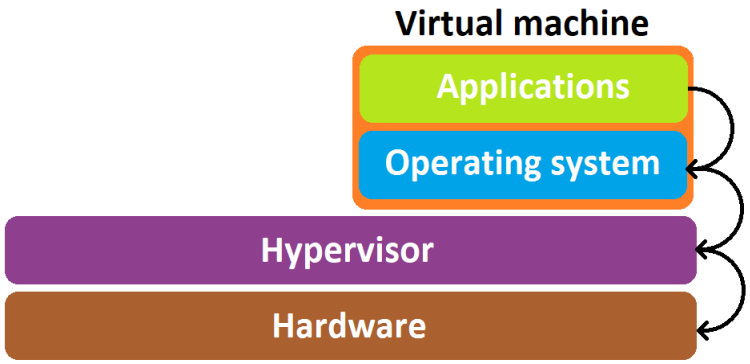

Figure 2.4: Software assisted virtualization

Hardware assisted virtualization

Introduced by IBM in 1972. This approach reduces the problem of massive overhead in software assisted virtualization by extending functionality of hardware (mainly CPU) by new instructions allowing virtual machines to directly access physical hardware without

expensive mediators. Normally, x86 operating systems need to have a direct access to hardware resources. Software-based virtualization solves this problem of access to hardware by virtualizing entire hardware layer but for price of wasting a big chunk of hardware resources. On the other hand, in hardware assisted virtualization is this overhead noticeably reduced because processor no longer needs to be emulated and can directly interact with application layer. Another advantage is that it will work 'right of the box' meaning that we don't need to upgrade or change anything, all we need to do is to use processor supporting hardware-based virtualization technology. Main representatives of this method are Intel VT and AMD-V.

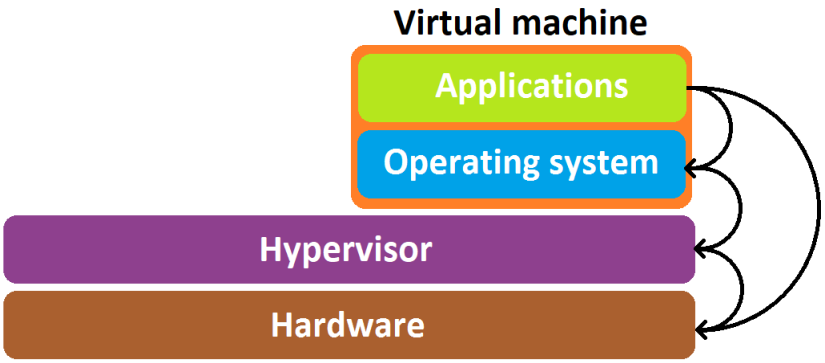

Figure 2.5: Hardware assisted virtualization

2.4.2 Paravirtualization

Introduced in 1972 by IBM, this technique was implemented to increase performance of virtualized environment closer to non-virtualized. To use paravirtualization, kernel of the operating system needs to be modified, mainly it needs to have replaced all privileged operation running only in ring 0 by so called 'hypercalls'. Hypercalls allow guest operating system to send system calls directly to the hypervisor without the need of hardware simulation. Virtual machine can therefore access some part of the hardware straight without going through virtualized hardware on the top of hypervisor which greatly increases performance of some operations. This only works for some parts of the hardware, for other parts, virtual machine still needs to use virtualized hardware. Paravirtualized virtual machine is 'aware' of the fact that it is being virtualized which gives it ability to use hypercalls. On the other hand, as mentioned above, in order to use this technique, operating system needs to be altered in a non-trivial way which typically limits its use to a Linux based operating systems because they allow such source code modifications. Paravirtualization was popularized by Xen hypervisor. Today, most virtualization solutions use it as a norm (for example Red Hat Xen, VMware's family and others).

2.4.3 Operating system virtualization

Also known as shared kernel virtualization, introduces 'light-weight' virtualization. To understand this concept, first we need to have at least very basic understanding of what are kernel and root file system and what are their roles. Kernel is central part of the operating system, which mediates communication between other parts of operating system and physical hardware. Root file system contains all the files, libraries and practically all

utilities necessary for operating system to work properly. In shared kernel virtualization every virtual machine has its own root file system but uses host operating system's kernel which they share among each other. Kernel has the ability to dynamically switch the current root file system to a different one without the need to reboot the whole system (technique also known as 'chroot'). In this case, virtual machines are referred to as 'containers' due to the fact that they are not completely separated but share host machine's kernel. Shared kernel means very little overhead compared to other virtualization concepts and thus higher performance rate. Despite its light-weight nature, this concept also supports advanced features such as isolating memory space, regulating memory, network, CPU and I/O usage. Some implementations even allow live migration. The biggest advantage of container-virtualization is superior efficiency and very little overhead compared to other types of virtualization because it doesn't have to emulate all the hardware. Interactions between software and hardware are handled by operating system's kernel inside container. The major disadvantage of this technique is that container's operating system must be compatible with kernel on which it runs (container operating system must be designed for type and version of kernel that is being shared). Further, container environments cannot execute some top-level actions, mount or dismount file systems and so on whereas fully virtualized solution gives us fully independent environment where user isn't restricted in any way. Well known solutions are Linux VServer, Jail for FreeBSD, Zone for Solaris, Virtuozzo for Windows, Rosetta for Mac OS and others.

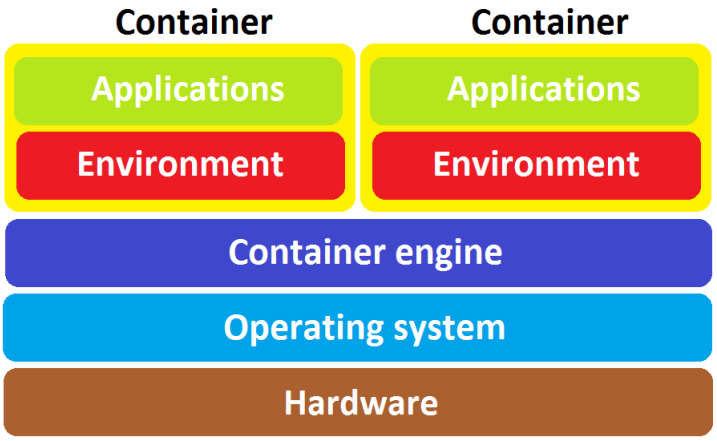

Figure 2.6: Operating system virtualization

2.4.4 Other types of virtualization

Network virtualization

Main idea is to create multiple virtual sub-networks (channels) that run on single physical network and are independent from each another. Each one of the virtual networks can have different bandwidth related, security, or other restrictions and we can regulate traffic on each channel independently. Monitoring individual networks with their own purposes and settings is also easier and faster. It increases reliability and durability of network too as if one of virtual networks is for whatever reason overloaded (or down), other networks are not affected. Most of the modern hypervisors implement virtual networking in some form. Network virtualization can be further divided into two sub-categories:

1. Internal: can be used for communication between software and virtual machines or to mimic network to external devices. Internal network uses virtual network devices that act as physical devices and enables single system to appear as a network.

2. External: used in Virtual local area networks (VLAN) and Virtual private networks (VPN) [7].

Application virtualization

This technique separates the application layer from the underlying operating system layer which means that application runs in 'encapsulated' state independently from operating system. Application is put into a capsule which contains copies of all shared resources that application would need to run as well as all DLLs (dynamically linked libraries), drivers, registry entries and so on.

In practical terms it means:

1. Application created for certain operating system can run on a different one, for example native Windows applications can be run under Linux distribution or vice-versa.

2. We can isolate suspicious or malicious software and inspect it without the danger of infecting the whole system.

3. Simultaneously running applications that would otherwise conflict each other.

4. Easy deployment applications.

Some well known examples are Wine which is used to run Microsoft Windows applications on Linux, ThinApp from VMware, Xenocode from Code System Corporations and others.

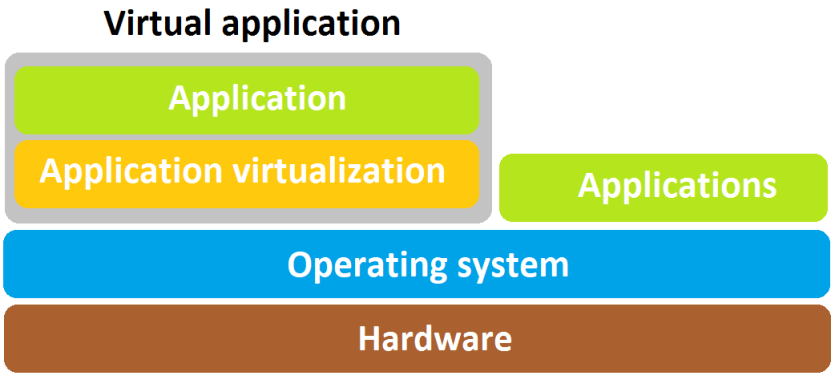

Figure 2.7: Application virtualization

Desktop virtualization

Similarly as application virtualization it separates desktop environment from underlying physical computer. Desktop virtualization functions on a client-server model. Client desktop environments run on virtual machines stored on servers and can access them via client device which can be standard PC, tablet, thin client which is specialized device capable of displaying virtual desktop environment or others. This technique is growing on popularity mainly because of cloud computing. It allows us to access desktop from any device or

location (in practical terms it means that we can work in same environment from anywhere without the need to bring 'work computer'). By using desktop virtalization we can use cheaper client desktop devices because far less processing power is required. Of course, businesses need to first invest into server hardware which is capable of storing and streaming desired quantities of desktop environments. But it will save money in a long run by cheaper user devices. Desktop environments are stored on central server so they can be also centrally managed and controlled which increases security of whole system. The main Disadvantage is that streaming desktop environments to plenty of end users is very demanding on network infrastructure. Desktop virtualization is mostly used by companies with a lot of off-shore employees.

Storage virtualization

Multiple separated hardware storage devices (that can be on different physical locations) abstracted into single pool of virtualized storage space that act as a single storage device locally connected to the computer. Storage virtualization is used to ignore differences between individual storage devices and simplify using them. It is often used for back-ups and archives and can be implemented with software or hybrid software-hardware solutions. Common examples are:

1. DAS: (Direct Attached Storage) is a disk subsystem that is directly connected to a host rather than going through a switched network, thereby giving the host exclusive access to the disks.

2. NAS: (Network-attached storage) is server dedicated to managing storage space. NAS devices have to be part of LAN and they can be added or removed dynamically meaning that after adding/removing device whole system doesn't need to be restarted but it continues functioning as if nothing have happened.

3. SAN: (Storage area network) is basically a sub-network containing only storage devices. Storage space managed by SAN can be accessed by any server in LAN or WAN. When new storage devices is added it is immediately available to any server in network. In practical sense SAN enables anyone within network have access to network's whole storage space.

Chapter 3

Virtualization software

In this chapter we will introduce the following concrete virtualization implementations.

1. Xen Project
2. KVM
3. VirtualBox
4. UML
5. Docker Container
6. Wine

3.1 Xen Project [20]

It is a well known type I hypervisor (see 2.1), which means it runs directly on the host hardware. Xen Project is widely used as a base for a number of open-source and commercial applications providing server and desktop virtualization, infrastructure as a service (IaaS), security applications, embedded and hardware appliances and so on. Worlds biggest clouds today also run on Xen Project hypervisor base. All operating systems based on recent Linux kernel are capable of running Xen project and have packages containing hypervisor and basic tools.

Xen is managed by a special privileged virtual machine called Domain-0 or Dom0. Privileged means that it has device drivers and direct access to the physical hardware. Domain-0 is a specially modified Linux kernel which is started by Xen hypervisor during initial stage of system start-up. Its role is to manage and control every other unprivileged virtual machine (also called Domain-Us or DomU) that is running on the hypervisor. Domain-0 exposes control interface to the user and Xen project hypervisor cannot run without it. Through user interface in form of toolstack (or control stack), the user can create, delete or configure virtual machines. Toolstack can by a driven by command line console, a graphical interface or a cloud orchestration stack (for example OpenStack [12] or CloudStack [1]).

Originally Xen only supported paravirtualization (see 2.4.2). Support for Domain-U running in paravirtualized state is now included within upstream Linux kernel but support of Domain-0 is not included in the standard Linux. This means that it is easier to use Linux machine as a guest than as a host. Nowadays, Xen supports the new virtualization processor extension added to the x86 architecture, known as Xen Hardware Virtual Machine (HVM).

HVM allows unmodified guest operating system to be virtualized on Xen hypervisor. The support of hardware virtualizaton extensions (Intel VT, AMD-V) is needed. These processor extensions allow for many of the privileged kernel instructions to be handled directly by hardware using 'trap-and-emulate' technique (previously in paravirtualization converted to hypercalls).

Trap-and-emulate works as follows: Operating systems running on top of the hypervisor are run as user-level processes. They are not running at the same level of privilege as a Linux operating system that is running on bare metal. But if the operating system code is unchanged, it doesn't know that it does not have the privilege for doing certain things that it would do normally on bare metal hardware. In other words, when the operating system executes some privileged instructions, meaning they have to be in a privileged mode or kernel mode to run on bare metal in order to execute those instructions, those instructions will create a trap that goes into the hypervisor and the hypervisor will then emulate the intended functionality of the operating system. Problem in some architectures is that some privilege instructions may fail silently which means that you would think that the instruction actually succeeded, but it did not, and you may never recognize the failure.

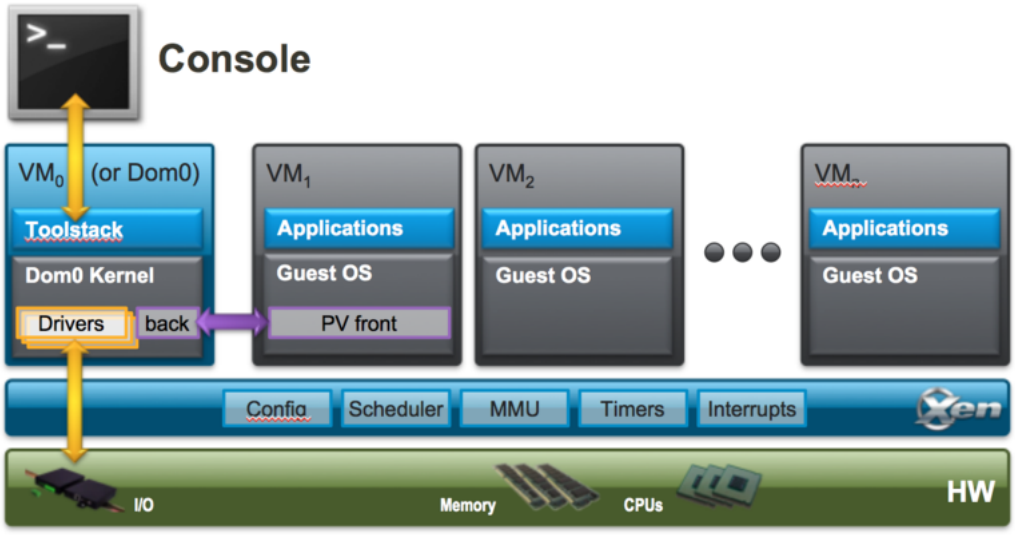

Figure 3.1: Xen architecture [20]

Here are the key features of Xen project hypervisor:

1. Minimal footprint: around 1 MB. Xen uses micro-kernel design which minimalizes memory footprint and interface to the guest and it is also more robust and secure than other types of hypervisors.

2. Driver isolation: hypervisor allows for the main device driver to run inside of a virtual machine. This is useful because if driver crashes it does not affect any other part of a system and virtual machine in which it runs can be rebooted and the driver restarted.

3. Support of many operating systems: although most installations use Linux as the domain-0, many other operating systems can by used such as NetBSD, OpenSolaris and others.

3.2 KVM [15]

KVM - Kernel-based Virtual Machine is a less known virtualization solution then Xen Project (see 3.1). It has host and guest support in an upstream Linux kernel released in early 2007. KVM is kernel module which when loaded turns host kernel into type I hypervisor. Running it requires Intel VT or AMD-V extensions enabled on a host system. By converting host machine kernel into hypervisor, KVM can take the advantage of already implemented components instead of implementing them from the scratch. For example it uses memory manager, scheduler, I/O stack, device drivers, security manager, network manager and others. In comparison to the Xen architecture which requires maintenance of both Xen hypervisor and Domain-0, KVM is loadable kernel module and is easier to patch and upgrade. From host's perspective, every virtual machine is a standard Linux process and it is treated as such.

Features:

1. Security: to improve security of virtual machines even further, KVM uses these approaches:
   (a) Security-enhanced Linux (SELinux) which establishes security boundaries around virtual machines.
   (b) Secure virtualization (sVirt) which boosts SELinux's capabilities and allow Mandatory Access Control (MAC) security to be applied.
2. Live migration: KVM supports live migration (see 6) of virtual machines.
3. Scheduling and resource control: every virtual machine is seen as a standard process which means that Linux scheduler allows the full control over resources allocated by it, and guarantees a quality of service. KVM offers the completely fair scheduler, control groups, network name spaces and real-time extensions.
4. Storage: KVM supports shared file system which means that virtual machines can be shared by multiple hosts. Disk images support thin provisioning. I.e. means that memory is allocated for virtual machine up to provisioned amount only when it needs it (Xen Project doesn't support thin provisioning, when virtual machine is provisioned for example 2 GB of RAM, after it starts, 2 GM of RAM are immediately allocated and can't be used elsewhere). This can lead to 'memory overcommit', state where more memory is assigned to virtual machines than is actually available on the system. KVM deals with memory overcommit in various ways:
   (a) Host can choose memory pages and write them to the disk. This leads to reducing performance because when virtual machine wants to access memory, host needs to read it from the disk which is significantly slower than RAM memory.
   (b) With VirtIO drivers, host can request virtual machines to shrink their cache memory in order to free as much space as needed. This is called 'ballooning' and requires cooperation among host and guests.
   (c) KSM (Kernel Samepage Merging) is a process of merging identical memory pages from multiple virtual machines into a single read-only memory chunk while removing all duplicates of it. If any guest needs to write into one of merged pages, host creates writable copy which guest can modify.

5. Lower latency: kernel divides processes with long computing times into smaller pieces which are scheduled and processes accordingly.

3.3 VirtualBox [17]

VirtualBox is a type II hypervisor (see 2.1) currently being developed by Oracle Corporation [9]. Oracle VM VirtualBox runs on Microsoft Windows, Mac OS X, Linux and Oracle Solaris systems and supports wide range of guest operating systems. With thousands of downloads each day it is the most popular cross-platform open-source virtualization solution.

Here are some of VirtualBox's main features:

1. Portability: VirtualBox has to run on an host operating system but its functionality is to a very large degree identical on all of them, same files and image formats are used. This allow us to create a virtual machine on one host and run it on another host with different operating system. Virtual machines can be imported and exported using an Open Virtualization Format (OVP) with which we can import virtual machines created with different virtualization software.

2. No hardware virtualization needed: VirtualBox doesn't require a processor support like Intel VT or AMD-V so it can be run even on older hardware not possessing those features.

3. Guest additions: guest additions are software packages that can be installed inside of virtual machines to improve their performance or improve their integration with host system. They consist of device drivers and system applications, for example on of guest additions is 'Shared folders addition' which provides an easy way to exchange files between host and guest. We can create a folder on host system and share it to the guest.

4. Hardware support:

   (a) Guest multiprocessing: VirtualBox can present up to 32 virtual CPUs to every virtual machine regardless of how many CPUs are present in host system.

   (b) USB device support: virtual USB controller allows to connect USB device to virtual machine without a need to install device-specific drivers to the host machine.

   (c) ACPI support: Advanced Configuration and Power Interface is an open standard that operating systems can use to configure hardware components and to perform power management and status monitoring.

   (d) Built-in iSCSI support: this allows us to connect from virtual machine directly to the iSCSI storage server without going through host system which highly reduces overhead.

   (e) PXE support: Preboot eXecution Environment (PXE) in short is a way to boot operating system from a server on a virtual machine. Advantages are obvious, we don't need to have a operating system on a hard drive connected to the virtual machine, we just need to connect to server and boot it from there.

5. Snapshots: we can create snapshots of the current state of a virtual machine and store it. When needed we can reverse current state of VM and load configuration from any

snapshot. This way we can periodically save backups for quick recovery in case of emergency.

6. Grouping: multiple virtual machines can be collected into a group. We can than perform same operations over the group as we can over individual virtual machines (for example start, pause, shutdown, close, ...). By using groups we can manage multiple virtual machines with same configuration, purpose, etc at the same time as well as we can still manage individual VMs that are part of a group. One virtual machine can be inside multiple groups and groups can be nested into hierarchy.

7. Remote machine display: VRDE - VirtualBox Remote Desktop Extension allows for a high-performance remote access to any running virtual machine.

3.4 UML [6]

User Mode Linux (UML) allows us to run Linux kernels as user mode processes under a host Linux kernel thus allowing us to run multiple independent virtual machines. Main difference between UML and other virtualization technologies is that UML is more of a virtual OS than virtual machine. Other solutions like from VMWare are real virtual machines in that they emulate physical hardware and any operating system that runs on physical platform can also run on emulated one. Advantage of this solution is that guest OS is host OS-independent, meaning that any OS able to run on hardware is able to run on top of VMWare. On the other hand, UML is basically just Linux kernel modified to run in user space, UML guest can run only on Linux platform which is serious limitation but being more of virtual OS has other advantages. Solutions such as Xen, BSD jail or Solaris zones are integrated into host operating system but UML runs as a process. This has some performance costs but gives UML host OS version independence. UML has many real-world uses but it's most popular use-case is kernel development and debugging as it was its original purpose, for that can be used normal process-level tools like gdb, gporf (profiling) or gcov (coverage testing). Another popular uses are driver development, safe kernel testing and education due its simpler nature than other solutions.

3.5 Docker Container [3]

Docker container is a operating system level technology (see 2.4.3) established and promoted by Docker Inc. Docker uses a technology called namespaces. When we run a container, Docker creates a set of namespaces for it. These namespaces provide a layer of isolation. Each aspect of a container runs in a separate namespace and its access is limited to that namespace. On Linux, Docker Engine uses these namespaces:

1. pid: process isolation

2. net: managing network interfaces

3. ipc: managing access to IPC resources

4. mnt: managing file system mount points

5. uts: isolating kernel and version identifiers

Docker container is operated by command-line tool called the Docker client which can run on the container host or through a remote interface connected to the container host. The main task of a Docker client is to pull images of containers from registry. Registry can be public or private and it is a repository of sources of 'ready to run' virtual workloads. Main public registry is Docker Hub which is operated by Docker Inc., but nowadays there are plenty of others. We can pull a container image using Docker daemon and from that image we can build working model for that container. A container is launched by running an image. An image is an executable package that includes everything needed to run an application: the code, a runtime libraries, environment variables, and configuration files. Images that are mostly the same, except for the last few steps, can reduce disk usage by sharing parent layers. A container is a runtime instance of an image - what the image becomes in memory when executed (that is, an image with state, or a user process). Image can also include directives for daemon to preload the container with other components prior to running or directives for the local command line after the local container image is build. The model of images and registries created standardized ways to build, load and manage containerized applications. Docker has been very successful in building a large open-source community which has contributed to the rising number of images in public and private repositories which attracts even more developers and enlarges open-source community. Docker image is defined by text-based Dockerfile which specifies a vase operating system image to start from, commands to prepare/build the image and commands to call when image is 'run'. (Docker runs multiple containerized workloads on the same OS. By using containers, only the programs and their immediate dependencies are hosted by containers, with critical resources provided by the underlying operating system. This means that containerized systems can load applications faster and consume less resources.). Docker is available on many different operation systems including most modern Linux distributions, Mac OSX and Windows.

OCI: The Open Container Initiative (OCI) [16] is a lightweight, open governance structure (project), formed under the auspices of the Linux Foundation, for the express purpose of creating open industry standards around container formats and runtime. The OCI was launched on June 22nd 2015 by Docker, CoreOS and other leaders in the container industry. The OCI currently contains two specifications: the Runtime Specification (runtime-spec) and the Image Specification (image-spec). The Runtime Specification outlines how to run a „file system bundle“ that is unpacked on disk. [tiez]

Docker Engine is a client-server application with these major components:

1. A server which is a type of long-running program called a daemon process
2. A REST API which specifies interfaces that programs can use to talk to the daemon
3. A command line interface (CLI) client

Docker uses a client-server architecture. The Docker client talks to the Docker daemon, which does the heavy lifting of building, running, and distributing Docker containers. The Docker client and daemon can run on the same system, or you can connect a Docker client to a remote Docker daemon. The Docker client and daemon communicate using a REST API. A Docker registry stores Docker images. Docker Hub and Docker Cloud are public registries and Docker is configured to look for images on Docker Hub by default. We can run our own private register. We can also buy or sell Docker images or distribute them for free in Docker store.

3.6 Wine [19]

Wine - Wine Is Not an Emulator acronym means that Wine is not a virtual machine, it does not emulate physical hardware and we are not supposed to install Windows or any Windows driver on top of it. Different software programs are designed for different operating systems and are generally not compatible with other operating systems, for example Windows programs can't run on Linux system because they use kernel calls that Linux system doesn't understand thus cannot interpret them. This is the main motivation for Wine. Wine is an implementation of Windows API and can be used as a library to port Windows applications to Unix, basically acting as a bridge between the two. It is a compatibility layer, every time a Windows program tries to perform an action that Linux doesn't recognize, Wine will translate it into one that it does. Wine can also recompile Windows program source code into format understandable for Linux. Even in recompiled form, Wine is still needed to run the program but there are many performance and other advantages to this process. Wine is an open source project.

Features: Wine is constantly growing in the features it supports, here are some of them:

1. Support for running, Win64, Win32, Win16 and Dos programs.
2. Optional use of external vendor DLLs.
3. MacOS and Android graphics support.
4. DirectX for games.
5. Support for alternative input devices such as graphics tablets
6. Winsock TCP/IP networking support.
7. Advances Unicode and foreign language support.
8. Fill featured Wine debugger and configuratable trace logging messages for easier troubleshooting.

Executables: Wine's main task is to run Windows executables, here are supported types:

1. DOS executable: very old programs for MS-DOS.
2. Windows NE executable (NE - New Executable): They were the native processes run by Windows 2.x and 3.x.
3. Windows PE executable (PE - Portable Executable): Introduced by Windows 95 and became the standard format for all later Windows versions. Portable Executable means that format of the executable is independent of the CPU, even if the code IS dependent of the CPU.
4. Winelib executable: applications written by using the Windows API but compiled as a Unix executable.

Wine architecture is close to the Windows NT architecture but several subsystems are not implemented yet.

Chapter 4

Products for virtualization

Even though virtualization technology was introduced more than 20 years ago it experienced a big boom only in past few years. There are now many companies developing some sort of virtualization technology and offering different products. This is great for customers which can choose from wide variety of virtualization tools depending on type/price/scale and so on. Every year more and more companies decide to virtualize their servers for various reasons. In this chapter we will look at biggest or fastest growing virtualization companies today and present some of their products

1. VMware
2. Citrix
3. Oracle
4. Microsoft
5. Red Hat

4.1 VMware [18]

4.1.1 VMware Workstation

Is basically industry standard for desktop virtualization - it is a hosted hypervisor (see 2.1). Workstation runs on Windows or Linux and can also run virtual machines with Windows or Linux operating system. Initially launched in 1999 is one of the longest running modern day virtualization applications. VMware Workstation Pro can save the state of a virtual machine as a snapshot which can later be restored, effectively returning the virtual machine to the saved state. It can also group multiple virtual machines into an inventory folder with which we can control (power on/off) all the virtual machines inside it at once. Key features:

1. High-performance 3d graphics: Workstation supports DirectX 10 and OpenGL 3.3 to run the most demanding 3D applications with near-native performance in Windovs virtual machines.
2. Massive virtual machines: with Workstation we can create virtual machines up to 16 CPUs, 8 TB virtual disks and 64 GB of memory to run high-performance demanding desktop and server applications in virtualized environments. To improve graphics performance we can allocate up to 2 GB of available host video memory.

3. High resolution display support: Workstation supports high-resolution display for desktops and also multiple displays.

4. Restricted access to virtual machines: virtual machine can be encrypted and protected by password.

5. Cross compatibility: we can create virtual machines that can run across the VMware product portfolio or even import VMs from another vendors. Virtual machines are transfered in Open Virtualization Format (OVF). OVF specifications provides a means of describing the properties of a virtual system. It is XML based and is used to describe a single virtual machine or a virtual appliance.

4.1.2 Workstation Player

Workstation Player is 'lighter' version of Workstation. It runs on the same core as Workstation Pro and vSphere (see 4.1.3) but delivers less features. On the other hand it is free of charge for personal non-commercial use. Workstation Player can still create and run virtual machines, but its functionality is limited as follows:

1. It can run only one virtual machine at a time instead of multiple virtual machines simultaneously.

2. It cannot create or manage encrypted VMs.

Workstation Player is a great tool for 'feeling up' virtualization before investing into full blown solutions or for a simple personal use.

4.1.3 VMware vSphere

Serves as a complete platform for implementing and managing virtual machine infrastructure on a large scale. vSphere consists of two core components: ESXi and vCenter Server.

1. ESXi is a bare-metal (type I) hypervisor. It has small footprint (150 MB) which minimizes security threats to the hypervisor. It can support up to 128 virtual CPUs, 6 TB of RAM and 120 devices and offers built-in UI based on HTML5 standards, vSphere command line interface and REST-based APIs. Thin architecture enables it to install in 10 minutes and boot 2 to 3 minutes.

2. vCenter Server: provides a centralized platform for managing VMware vSphere enironment, allowing automation and delivery of virtual infrastructure. From a single vCenter instance can be managed up to 1000 hosts and 10000 virtual machines. Moreover, these numbers can be scaled by using Linked Mode for managing up to 30000 virtual machines across 10 vCenter Server instances and vSphere for even more hosts and VMS. Linked Mode replicates roles, permissions and licenses across infrastructure so we can log in and view inventories of all vCenter Servers. We can also use 3rd-party plug-ins created by VMware partners. With open Web Client Plug-in SDK, vCenter Server has largest partner ecosystem in industry which means that we can implement back-up, data protection, server, network and security management directly from vCenter. Web Client Plug-in Certification Program can help to bring a better end-user experience in various ways. vCenter Appliances can be backed up to a file via industry standard protocols.

4.1.4 VMware ThinApp

VMware ThinApp is VMware's solution for application virtualization (see 2.4.4). ThinApp uses a build process to package application files and registry settings into a single application container that can be executed on a variety of operating systems without installation. Applications can be executed from a user's desktop, a network path, or removable media. Applications run entirely in user mode under the security context of the currently logged in user. To virtualize an application ThinApp uses a Capture and Build process which consists of two components:

1. Setup Capture: The Setup Capture wizard guides the process of capturing the application and applying administrator-supplied configurations specific to the package. Setup Capture takes a prescan snapshot, allows the administrator to install and configure the application, and takes a postscan snapshot. The difference between the prescan and postscan snapshots, which represents the application, is placed in the project directory.

2. The process by which the project directories and configuration settings are compressed and embedded into the package.

4.2 Citrix [2]

Citrix Systems, Inc. is an American multinational software company that provides server, application and desktop virtualization, networking, software as a service (SaaS), and cloud computing technologies. The company began by developing remote access products for Microsoft operating systems [//]. It licensed source code from Microsoft and has been in partnership with the company throughout its history. Citrix came to prominence in the 1990s as a leader in thin client technology, and its goal was to built a tool for accessing remote servers. The company had its first initial public offering in 1995 and, with few competitors, experienced large revenue increases between 1995 and 1999. Between 2005 and 2012, Citrix acquired more than a dozen other companies, allowing it to expand into server and desktop virtualization, as well as others.

4.2.1 XenServer

XenServer a the complete server virtualization platform from Citrix. The XenServer package contains everything needed to create and manage a deployment of virtual x86 computers running on Xen (see 3.1), the open-source paravirtualizing hypervisor with near-native performance. XenServer is optimized for both Windows and Linux virtual servers. XenServer runs directly on server hardware without requiring an underlying operating system (type I hypervisor - see 2.1). There are two methods by which to administer XenServer: XenCenter and the XenServer Command-Line Interface (CLI). XenCenter is a graphical, Windows-based user interface. XenCenter allows you to manage XenServer hosts, pools and shared storage, and to deploy, manage and monitor VMs from your Windows desktop machine. Key features include creating virtual machine templates from snapshots, XenMotion which allows us to live migrate VMs between hosts. XenServer 7 provides support for high-performance enhanced 3D graphics, with the widest variety of GPU pass-through and virtualized GPU vendor options. Nowadays, only XenServer includes support for Intel's Virtual Graphics Technology (GVT-g), a CPU embedded GPU with no extra hardware

required to facilitate enhanced graphics workloads. Starting with Citrix XenServer 7, the Enterprise version includes Direct Inspect APIs which allow third-party vendors to secure the OS.

4.2.2 XenDesktop

XenDesktop is a desktop virtualization software (see 2.4.4) platform developed and sold by Citrix Systems, that allows multiple users to access and operate Microsoft Windows desktops. These desktops are installed at a centralized server separate from the devices from which they are accessed. Citrix XenDesktop contains a profile management tool that manages profiles independently from the OS and delivers required profile data on-demand. XenDesktop also includes Citrix Receiver, a universal client built for virtually any device including Windows, Mac, Linux, iOS, Android, Chrome OS or Blackberry. XenDesktop includes a Windows app store delivered by Citrix StoreFront to provide a single aggregation point for all IT user services.

4.2.3 XenApp

Citrix XenApp is application virtualization software (see 2.4.4) that allows Windows applications access via individual devices from a shared server or a Cloud system. We can have applications installed on a XenApp server in a data center and launch them on any device (laptop/tablet/etc.). Application can be accessed from range of operating systems but streamed applications have to run on Windows.

4.3 Oracle [9]

Oracle Corporation is an American computer technology corporation headquartered in California. The primary specialization is developing and marketing database software and technology, cloud engineered systems and enterprise software products.

4.3.1 Oracle VM

The Oracle VM is a platform that provides a fully equipped environment with all the latest benefits of virtualization technology, it enables us to deploy operating systems and application software within a supported virtualization environment. Architecture of Oracle VM is composed of these parts:

1. Client Application: Oracle VM provides various user interfaces, we can use either CLI (command line interface) through SSH client, GUI accessible over web-browser or external applications such as Oracle Enterprise Manager, custom built applications or scripts using the Web Services API. All communication with Oracle VM Manager is secured using either a key or certificate based technology.

2. Oracle VM Manager: used to manage Oracle VM Server, virtual machines and resources and is composed of mutiple subcomponents functioning as one unit. It usually runs on standalone computer but can also run as a virtual machine however this method is restricted and not fully supported. Core application is Oracle WebLogic application running on Oracle Linux. Oracle VM Manager communicates with each Oracle VM Server via the Oracle VM Agent, using XML-RPC (Remote Procedure

Call using XML) over HTTPS. The Oracle VM Agent on each Oracle VM Server is equally able to send notifications, statistics and event information back. Even though Oracle VM Manager is critical component for configuration in Oracle VM infrastructure, the virtualized environment can run correctly even if Oracle VM Manager experiences downtime, still maintaining high availability and able to live migrate virtual machines on x86 platform.

3. Oracle VM Manager Database: Used by the Oracle VM Manager core application to store and track configuration, status changes and events. . Oracle VM Manager uses a MySQL Enterprise database that runs on the same host as Oracle VM Manager. The database is used exclusively by Oracle VM Manager and cannot be used by any other application. It is automatically and periodically backed-up and can be also backed-up manually.

4. Oracle VM Server: a virtualization environment providing a lightweight and secure server platform for running virtual machines. At least one Oracle VM Server is required, but to take advantage of clustering several are needed. Oracle VM Server is installed on a bare metal computer, and contains the Oracle VM Agent to manage communication with Oracle VM Manager. On x86-based systems, Oracle VM Server is based on an updated version of Xen hypervisor technology. In case of SPARC, Oracle VM Server takes advantage of the hypervisor that is already included within SPARC firmware. Groups of Oracle VM Servers are usually clustered together to create server pools. This allows Oracle VM Manager to handle load balancing. Virtual machines running within a pool can be easily moved between servers in that pool. In Oracle VM infrastructure server pools are required even if they consist of only one server.

5. External Shared Storage: Provides storage for a variety of purposes and is required to enable high-availability options afforded through clustering.

4.4 Microsoft [8]

Microsoft Corporation is an American technology company with headquarters in Washington.

4.4.1 Hyper-V

Hyper-V is Microsoft's hardware virtualization product. It is a type I hypervisor (see 2.1) based virtualization technology for certain x64 versions of Windows. It supports isolation in form of partition. Partition is a logical unit of isolation in which operating system runs. Hyper-V consists of these parts:

1. Parent partition: The microsoft hypervisor requires at least one root (parent) partition running Windows. The parent partition and has direct access to hardware devices and the virtualization management stack runs on it. The root partition creates the child partitions, assigns hardware devices except for processor scheduling and physical memory allocation which are handled by the hypervisor, parent also handles power management, plug and play operations and logging of any hardware occurs. It is the first partition created when hypervisor is started. The virtualization components

hosted in the parent partition are referred to collectively as the virtualization stack and these are its components:

   (a) Virtual Machine Management Service
   (b) Virtual Machine Worker Process
   (c) Virtual Devices
   (d) Virtualization Infrastructure Driver
   (e) Windows Hypervisor Interface Library
   (f) Virtualization Service Providers
   (g) Virtual Machine Bus

2. Child partitions: partitions which host the guest operating systems and its applications. Hyper-V supports many different guest operating systems including various releases of Linux, FreeBSD, and Windows.

Virtual Devices can also take advantage of a Windows Server Virtualization feature, named Enlightened I/O, for storage, networking, graphics, and input subsystems. Enlightened I/O is a specialized virtualization-aware implementation of high level communication protocols (such as SCSI) that utilize the VMBus directly. This means that VMBus can bypass any layer of device emulation thus making communication more efficient. On the other hand, this requires guests to be aware of hypervisor and VMBus. Hyper-V integration services provide Hyper-V enlightened I/O and a hypervisor aware kernel. Hyper-V requires a processor supporting hardware supported virtualization (Intel VT or AMD-V). Some of Hyper-V features:

1. Disaster recovery and backup: Hyper-V Replica creates copies of virtual machines which are then stored in another physical location and can be used to restore virtual machines. That is for disaster recovery, for backing-up we can use 2 approaches. First uses saved states of virtual machines. Second uses Volume Shadow Copy Service (VSS) which captures and copies stable images for back-up on running systems (particularly servers) without degrading performance and stability of the provided services. VSS enables creating services that can be backed-up by any vendor's back-up application using VSS.

2. Optimization: Each supported guest operating system has a customized set of services and drivers, called integration services, that make it easier to use the operating system in a Hyper-V virtual machine.

3. Portability: Hyper-V supports live migration, storage migration, import/export of virtual machines and other features.

4. Remote connectivity: Virtual Machine Connection is a remote connection tool for both Windows and Linux, it gives us console access to the guest even if operating system isn't booted yet.

5. Security: Hyper-V supports secure boot and shielded virtual machines.

6. Network Load Balancing: Hyper-V includes virtual switch capabilities that provide the ability to use the Windows Network Load Balancing (NLB) service to load-balance across virtual machines running on different servers.

4.5 Red Hat [13]

Red Hat is the world's leading provider of open source software solutions, using a community-powered approach to provide reliable and high-performing cloud, Linux, middle-ware, storage, and virtualization technologies.

4.5.1 Red Hat Enterprise Virtualization [14]

Red Hat Virtualization is a complete virtualization infrastructure solution for virtualized servers and workstations. Red Hat Enterprise Virtualization is fully open source software, based on the Kernel-based Virtual Machine (KVM see 3.2) hypervisor technology. It is composed of two main components:

Red Hat Enterprise Virtualization Manager (RHEV-M)

Has a centralized management system allowing system administrators to view and manage virtual machines and images. It provides a graphical user interface to manage the physical and logical resources within the virtual environment infrastructure. RHEV-M is built on the top of Red Hat Enterprise Linux and Red Hat JBoss Enterprise Application Platform (EAP) which is a Java based application server which provides a framework to support efficient development and delivery of cross-platform Java applications. RHEV-M also offers an advanced system dashboard for a deep resource utilization overview of CPU, memory, and storage host resources. We can operate it even using an open source, community-driven RESTful API. RHEV-M can be used to migrate workloads from other platforms to RHEV. It includes a data warehouse that collects monitoring data for hosts, virtual machines, and storage too. We can analyze our environments and create reports using any SQL supporting query tools.

Red Hat Enterprise Virtualization Hypervisor (RHEV-H)

The Red Hat Enterprise Virtualization Hypervisor is 64-bit bare metal hypervisor combining the advanced features of the KVM hypervisor with subset of packages from Red Hat Enterprise Linux 5 including only the components required to boot, run a management agent and manage virtual machines. The footprint is less than 100MB and it is delivered as a single image that can run on any Red Hat Enterprise Linux 5 certified hardware platform. Host is scalable up to 288 logical CPUs and 12TB and guest up to 240 vCPU and 6TB vRAM per virtual machine. RHEV-H uses Security-Enhanced Linux (SELinux) a joint project developed by the United States National Security Agency (NSA) and the Linux community to provide a security hardened operating environment with high levels of security, resource isolation and auditing, to present even smaller attack surface it does not support running applications other than the management agent and virtual machines, thus reducing maintenance and patching. The base file system image for RHEV Hypervisor is stateless, running only in memory, preventing changes to the base image.

Architecture is build around two main components:

1. KVM: (see 3.2).
2. QEMU: A multi-platform emulator used to provide full system emulation.

Features:

1. Automation and integration:
   (a) Red Hat CloudForms: enables automation of virtual events and provides reporting, chargeback and compliance enforcement.
   (b) OpenStack Glance and Neuron: to ease traditional workload migration to private clouds or to design applications that span virtual and private cloud environments.
   (c) Red Hat Ansible Automation: allows us to create/remove/update or powermanage virtual machines.
   (d) Simple Network Management Protocol (SNMP) allows integration of third-party monitoring systems.
2. Advanced storage performance and scale: Storage Pool Manager (SPM) can delegate storage operations to other data center hosts to improving performance. Block discard support allows reclaiming of storage space from the virtual machine and after deleting a disk from a storage domain.
3. Cross-platform support: full support is provided for Red Hat Enterprise Linux 5, 6 (32- and 64-bit), and 7 (64-bit). Support is also available for Windows Server 2008, 2008 R2, 2012 (32- and 64-bit), and 2016 and desktop systems Windows 7 and 10 (32- and 64-bit). Vendor support is provided for SUSE Linux Enterprise Server 10, 11, and 12.

4.5.2 oVirt [10]

oVirt is a complete virtualization management platform, licensed and developed as open source software. It is used to manage hardware nodes, storage and network resources, and to deploy and monitor virtual machines running in a data center. oVirt serves as upstream version of RHEV (see 4.5.1). While RHEV is a little less advanced then oVirt, it is more stable and better tested.

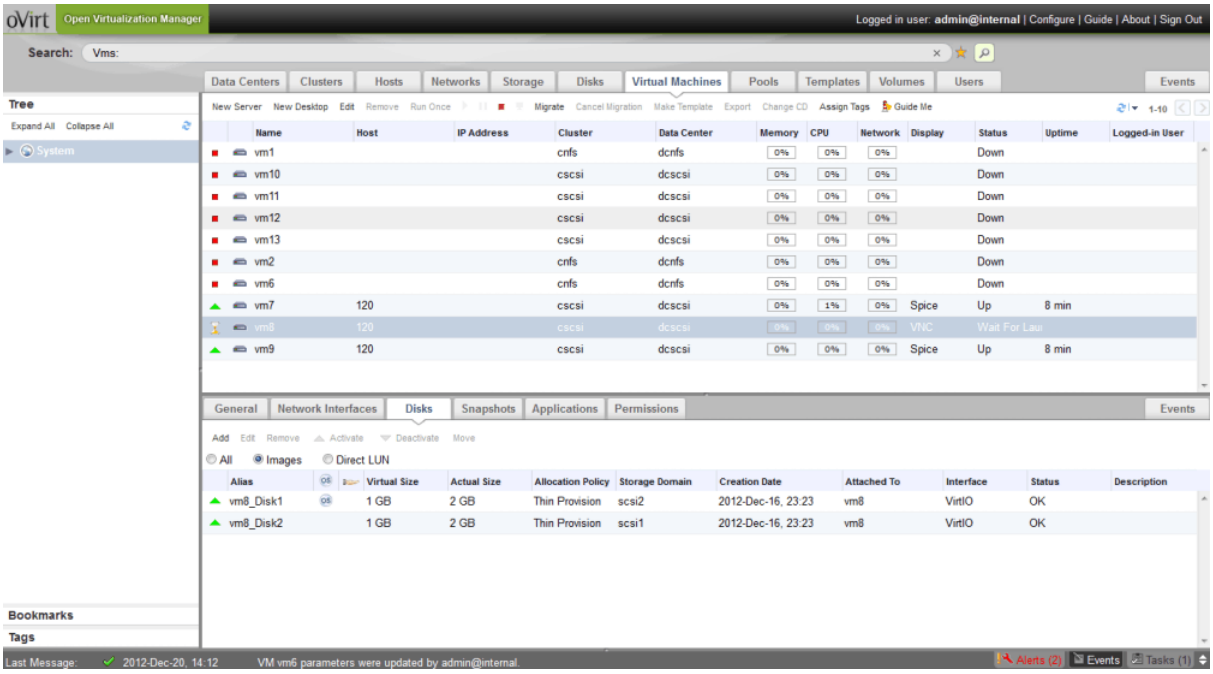

Figure 4.1: oVirt

oVirt architecture consists of three main components:

1. oVirt Engine: Centralized virtualization management engine with graphical administration console and programming interfaces. It is a JBoss-based Java application which runs as a web service. With oVirt Engine we can deploy, monitor, move, stop and create VM images, configure storage and network, etc. oVirt Engine provides two types of portals:

   (a) Admin Portal: provides a graphical management system for administrators to manage virtual machines, templates, desktops, storage, clusters, and datacenters.

   (b) User Portal: provides standard and power user access to the oVirt environment.

   (c) oVirt API: all platform commands via an open source, community-driven RESTful API.

   (d) Software Development Kit: provides an Python environment to support the development of custom software utilizing the APIs exposed by the engine.

2. host (node) on which we can run virtual machines. We can use multiple hosts.

3. storage nodes, which hold the images and ISOs corresponding to those virtual machines. Storage nodes can use block or file storage, and can be local or remote, accessed via NFS. They are grouped into storage pools, which can ensure high availability and redundancy.

The nodes are Linux distributions with VDSM and libvirt installed, along with some extra packages to enable virtualization of networking and other system services. The supported Linux distributions are Fedora 17 or oVirt-node. oVirt Node is an image-based, small footprint hypervisor on which we run virtual machines, it is basically a stripped-down distribution containing just enough components to allow virtualization. oVirt Node provides text-based GUI for manageability and easy installation.

Chapter 5

Problem and proposed solution

5.1 Disadvantages of Web Admin

oVirt's admin portal is called Web Admin and it is web based UI application running on the top of the engine. Sysadmins use it to perform advanced actions. It is powerful tool for managing oVirt environment but gaining some types of information is, according to the user feedback, unpractical and unnecessarily complicated.

Here are just few examples gathered form oVirt administrators. These kinds of data should be easier and faster to acquire:

1. which hosts contain unupgraded virtual machines?
   (a) Go to 'Compute' -> 'Hosts' tab
   (b) Choose specific host
   (c) Go to subtab 'Virtual Machines' to see which VMs run on that host
   (d) Go to 'Compute' -> 'Virtual Machines' tab
   (e) Choose specific virtual machine
   (f) In 'General' tab see compatibility version This has to be done for every virtual machine and every host.
2. which storage domains does particular virtual machine use?
   (a) Go to 'Storage' -> 'Domains' tab'
   (b) Go to 'Virtual Machines' subtab to see which VMs run on this storage domain. Remember or write down if virtual machine we are interested in is there and than do the same for every other storage domain.
3. which virtual machine highest pressure on the network?
   (a) Go to 'Compute' -> 'Virtual Machines' tab
   (b) Go to 'Network Interfaces' subtab
   (c) Open 'Statistics' sub-subtab. Then do this for every virtual machine and remember which was using the virtual network interface the most.

As we can see in examples above, if we want to get certain info about all entities (hosts, virtual machines, storage domains, networks, ...) we need to go through every entity's detail. In small environments, this isn't a major problem. But in larger ones, this becomes a very time consuming process.

5.2 Proposed solution

My solution to mentioned problem is a desktop application providing all the relevant information about all entities that system administrators work with. I have collaborated with several admins to design table showing all the needed data.

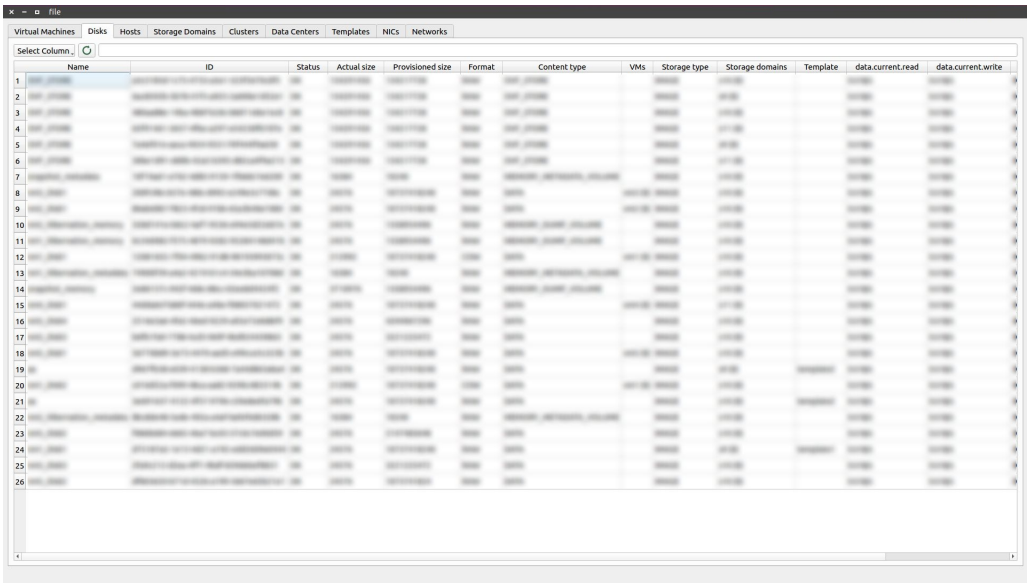

Figure 5.1: My solution

Key features:

1. hiding and showing columns
2. filtering
3. sorting columns
4. export table to .csv file
5. multiple columns are linked to other columns in different tabs for easy access.

Chapter 6

Implementation

Project is implemented in programming language Python version 3.6 and PyQT5 and referential environment is operating system Fedora version 27.

In this chapter, entity of virtual environment, e.g. virtual machine, disk, storage domain, host, and so on will e referred as entity. Likewise, my application will be referred only as application.

6.1 Back-end

Application uses Python-SDK tool to extract all the data from the oVirt API. The oVirt Python-SDK is an automatically generated software development kit for the oVirt engine API. This allows us to develop Python-based applications for automating a variety of complex administrative tasks in oVirt.

Application flow:

1. Application searches for config file.
2. After putting in login information and FQDN (fully qualified domain name) of a target virtual machine, connection is established. Connection is kept during whole runtime of application.
3. Low-level objects extract all the data needed from oVirt API.
4. High-level objects process raw data from low-level objects and form them into data structures containing refined data for all entities.
5. Tab-level objects wrap high-level objects interface and process data structures into form containing everything needed to be displayed. This includes adding row and column flags, adding filter restrictions, creating data structure for front end elements and others.
6. Tab-level objects are displayed to the user by the front-end module.

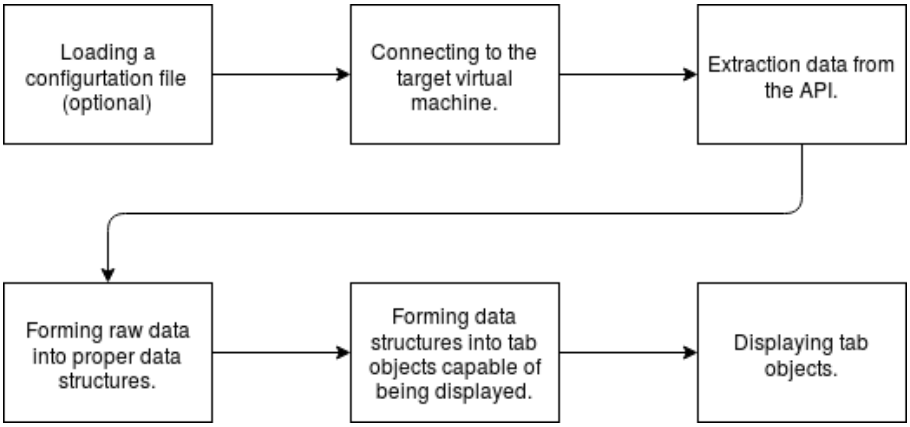

Figure 6.1: Application flow.

Division of flow into hierarchy of different level modules is very important. oVirt is a live and very quickly developing project. In the future, it is certain that some groups of data supported by this application will become obsolete and not needed anymore and on the other hand, there will come the demand for a new data. Whole application architecture is build with that in mind. To add or to remove column from entity, we only need to modify low-level class of given entity in trivial manner. To remove or add entirely new entity into the table, we only need to create low-level class of that entity which is simple, because if strictly follows the base low-level class. After that we create high-level class of that entity in which we only set filter restrictions, switch expressing if we want to include statistics and other attributes that cannot be set automatically. In the tab-level class we set links redirection and that is all, new entity is created, added and shown in the table. Application is very easily customizable in terms of entities it supports and data shown in each of them which will be valuable in the long run.

6.1.1 Supported features and functionality

Refresh: refresh functionality loads up to date data from API. As mentioned, connection is kept during whole run time of application. This is due to refresh functionality. If we close a connection after initial extraction of data, we would have to establish new connection every time we would like to refresh any tab which would considerably slow down process of getting new data. After losing the connection, application keeps on functioning properly but isn't able to update. Application doesn't support automatic refresh, it needs to be evoked by clicking 'refresh button'. This is due to demand of administrators I collaborated with. While investigating some failure they don't have an exclusive access to the failed virtual machine. Someone else can, for safety's sake, turn it off, restart it, delete it or something else and in that case, admins will lose the data they were working with. Automatic refresh is also very taxing and slow operation. In an environment with tens or hundreds of virtual machines, disks, etc. loading new set of data every couple of seconds would greatly slow down the application and would put unnecessary pressure on the network connection. It is also not necessary, if admins require latest data they can simply press the refresh button.

Hiding and showing columns: entities have very many attributes and working in very large tables is impractical mainly because of horizontal scrolling. To make it easier, we can select only some currently relevant columns to be shown while other will be hidden.

This configuration can be saved into 'config' file which will be loaded upon next start of the application so we don't have to always set it manually. If needed, we can still filter by hidden columns.

Filtering: resolving filters is done by simple search engine. It supports filtering by multiple filters at the same time. Filters are in AND relationship (OR was not required by admins as there is no real use case in admin's work for it). Filters are case insensitive, and consist of these three parts: 'searched column' 'operator' 'value'. 'Searched column' is name of the column we want to filter by. 'Operator' can be '=' (equals), '>' (greater than) or '<' (less than). Columns are divided into two categories: columns containing string data (names of entities and various variables) and columns containing numeric data (values). 'Value' is value we want to filter by, it supports all Python regular expressions. Some cells contain not only one value but list of multiple values while showing only the first one and in brackets number of others. Example: one virtual machine 'vm1' uses three disks: 'Disk1', 'Disk2' and 'Disk3'. In cell is shown 'Disk1 (2)'. We can filter these kinds of cells by string regular expression or by number. If we filter them by number (disks = 2) result will be all virtual machines with 2 disks. In this manner we can similarly use '>' and '<'. If we use filter 'disks = Disk2' result will be virtual machine 'vm1' and cell will now show 'Disk2 (2)'. Columns containing numeric data are simply filtered by value. We don't have to include units in the filter, as default, engine will filter by the column unit. This significantly simplifies some of admins work tasks.

Flags: application uses large set of data. To manipulate with it as little as necessary two sets of flags are used: row flags, and column flags. Filtering and hiding columns doesn't affect data, it only changes row and column flags accordingly and when printed for the user, application only prints data marked as printable by these flags. This way we only need one table of data instead of two: 'all data table' and 'currently shown data table' or even worse, constantly deleting unfit data and loading them back from the API. Not only it noticeably reduces the amount of memory needed but it increases performance as well. Flags can be stored in the 'config' file and loaded upon next start of the application so we don't have to adjust it every time and can make it more personalized.

Export: exporting to .csv files: this is great tool for archiving data. We can choose which tables we want co save, filters and hidden columns also affect exported .csv file.

Links between columns: some entities are in 'many to many' relationships. For example, one virtual machine can use multiple disks and similarly, one disk can contain multiple virtual machines. This relationship is hard to express in simple text table. Solution works like this: in 'Virtual machines' tab, only first disk used by any particular virtual machine is shown. In same cell there is number in brackets, this number represents number of other disks that virtual machine uses. To make it apparent, cells containing links to other tabs have light blue color. If we double click on that cell we are redirected into 'Disks' tab with only disks with virtual machine which we were redirected from shown. This way we can very simply see for example all the statistics of disks that one virtual machine is running on.

Insecure mode: communication between application and API is not additionally secured. Added security layers are not necessary as in order to connect to the API we have

to be connected into Red Hat's private virtual network (VPN) in the first place, so the communication is already secure.

Ordering: every column can be ordered in ascending or descending order which is indicated by small arrow icon in the header of the column. Ordering and filtering use special 'custom comparison' mechanism. Table is structured in a way that cells can contain strings, integer numbers, floating point numbers, numbers with various units or without one, lists of multiple data or a special value 'None' (which is Python equivalent of 'null'). 'Custom comparison' mechanism uses operands '=', '<', '>' and can compare strings, numbers in diverse forms, lists and 'None' with each other. One limitation is that list can't be used with operators '<' and '>' which wouldn't make sense in the context of given application, they can only use '='. Here are some examples:

1. 'vm2' == ['vm1', 'vm2', 'vm3'] -> True (and 'vm2' will switch positions with 'vm1')
2. 'vm1' > None -> True
3. 10.2 MB < 5.245 MB -> False

6.2 Front-end

Front end part of application uses PyQT5 module which is Python wrapper around C++ library QT5. Main focus is on simplicity and clearness of the user interface. In the spotlight is the most important part of the whole application - the table of data. Whole idea of the application is to be an effective and well functioning working tool for system administrators, so main goal is to make it minimalistic and intuitive.

Every entity is displayed in separate tab which includes drop-down menu of checkboxes for showing/hiding columns, refresh button, line edit for entering filters and table of data. In table, every virtual machine, disk, etc. printed into row and its data are in columns.

Chapter 7

Evaluation

My application can be installed through a RPM package or cloned and build from a public github repository [5]. To test my application I asked four oVirt system administrators to use, test and review it. The application was tested on a large number of different virtual environments ranging from small, with only few virtual machines, hosts, disks, etc., to large ones.

First complaint I received was that large environments can take quite some time to load (around 30 seconds). I already noticed this problem during development. It is simply due to downloading large pool of data from the oVirt API which is highly dependent of the size of the environment and speed of the internet connection. A possible solution would be to download only some part of the information and, if needed, download the rest. I decided to not implement this. If every entity would be fully loaded only after clicking on it, user would have to wait after switching on every new tab. Administrators very rarely use only one or two entities. This way, their work flow would be disturbed by loading screens. Now, user only waits once during the initial load of the application and after it is fully loaded, user can work with a whole environment without additional delays, which leads to more fluid user experience.

Second complaint was aimed to the lack of some displayed data. During the development process, I was collaborating with several system admins, but not with all of them. Current state of my application is good enough for admins I worked with, but in the future, if my application will be used more broadly, some new types of data will be added.

Apart from these minor complaints, overall feedback was positive. My application was described as useful tool which makes everyday work tasks faster and simpler. Admins were able to use it with almost no explanation beforehand.

Chapter 8

Conclusion

The thesis has two main goals. The first one is to introduce virtualization technology. First of all, it means familiarize the reader with the question, why is it even reasonable to explore this concept, what advantages and pitfalls it brings. Then it describes various architectures used to achieve some form of hardware virtualization as well as specific implementations of those architectures and solutions using them.

The second goal is to propose and implement a solution to some of the oVirt administrator's tool problems. The currently available tool for overseeing of virtual environment is called Webadmin and this thesis focuses on it's issues related to effectiveness of user interface and accessibility of provided data. The result is desktop application providing all the needed information and correlations between types of data greatly reducing time needed to look them up. This in major way helps with day-to-day work tasks of system administrator's. The resulting solution is based upon requests from community of admins which shaped and formed a the final look and functionality of application. The result application was tested by number of administrators as their standard working tool and was met with a positive feedback and is currently waiting to become a part of standard toolset associated with an oVirt environment.

To create the mentioned application, I needed to understand technical aspects of the oVirt project as well as its main admin interface Webadmin. Through communication with both of my supervisors and several engineers at Red Hat, we came up with an resulting application. Creating of my solution has, apart of other things, taught me a lot about a non-technical aspects of software development process, mainly collaboration with larger teams of people. For a technical part, I learned to use a PyQt5 package and ovirt-python-sdk as well as python version 3.

I plan to continue to maintain my application. In the future, there is going to be added new functionality primarily including manipulation with virtual environment, as for now, it is only for data overview. Adding new functionality is the main reason the whole application is build as a collection of mostly independent modules, this means that new modules can be incorporated very easily from a code base point of view.

Bibliography

[1] Apache CloudStack: *Apache Cloudstack: Open Source Cloud Computing.* [Online; visited 21.04.2018].
Retrieved from: https://cloudstack.apache.org/

[2] Citrix: *Mobilize Your Business with Secure App and Data Delivery - Citrix.* [Online; visited 02.05.2018].
Retrieved from: https://www.citrix.com/

[3] Docker: *Docker - Build, Ship, and Run Any App, Anywhere.* [Online; visited 24.04.2018].
Retrieved from: https://www.docker.com/

[4] Gerard Blokdijk, Ivanka Menken: *Virtualization - The Complete Cornerstone Guide to Virtualization Best Practices.* Emereo Publishing. 2009. ISBN 978-1-921523-91-5.

[5] GitHub: *xmacko10/IBP:Bachelor's Thesis.* [Online; visited 14.05.2018].
Retrieved from: https://github.com/xmacko10/IBP

[6] Jeff Dike: *User Mode Linux.* Prentice Hall. 2006. ISBN 978-0131865051.

[7] Michael Pearce, Sherali Zeadally, Ray Hunt: *Virtualization: Issues, security threats, and solutions.* [Online; visited 05.03.2018].
Retrieved from: http://www.profsandhu.com/cs6393_s14/csur_virt_2013.pdf

[8] Microsoft: *Microsoft - Official Home Page.* [Online; visited 05.05.2018].
Retrieved from: https://www.microsoft.com/en-us/

[9] Oracle: *Oracle | Integrated Cloud Applications and Platform Services.* [Online; visited 04.05.2018].
Retrieved from: https://www.oracle.com/index.html

[10] oVirt: *oVirt.* [Online; visited 08.05.2018].
Retrieved from: https://www.ovirt.org/

[11] Patrice Clemente, Jonathan Rouzaud-Cornaba: *Security and Virtualization: a Survey.* [Online; visited 05.03.2018].
Retrieved from: https://hal.archives-ouvertes.fr/file/index/docid/995214/filename/RR-2010-06.pdf

[12] Rackspace Cloud Computing: *Open source oftware for creating private and public clouds.* [Online; visited 21.04.2018].
Retrieved from: https://www.openstack.org/

[13] Red Hat: *Red Hat - We make open source technologies for the enterprise.* [Online; visited 07.05.2018].
Retrieved from: https://www.redhat.com/en

[14] Red Hat: *Virtualization for the enterprise.* [Online; visited 07.05.2018].
Retrieved from: https://www.redhat.com/en/technologies/virtualization/enterprise-virtualization

[15] Red Hat: *What is KVM?* [Online; visited 22.04.2018].
Retrieved from:
https://www.redhat.com/en/topics/virtualization/what-is-KVM

[16] The Linux Foundation: *Open Containers Initiative.* [Online; visited 24.04.2018].
Retrieved from: https://www.opencontainers.org/

[17] VirtualBox: *Oracle VM VirtualBox.* [Online; visited 22.04.2018].
Retrieved from: https://www.virtualbox.org/

[18] VMware: *VMware - Official site.* [Online; visited 02.05.2018].
Retrieved from: https://www.vmware.com/

[19] WineHQ: *Wine Developer's Guide/Architecture Overview - WineHQ Wiki.* [Online; visited 26.04.2018].
Retrieved from:
https://wiki.winehq.org/Wine_Developer%27s_Guide/Architecture_Overview

[20] Xen Project: *The Xen Project, the powerful open source industry standard for virtualization.* [Online; visited 21.04.2018].
Retrieved from: https://www.xenproject.org/

Appendix A

Building and usage

A.1 Introduction

ovirt-inventory is an application for easy access to the RHEV inventory.

A.2 Building

To run *ovirt-inventory*, user needs python3 and ovirt-engine repository already installed.

1. User can clone git repository containing source codes from [5].
   In that case, user also needs to install python3-qt5 and python3-ovirt-engine-sdk4.
2. User can use a RPM file [5] which will install everything needed.

A.3 Usage

Logging in

After launch, input dialog containing three input fields is displayed.

- *username* field expecting username and domain in format: username@domain
- *password* field expecting corresponding password
- *url* field expecting FQDN of the target virtual machine

Hide/show columns

By checking or unchecking checkboxes in 'Select Column', drop-down menu user can hide or show table columns.

Refresh

By pressing 'Refresh Button' user can load up-to-date data.

Filter

User can write custom filters into an filter field. Filter needs to be in format 'column name' '<|>|=' 'value'. Filter supports evaluating multiple filters at the same time. All sub-filters are in 'AND' relation and need to be separated from each other by ','.

Ordering

User can order items in columns in ascending and descending order by clicking on column he wants to order.

Redirecting

Some table cells contain multiple values (for example one virtual machine can contain multiple disks). In that case, the cell displays first item followed by the number in parentheses expressing how many other items are in that cell. By double clicking such cell, user is redirected to the corresponding tab and right filter is applied.

Export to .csv

User can export current table into .csv file. Go to 'File' -> 'export'.

Save

User can save current configuration of all tables into config file by going 'File' -> 'export'. These information will be saved in a config file:

- user name
- domain
- checkboxes from all the tables