ISSN XXXX XXXX © 2017 IJESC

Research Article Volume 7 Issue No.6

A Comparative Study on Virtual Machine versus Containers for Embedded Systems such as Enterprise Routers

Sumanth .N[1], Dr K. Satyanarayan Reddy[2]
PG Scholar M.Tech[1], Professor & HOD[2]
Department of CSE[1], Department of ISE
Cambridge Institute of Technology, Visvesvaraya Technological University, Belgaum, Bangalore, India

Abstract:
In recent times, virtualization as an abstraction from physical hardware has become a popular solution to resource isolation and server consolidation. Virtualization solutions allow multiple operating systems and applications to run in independent partitions all on a single computer. Using virtualization capabilities, one physical computer system can function as multiple "virtual" systems. With the x86 hardware architecture supporting Intel virtualization (VT-x) and Intel® Virtualization Technology for Directed I/O (VT-d) running virtual machines have become quite efficient. While virtual machines are getting quite popular, we see that in parallel Linux containers are also becoming a popular choice to provide an operating system level virtualization method for deploying and running distributed applications without launching an entire VM for each application in big systems such as servers. We propose to investigate these two forms of virtualized platforms (i.e., hypervisor-based platform and container-based platform) in terms of limitations, advantages, features keeping in mind embedded systems especially Enterprise Routers.

Keywords: Virtualization, Virtual Machines, Hypervisor, Linux Containers (LXC), embedded systems, Routers.

I. INTRODUCTION

Virtualization can be defined as the art of running one computer operating system on top of another, and has a long and venerable history[1] , [2], [3], [4]. Virtualization typically refers to the creation of virtual machine that can virtualize all of the hardware resources, including processors, memory, storage, and network[8]. With the virtualization, physical hardware resources can be shared by one or more virtual machines. Virtualization requires hypervisor which enables communication between hardware and a virtual machine, hence virtualization accomplishes with this abstraction layer. With virtualization we can launch multiple virtual machines with each virtual machine having its own kernel space. Container based virtualization is very fast and efficient [4].

It's based on the premise that an OS kernel provides different views of the system to different running processes. This sort of segregation or compartmentalization (sometimes called "thick sandboxing") can be useful for ensuring guaranteed access to hardware resources such as CPU and IO bandwidth, while maintaining security and efficiency. LXC is similar to other OS-level virtualization technologies on Linux The goal of LXC is to create an environment as close as possible as a standard Linux installation but without the need for a separate kernel. LXC offers complete logical isolation from a container to the host and all other containers. Each container will share the kernel with the host (and other containers).

No kernel need to be present and/or mounted on the containers /boot directory. Also we will be able to run different Linux distributions on the same host kernel in different containers This paper looks at two different ways of achieving resource control today, viz., containers and virtual machines and compares the advantages and disadvantages of these technologies with respect to embedded systems, especially Enterprise Routers.

II. VIRTUAL MACHINES

Kernel-based Virtual Machine (KVM) is a virtualization technology for Linux systems that turns it into a hypervisor and is designed for x86 processor architecture. The hypervisor is built on Linux kernel, running an open source operating system. It provides support for multiple guest OS include windows, BSD, and Solaris. KVM is a full virtualization technique where you can provide virtualized hardware, that allocates the number of CPUs, memory, hard disk space to the guest operating system. The goal of a Virtual Machine Manager (abbreviated as VMM or called a hypervisor) is to abstract the hardware of a computer, thus creating a virtual computer. A physical computer hosting a VMM is called a host, while the program running inside a virtual machine is called a guest. The interface that is provided to the guest is identical (or nearly identical in certain cases) to that of the host. All guests and the host are mutually independent from each other. The VMM maps the virtual CPU(s) of all actually running VMs to the physical CPU(s) of the host. Hence, there are usually more VMs running on a host than physical CPUs are attached to it, causes the need of some kind of scheduling. Therefore a VMM uses a certain scheduling mechanism to assign a certain share of the physical CPUs to each virtual CPU. A VMM has to deal with memory management, also. It maps an amount of physical memory into the VMs address space and also has to handle fragmentation of memory and swapping. Since some VMs need more memory than others, the amount of assigned memory is defined and often dynamically adjusted by using the management tools. Usually, the VMs don't have access to the physical hardware and don't even know about it either. Only if direct access is desired, devices may be passed through directly[7]. For running legacy software this may be a point. But in more common scenarios the VMM provides virtual I/O devices like network cards, hard disks and cd drives. Since a VMM provides different VMs mostly with same hardware, it is much easier to migrate

them between hosts running the same VMM. The drivers for the virtual I/O devices need to be installed only once in this case.

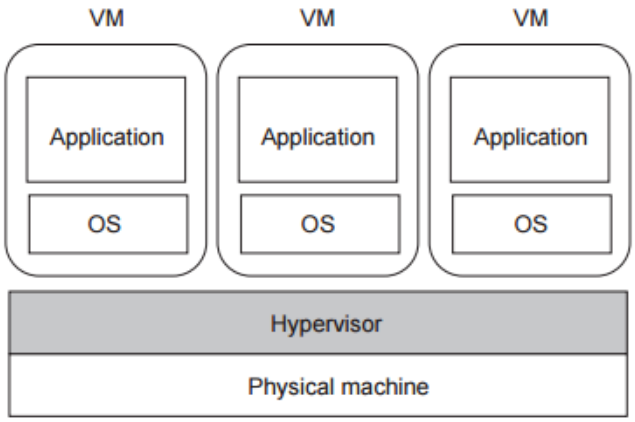

Figure .1.Virtual Machines on a hypervisor

Intel Virtualization Technology (Intel VT) and AMD Virtualization (AMD-V) support functions in Intel and AMD cpu that provide hardware support so that Virtualization and be implemented without having significant performance penalty.

III. LINUX CONTAINERS

Virtualization utilizing containers is a simpler alternative to hypervisor-based virtualization. Rather than running a full OS on virtual hardware, container-based virtualization modifies an existing OS to provide extra isolation. Generally this involves adding a container ID to every process and adding new access control checks to every system call. Thus containers can be viewed as another level of access control in addition to the user and group permission system. In practice, Linux uses a more complex implementation described below. Linux containers are a concept built on the kernel namespaces feature, originally motivated by difficulties in dealing with high performance computing clusters [9]. This feature, accessed by the clone() system call, allows creating separate instances of previously-global namespaces. Linux implements file system, PID, network, user, IPC, and hostname namespaces. For example, each file system namespace has its own root directory and mount table, similar to chroot() but more powerful. Namespaces can be used in many different ways, but the most common approach is to create an isolated container that has no visibility or access to objects outside the container. Processes running inside the container appear to be running on a normal Linux system although they are sharing the underlying kernel with processes located in other namespaces. Unlike a VM which runs a full operating system, a container can contain as little as a single process. A container that behaves like a full OS and runs process like initetc and demons such asinetd, sshd, syslogd, cron, etc. is called a system container while one that only runs an application is called an application container. Both types are useful in different circumstances. Since an application container does not waste RAM on redundant management processes it generally consumes less RAM than an equivalent system container or VM. The process of isolation is implemented for containers at the level of OS for the host machine, thereby avoiding the overhead due to virtualized device drivers and hardware [9]. It is also known as operating system level virtualization as several isolated user-spaces run at the operating system level allowing abstractions directly to guest process[10], [11].OS level virtualization does not rely on hypervisor unlike Para and full virtualization.

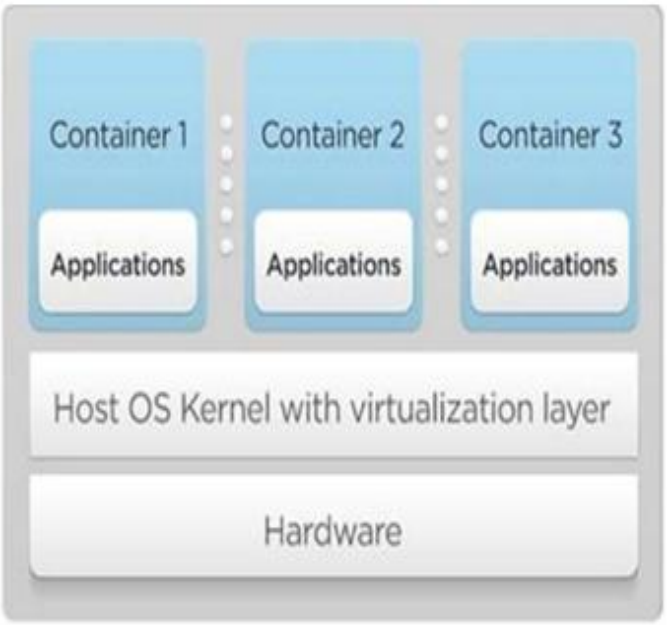

Figure 2. Containers on Host Operating system

IV. CHARECTERISTICS OF AN ENTERPRISE ROUTER

An enterprise router is designed to operate in the Internet backbone, or core or edge of the network. To fulfill this role, a router must be able to support multiple telecommunications interfaces of the highest speed in use in the core Internet and must be able to forward IP packets at full speed on all of them. Very large computer networks commonly use a hierarchy of routers[19]. At the top of this hierarchy are core routers, the fastest and most powerful class. A core router is a type of very powerful computer router used in large computer networks. It is the fastest, most powerful, and most expensive class of router available[18].

A core router sits at the heart of a network and manages the flow of data packets within the network, often relying on lesser routers for connectivity. A single core is capable of processing millions of packets every second. It generally sits in the "center" of very large networks and sends and receives packets to lower classes of routers, such as edge routers, which sit on the edge of a network and transfer packets to other networks. These routers can communicate with one another using the Border Gateway Protocol (BGP) and may share information about the best routes to take or network destinations that have become unreachable.

Early incarnations of the core router contained a "global routing table," a database containing virtually every possible route a given packet could take to reach its destination. These routers, therefore, were considered to be in the core or backbone of the Internet and were an essential component of early Internet architecture. As the Internet grew, however, even the most advanced router couldn't keep up with the number of possible routes. Large networks were subdivided into smaller units known as autonomous systems (AS). The modern core router still maintains a large routing table; the scope of this table is confined to the AS rather than the Internet as a whole, however, making the concept of a "core" Internet largely obsolete.

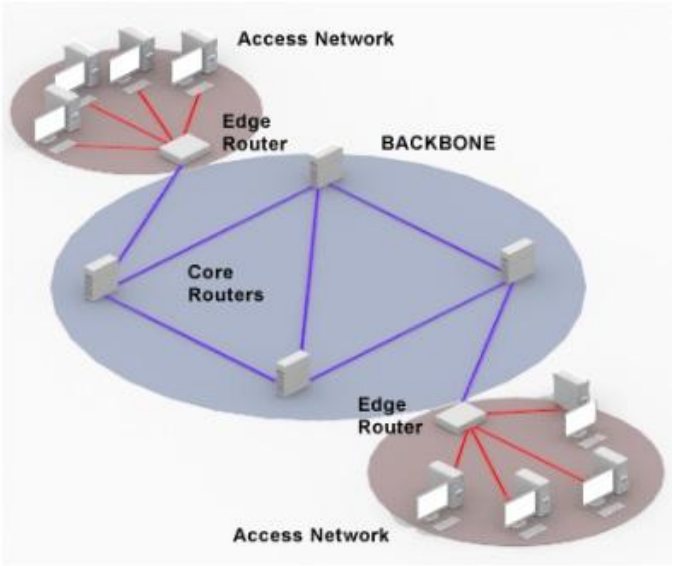

Figure .3.Enterprise Routers

An edge router[17] is a specialized router residing at the edge or boundary of a network. This router ensures the connectivity of its network with external networks, a wide area network or the Internet. An edge router uses an External Border Gateway Protocol, which is used extensively over the Internet to provide connectivity with remote networks. Edge routers use External BGP Protocol for data transmission because they are intermediary devices between two different networks and operate at the external or border layer of the network. There are several types of edge routers, including edge routers placed at the outer boundary of the network as an essential device for connecting the host network with the Internet. Whenever a node sends data on a network unmonitored by the host administrator, the data packet is sent to the last router on the authorized network, which is the edge router. There are two types of edge routers, the subscriber edge router and label edge router. The subscriber edge router is used in scenarios where it serves at the border device. The label edge router is used in Multi Protocol Label Switching (MPLS) and assigns labels to outbound data transmissions. Instead of providing communication with an internal network, which the core router already manages, an edge router may provide communication with different networks and autonomous systems. The typical Internet core/edge link speed is 40 Gbit/s, with many links at higher speeds, reaching or exceeding 100 Gbit/s, provisioning the explosion in demand for bandwidth in the current generation of cloud computing and other bandwidth-intensive (and often latency-sensitive) applications such as high-definition video streaming (IPTV) and Voice Over IP.

V. ASSESSMENT

A virtual machine has its own set of requirements and characteristic and so do containers. We can see how these requirements and characteristics will suit the requirements of embedded system like an enterprise router. Enterprise router being a very critical piece of internet backbone we see how a particular characteristic of the virtualization technology will affect it. The behavior of various parameters under the defined workload is observed carefully and recorded, to study its characteristics and get a consolidated comparison between

A. VT-x Technology

With VT-x Intel introduced hardware support for virtualization[13]. "vmx" stands for Virtual Machine Extensions, which added new instructions like vmptrld, vmptrst, vmclear, vmread, vmwrite, vmcall, vmlaunch, vmresume, vmxoff, and vmxon. Without this there was a huge performance impact for virtual machines to be launched on a host Operating system. Not all processors support VT-x technology, for example few of the Intel Atom processors do not support VT-x. Quite a few enterprise routers use Intel Atom processors, as atom processors consume less power when compared to server calls processor. Hence the choice of processor has a bearing on the decision to run Virtual machines on a enterprise router. Hence if we choose to run a virtual machine we need to choose a high end processor, which will consume more power and would have larger cooling requirements etc. Linux containers do not have a dependency on the need for hardware support for virtualization.

B. VT-d Technology

VT-d Technology[13]allows guest virtual machine to directly use peripheral devices, such as Network processors etc. through DMA and interrupt remapping. This is sometimes called PCI pass through. This allows operating systems to eliminate bounce buffers needed to allow themselves to communicate with peripheral devices whose memory address spaces are smaller than the operating system's memory address space, by using memory address translation. At the same time, it also allows operating systems and hypervisors to prevent buggy or malicious hardware from compromising memory security. Enterprise routers need VM's to have full control of the pCIE endpoint such as network processor as it has to program the huge network routing table. Also the routing table keeps on changing with the addition of new routes and same has to be updated constantly. If the VM does not have full control of the endpoint then every write to the endpoint will result in VM exit leading to huge performance penalty. Hence processor with VT-d is a must for enterprise router. Linux containers do not have a dependency on the need for hardware support for virtualization as containers share the same kernel as the host operating system, hence the container can take full control of the pCIE endpoint such as network processor. Not all processors support VT-d technology, for example few of the Intel Atom processors do not support VT-x. Quite a few enterprise routers use Intel Atom processors, as atom processors consume less power when compared to server calls processor. Hence the choice of processor has a bearing on the decision to run Virtual machines on a enterprise router. Hence if we choose to run a virtual machine we need to choose a high end processor, which will consume more power and would have larger cooling requirements etc. Linux containers do not have a dependency on the need for hardware support for virtualization.

C. Network Operating systems

The leaders who are also the legacy players in enterprise routing segment have their own propitiatory network operating system like, Cisco has IOSXR [14] and Juniper has JUNOS [15]. Proprietary network stacks, network protocols and other software have been developed over years by them. For an enterprise router where the scale is very important, these routing stacks are the heart of system. These network operating systems do not have support for containers etc. Hence moving their all their legacy software to containers is not possible, hence running their network operating system as virtual machine is the only option for them. The newer

players like cumulus who are developing the Network operating systems are developing on top of Linux. These players are taking advantage of the features that containers offer and are developing their network operating systems using containers.

D. Number of Instances of Operating Systems.

Each virtual machine has its own version of kernel as a result the amount of resources that each virtual machine consumes is high. As a result of which the number of virtual machines that can be launched is quite small. In an enterprise router usually 2 and maximum of 3 virtual machines are launched. Linux containers are pretty light weight i.e. they use very less resources as the containers share the kernel with the host system. This is a major advantage in a enterprise router as set of related processes can be bundled together as package and launched in a container. With container taking less resource than a virtual machine we can launch many containers as compared to only couple of Virtual machines.

E. Upgradability of Kernel

In a virtual machine each virtual machine has its own kernel. The kernel that runs in virtual machine can be same or different. In any case if the kernel has to be patched all the virtual machine's have to be brought down and kernel upgraded. Also patching the kernel of virtual machine and rebooting takes more time.

With the above reason in a enterprise router where the downtime [5]is supposed to very minimal even during upgrade virtual machine have major disadvantage.

In container each container shares the kernel with the host system. Hence if the host kernel is upgraded the whole system is upgraded.

Since the downtime for a upgrade[12] is less compared to a virtual machine, containers have a major advantage over virtual machine[6].

F. Size of image and memory usage

Each virtual machine has its own kernel along with the application. Hence the size of the image and size of the memory used will be higher in virtual machines[16].

In an Enterprise router the if the size of the image is bigger the upgrade time will become bigger and this is not desirable. Also memory is premium in an enterprise router and virtual machines have a distinct disadvantage over container.

Various flavours of containers can be launched on a host operating system. The size of the image varies depending on the flavor of container launched. The size of a Yoctocontainer can be as small as 30 MB. With the size of the being very small compared to a virtual machine, it takes less memory and also the upgrade times will be faster, hence its preferred in a enterprise router.

VI. RESULTS

The following table summarizes the results of the comparative study that was done to analyze weather virtual machines or containers are better for an enterprise router.

	Virtual Machine	Container
VT-x Technology	Processor with VT-x is a must to run virtual machine. Hence Intel atom processor cannot be used, leading to more power consumption	Not dependent on any hardware technology. Hence atom processor can be used, leading to more power consumption
VT-d Technology	Processor with VT-d is a must as enterprise routers have a lot of network packets which needs to be routed directly to VM. Without VT-d this is not possible	Not dependent on any hardware technology, hence the network packets can be directly given to a container.
Network Operating systems	Legacy players like Cisco and Juniper can deploy their existing Operating systems as a virtual machine leading to cost savings as there is no need to port all the existing software into new environment.	Newer players like cumulus who are developing the Network operating systems are developing on top of Linux.
Upgradability of Kernel	Each virtual machines has its own kernel hence all the kernels have to be patched leaving to more upgrade times.	In container each container shares the kernel with the host system. Hence if the host kernel is upgraded the whole system is upgraded, leading to lesser upgrade times.
Size of image and memory usage	Virtual machine has its own kernel along with the application. Hence the size of the image and size of the memory used will be higher in virtual machines	The size of the image varies depending on the flavor of container launched. The size of a Yocto container can be as small as 30 MB

VII. CONCLUSIONS

This study analyzes the behaviors, advantages and disadvantages of virtualmachines and containers with respect to enterprise routers. As the survey clearly states, that no one, technology can be substituted by another right away, each having its own merits and demerits and each technology can be used efficiently with proper configuration suiting your particular needs. However we see that containers have a major advantage over virtual machines with respect to enterprise router. However we see that custom software such as routing

protocol stack etc that are critical for an enterprise router have been developed over a decadeover custom operating systems. Hence for the legacy players virtual machine is the choice for new developments and for new players containers is the choice for new developments.

VIII. ACKNOWLEDGMENT

The Author acknowledges the support given by the chairman Mr. D.K. Mohan of Cambridge Institute of Technology, Bangalore, Karnataka, India for providing the necessary library resources for this carrying out this work.

IX. REFERENCES

[1]. Xavier, M.G., Neves, M.V. and De Rose, C.A.F., 2014,February. A performance comparison of container based virtualization systems for mapreduce clusters. In 2014 22nd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (pp. 299-306). IEEE.

[2]. Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris,Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen andthe Art of Virtualization. ACM SIGOPS Operating Systems Review,37(5):164–177, 2003.

[3]. Avi Kivity, Yaniv Kamay, Dor Laor, Uri Lublin, and Anthony Liguori.KVM: the Linux Virtual Machine Monitor. In Proceedings of the Linux Symposium, volume 1, pages 225–230, 2007.

[4]. LXC. LXC - LinuX Container. https:// linuxcontainers. org/, visited June 2017.

[5]. Jim Gray and Daniel P. Siewiorek. High-Availability Computer Systems. Computer, 24(9):39–48, 1991.

[6].Wubin Li, Ali Kanso, and Abdelouahed Gherbi. Leveraging Linux Containers to Achieve High Availability for Cloud Services. In Proceedingsof the IEEE International Conference on Cloud Engineering(IC2E 2015). IEEE, 2015. accepted.

[7]. Steven Osman, Dinesh Subhraveti, Gong Su, and Jason Nieh. The Design and Implementation of Zap: A system for Migrating Computing Environments. ACM SIGOPS Operating Systems Review, 36(SI):361–376, 2002.

[8]. G. J. Popek and R. P. Goldberg. Formal requirements for virtualizable third generation architectures. Commun. ACM, 17(7):412–421, July 1974.

[9].A. M. Joy, "Performance comparison between Linux containers and virtual machines," in Computer Engineering and Applications (ICACEA), 2015 International Conference on Advances in, 2015, pp. 342–346.

[10]. M. G. Xavier, M. V. Neves, F. D. Rossi, T. C. Ferreto, T. Lange, and C. A. De Rose, "Performance evaluation of container-based virtualization for high performance computing environments," in Parallel, Distributed and Network-Based Processing (PDP), 2013 21st Euromicro International Conference on, 2013, pp. 233–240.

[11].S. A. Babu, M. J. Hareesh, J. P. Martin, S. Cherian, and Y. Sastri, "System performance evaluation of para virtualization, container virtualization, and full virtualization using xen, openvz, and xenserver," in Advances in Computing and Communications (ICACC), 2014 Fourth International Conference on, 2014, pp. 247–250.

[12].W. Felter, A. Ferreira, R. Rajamony, and J. Rubio, "An updated performance comparison of virtual machines and linux containers," in Performance Analysis of Systems and Software (ISPASS), 2015 IEEE International Symposium On, 2015, pp. 171–172.

[13]."Understanding Full Virtualization, Paravirtualization, and Hardware Assist." [Online]. Available: http://www. vmware.com/techpapers/2007/understanding-full-virtualizatio n -paravirtualizat-1008.html visited June 2017

[14]. "Cisco IOS XR Software" http://www. cisco.com/ c/en/us/products/ios-nx-os-software/ios-xr-software/index. html visited June 2017.

[15]. "JUNOS OS - Advanced end-to-end network operating system that spans routing, switching, and security devices, both physical and virtual." https://www. juniper.net/ us/en/products-services/nos/junos/visited June 2017.

[16].C. N. Corrêa, S. C. de Lucena, D. de A. L. Marques, C. E. Rothenberg, and M. R. Salvador, "An experimental evaluation of lightweight virtualization for software-defined routing platform," in 2012 IEEE Network Operations and Management Symposium, 2012, pp. 607–610.

[17]. Don Caldwell, Anna Gilbert, Joel Gottlieb, Albert Greenberg, Gisli Hjalmtysson, and Jennifer Rexford. The cutting EDGE of IP router configuration. In Second Workshop on Hot Topics in Networks (HotNets-II), November 2003.

[18]. L. Kleinrock and F. Kamoun. Hierarchical routing for large networks: Performance evaluation and optimization. Computer Networks, Vol. 1, pp. 155-174.

[19]. L.Subrmanian, S. Agarwal, J. Rexford and R.Katz. Characterizing the internet hierarchy from multiple vantage points. In Proc. IEEE INFOCOM, 2002.