ISSN:0975-9646

Operating System Virtualization for Ubiquitous Computing

Vikram S. Vairagade
Department of CSE, NIT
Nagpur University , INDIA

Prof.Chanchal V. Dahat
Department of ETC, NIT
Nagpur University, INDIA

Anjali V. Bhatkar
Department of IT, TGPCET
Nagpur University, INDIA

Abstract-The basic concept of virtualization is to provide the benefits of the services and components irrespective of its physical presence. Operating System (OS) virtualization describes the abstraction of an operating system from any underlying hardware architecture. OS virtualization is needed as it provides feature of transparent migration of applications. Ubiquitous computing is an advanced computing concept where computing is made to appear everywhere and anywhere. The infrastructure of ubiquitous computing which exploits virtualization to make computing. In order to enable ubiquitous environment and servers to be shared the application of various operating systems with the desktop of user. Using the ubiquitous environment, the applications could be run in the host system without installation. The key concept of the system it to operate the desktop of user by any handy device (like smart phones or tablet) through web browser irrespective of the location of user.

Keywords: Ubiquitous Computing, OS Virtualization, Virtual machine, trusted computing, cloud computing.

1. INTRODUCTION:

The current scenario in computer science is considerably shifting towards cloud -based services. The abstraction of the maintenance issues from the device has been the main motive behind this. In this upcoming era of cloud-based software, applications live on the cloud as services that can be accessed with a web browser. The services consist of data, computation and other resources that can be at anywhere in the sphere. The services and applications require no installation; this makes the deployment of applications on the cloud server exceptionally simple and rapid. Ideally, applications should also support user interaction and collaboration, i.e., allow multiple clients devices to interact and share the same data and application set over the Internet. There is another technique currently occurring in the present days: mobile devices are becoming an important application platform and a gateway to the Web. While the Web has conventionally been accessed from a personal computer, the increasing network bandwidth, processor speeds, memory capacity, and network service plans are rapidly making mobile web usage and mobile software applications more practical. These two transitions – the movement towards cloud-based software and towards web-enabled mobile devices – are transforming present era in many important ways. We believe that in the long run the popularity of the Cloud will make it the well used application platform in the world. We also believe the growing popularity of mobile devices – there are already about three billion mobile device users today – will lead us to a common application platform that can be used with different devices, including desktop computers and mobile devices.

Nowadays, Software as a Service (SaaS) has been an infrastructure for the cloud computing, in which software is stored in the warehouses, applications, which are downloaded on demand, run on a host system and the agents used by customers become an input and display device only. Therefore, on the host system, a virtual execution environment should be provided, which could assist applications to run simultaneously and keep the personalized data during the execution period. An approach for the virtual execution environment is the virtual machine technology, in which each application runs in its own OS. However, this approach could waste the hardware resources because of the duplication of OS. Another light-weight approach is to provide a solution of virtual execution environment for the Windows or Linux-based applications using the application virtualization technology.

2. VIRTUAL OS FOR UBIQUITOUS COMPUTING

In this section, we first describe about the requirements for a ubiquitous computing system and then the utilization of virtualized Operating Systems as such an infrastructure in that system. In this section we also describe the resource management to create stable computing environments for ubiquitous computing, and configuration system to ease the use of virtual Operating Systems.

2.1Requirements

Ubiquitous computing is an advanced computing concept where computing is made to appear everywhere and anywhere. The ubiquitous computing environment is utilized to share multiple users the same environment. For example, an environment constructed in a home is shared by the family members, and one in an office space is shared by the office employees. A same user of a ubiquitous computing environment uses ubiquitous computing devices embedded in it. Means the same environment is shared by many users and also devices are also shared by them. These shared computing devices are known as servers. Such an environment where sharing servers can happen everywhere. There are two major requirements to the infrastructure of computing environment

a) Security
b) Stability

A)Security
A security environment should have the potential to prevent virus attacks from intruding into the environment. There may be a hidden software problem that creates a security hole. Such a security hole may attack and allow an intrusion in the environment, the requirement of secure infrastructure is that it should contain the security breach in the infected environment and it should not affect other environments. In private ubiquitous computing environments, different levels of security for different users should also be implementing so that they have many different preferences and attitudes for the security.
b)Stability
Stability is necessary to support application which is time sensitive. Timing requirements are there in time sensitive applications and there is a requirement of certain amounts of CPU times in time frames to be allocated to them. The time sensitive applications should not disturb execution of some other stable non-time sensitive applications.
PDAs (Smartphone, tablet) should not share, but similar requirements can be applied because they are also utilized in such similar shared environments.

2.2 Operating Systems Virtualization

For the above requirements, we utilize virtual Operating Systems with the CPU resource management functionality. Virtual Operating Systems are the Operating Systems of which multiple instances can run on one single computer machine. Basically there are 2 models to realize virtual Operating Systems, Virtual Machine Monitor that creates a Virtual Machine in which an Operating Systems runs and other is to access an OS personality server running on top of host kernel, which will be a microkernel or a monolithic kernel. These 2 models are different in only a way for the underlying layers to provide abstractions of computing resources from here we use the host kernel to describe the underlying layers that provides the abstractions of computing resources. Opposite of this to the host kernel, the Operating Systems kernel in a virtual Operating Systems environment is known as a guest kernel.
The Operating Systems virtualization of can isolate single user's execution environment from the others that share the similar ubiquitous server since users can own and use their personal installations of Operating Systems. The execution of an untrusted application can be done in separate virtual Operating Systems. Moreover, virtual Operating Systems can append advanced security features by using virtual machine based intrusion detection technologies. The other significance of using virtual Operating Systems for personal ubiquitous computing environments is that different virtual operating Systems can run parallel on the same ubiquitous server. Such a feature makes it possible to support legacy applications and it protects the past investments and enables phased transitions to new application platforms.
Fig. 1 shows the model realizes virtual operating systems on top of host kernel. The fig. shows an example configuration on ubiquitous server that is shared by User A and User B. They use their own virtual operating systems. User A uses 2 virtual operating systems, VOS 1 and VOS 2. User B uses one virtual operating system, VOS 3. There is another virtualized OS, VOS 4, which is due for core system services. Applications in VOS1 and VOS 3 can communicate with a server in VOS 4 through internal connections provided by the host kernel. Separate virtual operating systems; VOS 4 is used to run the core system services in order to be isolated from any faults and security breaches that can happen in user's virtual operating systems.

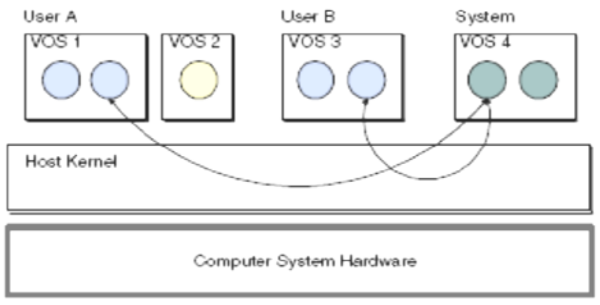

Figure 1. Virtualized Operating Systems

Users can also employ a separate virtualized OS to run an untrusted application. For example, User A uses VOS 2 to run an untrusted application. The host kernel provides CPU resource management to guarantee and to limit the allocation of CPU times to each virtualized OS. A virtualized OS that runs time sensitive applications can always receive certain
CPU times needed for their stable execution. A virtualized OS that runs untrusted applications can be configured to receive limited CPU times, so that their effect to the other virtualized Operating Systems can be restricted even if they fall into an infinite loop.
More details of resource management are described in the next section, and more application scenarios using virtualized Operating Systems are presented in Section 3.

2.3 Resource Management

In order to have an execution environment completely isolated from the others, the functionality of CPU resource reservation is required to protect CPU resources allocated for the environment. One computing environment created by a virtualized OS is completely isolated if the execution of programs in that environment is not affected by activities in the other virtualized Operating Systems that share the same computer system. It means that programs in that environment can run as if its OS occupies a single computer system. Since personalized ubiquitous computing environments require guaranteed allocation of CPU times for their time sensitive applications as described above, those CPU times have to be allocated to their virtualized Operating Systems that host those environments.
Thus, the CPU resource reservation mechanisms are needed in the host kernel. Such reservations of CPU times also work as CPU resource protection especially if actual utilization of CPU times can be enforced to cap the maximum CPU time

allocation. The enforcement of CPU time utilization prevents applications, and thus virtualized Operating Systems, from overusing CPU times. It can limit the negative effects on CPU resource allocation to the other virtualized Operating Systems.

Figure 2 depicts the overview of the architecture and its components. There are the resource management subsystems in both the host and guest kernels.

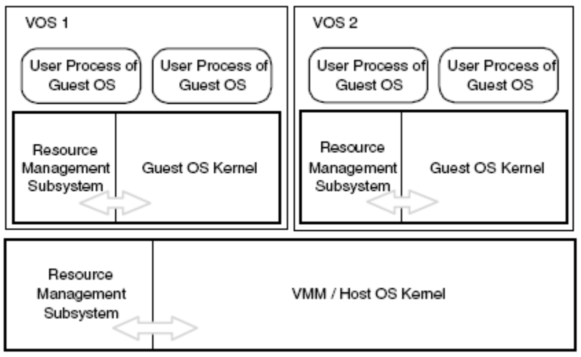

Figure 2. Architecture of Virtualized OS with Resource Management Mechanisms

The resource management mechanisms in the host and guest kernel control the allocation of CPU times to virtualized Operating Systems and the guest kernel's user processes, respectively. An appropriate CPU time reservation is made for a virtualized OS in order to enable CPU time reservations in the virtualized OS.

2.4. Configuration and Instantiation

In order to be able to use virtualized Operating Systems as our ubiquitous computing infrastructure, we also need a system that eases the instantiation and configuration of virtualized Operating Systems. Our instantiation and configuration system helps users to create new virtualized Operating Systems and to configure them for specific uses. For example, when a user would like to create a new virtualized OS to run an untrusted application that accesses Internet sites and sends back retrieved information, the virtualized OS needs an Internet connection and an internal connection to the originating virtualized OS.

The user can use the instantiation and configuration system to automate the process of the creation and configuration of such a virtualized OS and the execution of the specified application in it.

3. Application Scenario

In this section we describe 2 application scenarios of using virtualized Operating Systems on a server and a PDA.

A user can use multiple instances of virtualized Operating Systems to configure environments to isolate untrusted applications and to contain them in separate virtualized Operating Systems. Such separate Virtualized Operating Systems can be associated with limited CPU resources by using our resource management mechanisms, so that their overuse of CPU times does not affect the other applications. Virtualized Operating Systems that run untrusted applications are exposed to risks of being intruded by viruses. Infected virtualized Operating Systems can simply be removed without losing any important information. In contrast to untrusted applications, trusted time sensitive applications can consume reserved CPU times to maintain the desired Operating Systems specified by their users. Our resource management mechanisms guarantee the allocation of reserved CPU times to those applications.

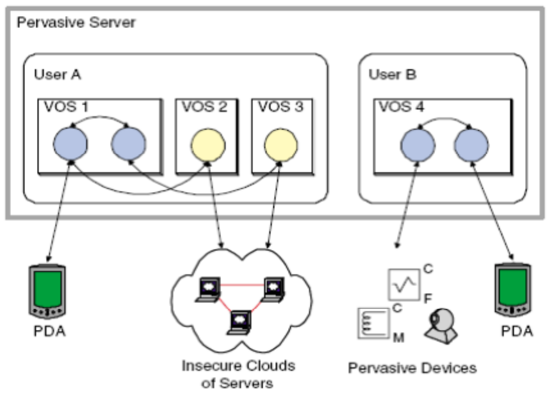

Figure 3. Application Scenario on a Server

Figure 3 depicts our application scenario on a server. User A uses three virtualized Operating Systems, VOS 1, 2, and 3. User A runs untrusted applications, which were downloaded from the Internet, in VOS 2 and 3. The allocation of CPU times to VOS 2 and 3 is capped to limit their CPU resource usage.

By using such a configuration, VOS 1 can be protected from the untrusted applications in VOS 2 and 3. Even if they have software bugs and fall into a busy infinite loop trying to use CPU times as much as possible, our CPU resource management mechanisms limit their CPU resource usage by certain amounts. If they were viruses and intruded VOS 2 and 3, the intrusion does not affect VOS 1 and those contaminated virtualized Operating Systems can be simply abandoned. User B also uses the same server at the same time. User B uses only one virtualized OS, VOS 4, in which time sensitive applications, which control devices, are running. Our software architecture enables those time sensitive applications running in VOS4 to reserve CPU times and to guarantee their timely execution. Any activities of User A do not affect the execution of User B's applications. Therefore, User A and B can securely share the same server.

A PDA is a personal device that is not shared with other users; thus, there is no need to consider its sharing. Virtualized Operating Systems, however, can be used to configure a PDA to internally realize protected domains by running applications and core services in different virtualized Operating Systems. A firewall running in a separate virtualized OS can also be used to reinforce the security of the PDA.

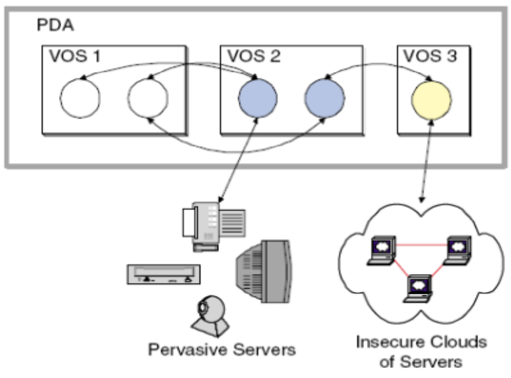

Figure 4. Application Scenario on a PDA

Figure 4 depicts our application scenario on a PDA. Core services, such as a storage service and a window system, are running in VOS 1. User applications are running in VOS 2. Connections to the Internet are provided through the firewall running in VOS 3. By employing the firewall and having it run in the separate virtualized OS, VOS 1 and VOS 2 can be protected from malicious attacks through Internet connections. By running user applications in VOS 2, core services running in VOS 1 can be isolated from software faults due to bugs in user applications running in VOS 2.

4. RESULTS FROM THE CURRENT PROTOTYPE

We developed a prototype virtualized OS environment on Linux/RK [8] and by adapting its resource management mechanisms to UML (User-Mode Linux) [1].We call UML with the resource management mechanisms UML/RK. This section evaluates our current prototype and shows a virtualized OS environment can isolate resource management from each other.

First, we show the share of the CPU resource used by a whole virtualized OS environment can be reserved and also limited in order to create an isolated execution environment.

Figure 5 (a) shows the CPU times consumed by all processes that creates a virtualized OS environment of UML/RK in each period of CPU time replenishment. The virtualized OS environment created by UML/RK consisted of the total of 10 processes that includes the kernel, its supporting programs, and its user processes of UML/RK. The consumed times were calculated from the CPU cycles actually executed within each period. UML/RK was booted with the reservation parameter of 20 millisecond CPU time within 50 millisecond period. Figure 5 (a) shows the result of the execution of UML/RK in which a program that executed an infinite busy loop was running. We created 8 disturbing processes also being executed on Linux/RK aside of UML/RK. The disturbing processes executed an infinite busy loop trying to consume CPU cycles as much as possible.

The results show that UML/RK received the reserved CPU times even with disturbing processes running aside of it, and the received CPU times were also limited as specified; thus, the execution of UML/RK was correctly isolated. Next, we show that CPU resource management is possible in a virtualized OS environment. In order to evaluate that CPU times can be effectively reserved and also be enforced by our

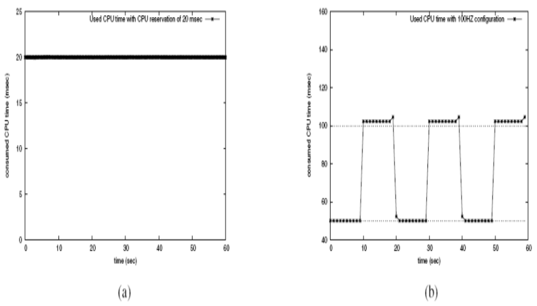

Figure 5. CPU Time Reservation and Enforcement for a Virtualized Operating System Environment

CPU resource management mechanisms in UML/RK, a benchmark program ran with a reservation of the CPU resource in a UML/RK virtualized OS environment.

Figure 5 (b) shows consumed CPU times calculated from the CPU cycles actually executed within each period. UML/RK was booted with the initial reservation parameter of 20 millisecond CPU time within 50 millisecond period. Figure 5 (b) shows the result of the execution started with the reservation of 50 millisecond every 1 second. After 10 seconds, the reservation parameter was changed to 100 millisecond every 1 second. 10 seconds later, the reservation parameter was changed again to 50 millisecond every1 second. It repeatedly changed the reservation parameter every 10 seconds. The results show that UML/RK can rigidly enforce their CPU times and the CPU times obtained by processes change promptly when the reservation parameters are changed.

5. EXPECTED OUTCOME

The user can operate the various applications of different operating system which are available on desktop of user through any device (like smart phones or tablet) ubiquitously using internet. If the user makes any changes to a particular application then the system will synchronization that application on host system.

6. CONCLUSION

This paper gives idea about ubiquitous computing infrastructure architecture that is based on virtualized Operating Systems to provide secure, stable, and isolated computing environment. Our architecture enables ubiquitous devices and ubiquitous servers to be shared securely.

The design presented for a ubiquitous desktop resolves key issues inherent in managing multiple services per user.
Specifically they include

a. Providing multiple services to a user via a matching profile service which centrally manages the relationship between users and their services
b. Higher level authentication to support the access to multiple services using single sign-on policies.
c. Simultaneous mobility of multiple services between thin client devices.
d. Virtualization of services and centralized storage of services to ensure that the virtual machine host technology becomes a scalable commodity allowing future expansion of the infrastructure.

REFERENCES:

[1] Hailei Sun, Tianyu Wo, “Virtual Execution Environment For Windows Applications”, Proceedings of IEEE CCIS, 2011, DOI:10.1109/CCIS.2011.6045094.

[2] Paul Doyle, Mark Deegan, Ciaran O’Driscoll, Michael Gleeson, Brian Gillespie, “Ubiquitous Desktops with Multi-factor Authentication”, IEEE, 978-1-4244-2917, 2008, DOI:10.1109/ICDIM.2008.4746797.

[3] Duy Le and Haining Wang, “An Effective Memory Optimization for Virtual Machine-Based Systems”, IEEE Transactions On Parallel And Distributed Systems, Vol. 22, No. 10, October 2011, DOI:10.1109/TPDS.2011.37.

[4] Deka Ganesh Chandra and Dutta Borah Malaya, “A Study on Cloud OS”, International Conference on Communication Systems and Network Technologies, 2012, DOI:10.1109/CSNT.2012.154.

[5] Helmut Hoyer, Andreas Jochheim, ChristofRöhrig, and Andreas Bischoff, “A Multiuser Virtual-Reality Environment for a Tele-Operated Laboratory”, IEEE Transactions On Education, Vol. 47, No. 1, February 2004, DOI:10.1109/TE.2003.822263.

[6] Lawton, G., “Moving the OS to the Web”, Computer Journal, Vol. 41, No. 3, p.16-19, 2008, DOI:10.1109/MC.2008.94.

[7] Bente,I. Hellmann, B. ; Rossow, T. ;Vieweg, J. ;von Helden, J., “On Remote Attestation for Google Chrome OS”, 15th International Conference on Network-Based Information Systems (NBiS), 2012 , DOI:10.1109/NBiS.2012.55.

[8] Carlos Oliveira, “Enabling Ubiquitous Workplace Through Virtualization Technology” , ICAS 2013, The Ninth International Conference on Autonomic and Autonomous Systems , IARIA, 2013, ISBN:9781-61208-257-8

[9] •Ichiro Satoh, “Location-based services in ubiquitous computing environments” , International Journal on Digital Libraries, Springer-Verlag, June 2006, Volume 6, Issue 3, pp 280-291, DOI:10.1007/s00799-006-0006-1

[10] Herman Slatman, “ Opening Up the Sky: A Comparison of Performance-Enhancing Features in Sky Drive and Dropbox”, 18th Twente Student Conference on IT, University of Twente, Faculty of Electrical Engineering, Mathematics and Computer Science, Enschede, The Netherlands, January 25, 2013.

[11] http://www.vmware.com/virtualization/

[12] • http://www.rcet.org/ubicomp/what.html