A Virtualization Approach in Smart phone Using Cloud computing for machine to machine Communication.

[1]Naushad Ahmad Usmani [2]Mohammed Waseem Ashfaque
[1]Department of IT,Buraimi University College,Buraimi,Oman
[2]Department of IT,Buraimi University College,Buraimi,Oman

ABSTRACT:- Virtualization allows instances of multiple operating systems to run concurrently on a single machine Its means that separating hardware resource from a single operating system and Each "guest" OS is managed and looked after by a Virtual Machine Monitor (VMM), also referred as hypervisor. Because the virtualization system lies between the guest and the hardware resources, it can control the guests OS and use of all hardware resources like CPU, memory, and storage, even guest OS are allowing to switch over from one machine to another. Virtualization and Smart phone or mobile have been two of the greatest trends to hit up enterprises in IT sector. Virtualization from a server perspective has been a disruptive force in the IT world, and these results into the form of VMware, one of the largest software firm in terms of market cap. And hence all over 50% of servers are now being virtualized But implementing some form of virtualization on mobile devices is not yet widely implemented in the enterprise market, and so small companies offering their solution and services, it appears that virtualization on mobile is turning the corner and will be heading towards widespread adoption.

Keywords: - Virtual Machine Monitors, Hypervisors, virtualization, cloud computing.

Introduction:-

Over the last 10 years, the trend in the data center has been towards decentralization, also known as horizontal scaling. Centralized servers were seen as too expensive to purchase and maintain. Due to this expense, applications were moved from a large shared server to their own physical machine, often using commodity hardware. Decentralization helped with the ongoing maintenance of each application, since patches and upgrades could be applied without interfering with other running systems. For the same reason, decentralization improves security since a compromised system is isolated from other systems on the network. However, decentralization's application sandboxes come at the expense of more power consumption, less physical space, and a greater management effort which, together, account for up to $10,000 in annual maintenance costs per machine1. In addition to this maintenance overhead, decentralization decreases the efficiency of each machine, leaving the average server idle 85% of the time2. Together, these inefficiencies often eliminate any potential cost or labor savings promised by decentralization. Virtualization is a modified solution between centralized and decentralized deployments. Instead of purchasing and maintaining an entire computer for one application, each application can be given its own operating system, and all those operating systems can reside on a single piece of hardware. This provides the benefits of decentralization, like security and stability, while making the most of a machine's resources. Modern computers are sufficiently powerful to use virtualization to present the illusion of many smaller *virtual machines* (VMs), each running a separate operating system instance. This has led to a resurgence of interest in VM technology. In this paper we present Xen, a high performance resource-managed virtual machine monitor (VMM) which enables applications such as server consolidation [1,2], co-located hosting facilities [3], distributed web services [4], secure computing platforms[5,6] and application mobility [7,8]. Successful partitioning of a machine to support the concurrent execution of

multiple operating systems poses several challenges. Firstly, virtual machines must be isolated from one another: it is not acceptable for the execution of one to adversely affect the performance of another. This is particularly true when virtual machines are owned by mutually untrusting users. Secondly, it is necessary to support a variety of different operating systems to accommodate the heterogeneity of popular applications. Thirdly, the performance overhead introduced by virtualization should be small. As virtualization disentangles the operating system from the hardware, a number of very useful new tools become available. Virtualization allows an operator to control a guest operating system's use of CPU, memory, storage, and other resources, so each guest receives only the resources that it needs,With virtualized deployments, it is possible to treat computing resources like CPU, memory, and storage as a hangar of resources and applications can easily relocate to receive the resources they need at that time.

LITERATURE REVIEW:-

The Approaches & Overviews:-

1) Virtualization comes in a variety of implementations. In its basic form known as "full virtualization" the hypervisor provides a fully emulated machine in which an operating system can run. VM Ware® is a good example. The biggest advantage to this approach is its flexibility: one could run a RISC based OS as a guest on an Intel-based host. While this is an obvious approach, there are significant performance problems in trying to emulate a complete set of hardware in software. Even with painstaking optimization, it is very difficult to get useful performance from a fully virtualized environment.[9].

2)At the other end of the spectrum is the Single Kernel Image (SKI), in which the host OS spawns additional copies of itself. This kind of virtualization can be found in Swsoft Virtuozzo and Sun® Solaris® Zones. SKI can be thought of as "lightweight" virtualization. While this approach avoids the performance problems with pure emulation, it does so at the expense of flexibility. It is not possible, for instance, to run different versions or even different patch levels of a particular operating system on the same machine. Whatever versions exist in the host, that same software will be provided in the guest. SKI also sacrifices the security and reliability provided by other virtualization methods. If the kernel is exploited, all OS instances resident on the system will be compromised.[9].

3)"Para virtualization," found in the XenSource® open source Xen product, attempts to reconcile these two approaches. Instead of emulating hardware, paravirtualization uses slightly altered versions of the operating system which allows access to the hardware resources directly as managed by the hypervisor. This is known as hardware-assisted virtualization, and improves performance significantly. In order to retain flexibility, the guest OS is not tied to its host OS. Drastically different operating systems can be running in a hypervisor at the same time, just as they can under full virtualization. In this way, para virtualization can be thought of as a low-overhead full virtualization [9].

Xen:- With the release of Xen 3.0, virtualization reaches maturity. Xen is the first virtualization solution to support Intel's VT technology which permits each guest OS to run at full processor speed, with only 0.5% to 3.5% overhead typically incurred by the virtualization process. Guests can be migrated from one machine to another in less than 100ms. Through the hypervisor, operators can control the use of CPU, memory, block, and I/O devices dynamically.. Xen enables users to dynamically instantiate an operating system to execute whatever they desire. In the XenoServer project [10, 11] we are deploying Xen on standard server hardware at economically strategic locations within ISPs or at Internet exchanges. We perform admission control in this direction elsewhere [12]; this paper focuses on the VMM

There are a number of ways to build a system to host multiple applications and servers on a shared machine. Perhaps the simplest is to deploy one or more hosts running a standard operating system such as Linux or Windows, and then to allow users to install applications being provided by conventional OS .More importantly, such systems do not adequately support performance isolation; the scheduling priority, memory demand, network traffic and disk accesses of one process impact the performance of others.[13],Linux/RK [14], QLinux [15] and SILK [16].[17] within the

operating system. Performing multiplexing at a low level can mitigate this problem, as demonstrated by the Exo kernel [18] and Nemesis [19] operating systems. Unintentional or undesired interactions between tasks are minimized. We use this same basic approach to build Xen, which multiplexes physical resources at the granularity of an entire operating system and is able to provide performance isolation between them. In contrast to process-level multi plexing this also allows a range of guest operating systems.

APPROACH & OVERVIEW(XEN):-

In a traditional VMM the virtual hardware exposed is functionally identical to the underlying machine [20]. Although *full virtualization* has the obvious benefit of allowing un modified operating systems to be hosted, it also has a number of drawbacks. This is particularly true for the prevalent IA-32, or *x86*, architecture. Support for full virtualization was never part of the x86 architectural design. Certain supervisor instructions must be handled by the VMM for correct virtualization, but executing these with insufficient privilege fails silently rather than causing a convenient trap [21]. Efficiently virtualizing the x86 MMU is VMware's ESX Server [22] dynamically rewrites portions of the hosted machine code to insert traps wherever VMM intervention might be required. This translation is applied to the entire guest OS kernel (with associated translation, execution, and caching costs) OS to better support time-sensitive tasks, and to correctly handle TCP timeouts and RTT estimates, while exposing real machine addresses allows a guest OS to improve performance by using super pages [23] or page coloring [24]. We avoid the drawbacks of full virtualization by presenting a virtual machine abstraction that is similar but not identical to the underlying hardware an approach which has been dubbed para virtualization[25].We still the discussion so far into a set of design principles:

1. Support for unmodified application binaries is essential, or users will not transition to Xen. Hence we must virtualized all architectural features required by existing standard ABIs.
2. Supporting full multi-application operating systems is important, as this allows complex server con_gurations to be virtualized within a single guest OS instance.
3. Paravirtualization is necessary to obtain high performance and strong resource isolation on uncooperative machine architectures such as x86.
4. Even on cooperative machine architectures, completely hiding the effects of resource virtualization from guest OS risks both correctness and performance. Firstly, Denali does not target existing ABIs, and so certain architectural features from their VM interface. For example, Denali does not fully support x86 segmentation although it is exported (and widely used1) in the ABIs of NetBSD, Linux, and Windows[18]. Hence each virtual machine essentially hosts a single-user single-application unprotected operating system.. In Xen,[26], we are unaware of any published technical details or evaluation. Thirdly, in the Denali architecture the VMM performs all paging to and from disk. This is perhaps related to the lack of memory management support at the virtualization layer. Paging within the VMM is contrary to our goal of performance isolation: [27]).Finally,In the following section we describe the virtual machine abstraction exported by Xen and discuss how a guest OS must be modified to conform to this.

Red Hat and Xen Adoption:- Given the current level of performance and maturity, and the possibilities it provides, Xen is the undisputed leader in open-source virtualization. Dozens of corporations and universities are involved in the project, including Red Hat, IBM® , Oracle® , Intel® , AMD® , Cisco® , and Veritas® . Red Hat has been an early adopter of Xen, an active contributor, and has already incorporated the code into its Fedora distribution. Xen is also a crucial component of Red Hat Enterprise Linux® 5. Virtualization is only one part of Red Hat's larger strategy to commoditize each major computing component, and make it simple for administrators to bring computing resources to the application that needs them. With the advent of the Global File System (GFS), storage is an easily allocated resource and still delivers a high level of performance. Red Hat Network (RHN) allows administrators to treat systems as generic resources that can be easily installed, upgraded, asked, and reallocated.

The Virtual Machine Interface:- Table represents an overview of the para virtualized x86 interface, factored into three broad aspects of the system: memory management, the CPU, and device I/O. In the following we address each machine subsystem in turn, and discuss how each is presented in our para virtualized architecture. Note that although certain parts of our implementation, such as memory management, are specific to the x86, many aspects (such as our virtual CPU and I/O devices) can be readily applied to other machine architectures.

Memory management:- Virtualizing memory is undoubtedly the most difficult part of para virtualizing an architecture, both in terms of the mechanisms required in the hypervisor and modifications required to port each guest OS. The task is easier if the architecture provides a software managed TLB as these can be efficiently virtualized in a simple manner., including Alpha, MIPS and SPARC. Associating an address-space identifier tag with each TLB entry allows the hypervisor and each guest OS to efficiently coexist in separate address spaces because there is no need to push the entire TLB when transferring execution. Unfortunately, x86 does not have a software-managed TLB; instead TLB misses are serviced [28].

CPU:-Virtualizing the CPU has several implications for guest OSes. Principally, the insertion of a hypervisor below the operating system violates the usual assumption that the OS is the most privileged entity in the system. In order to protect the hypervisor from OS misbehavior (and domains from one another) guest OSes must be modified to run at a lower privilege level. Many processor architectures only provide two privilege levels. In these cases the guest OS would share the lower privilege level with applications. The guest OS would then protect itself by running in a separate address space from its applications, and indirectly pass control to and from applications via the hypervisor to set the virtual privilege level and change the current address space. Again, if the processor's TLB supports address-space tags then expensive TLB, OS code typically executes in ring 0 because no other ring can execute privileged instructions, while ring 3 is generally used for application code.

1. This prevents the guest OS from directly executing privileged instructions, yet it remains safely isolated from applications running in ring 3. Privileged instructions are para virtualized by requiring them to be validated and executed within, which would normally read the faulting address from a privileged processor register (CR2); since this is not possible, we write it into an extended stack frame2. When an exception occurs while executing outside ring 0, Xen's handler creates a copy of the exception stack frame on the guest OS stack and returns control to the appropriate registered handler.

I/O Device:- Rather than emulating existing hardware devices, as is typically done in fully-virtualized environments, Xen exposes a set of clean and simple device abstractions. This allows us to design an interface that is both efficient and satisfies our requirements for protection and isolation. To this end, I/O data is transferred to and from each domain via Xen, using shared-memory, asynchronous buffer descriptor rings.

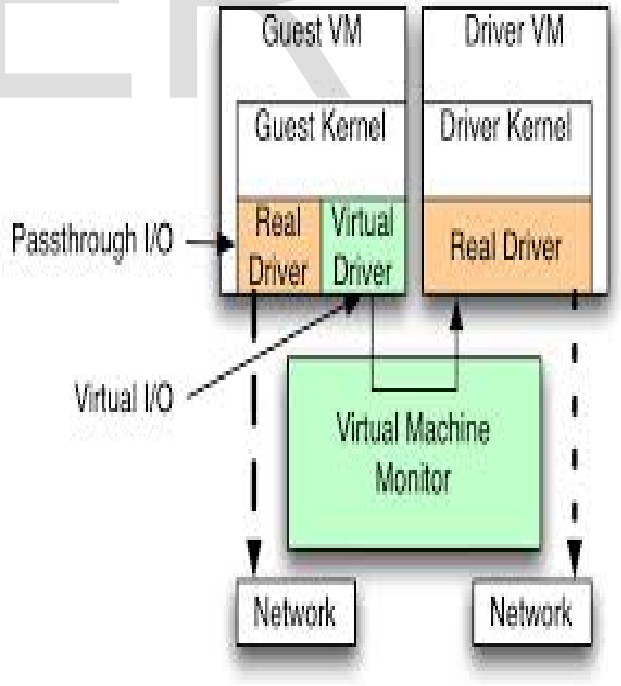

Table 1: The par virtualized x86 interface.

Memory Management Segmentation	Cannot install fully-privileged segment descriptors and cannot overlap with the top end of the linear address space.
Paging	Guest OS has direct read access to hardware page tables, but updates are batched and validated by the hypervisor. A domain may be allocated dis contiguous machine pages
I/O Device Network, Disk, etc.	Virtual devices are elegant and simple to access. Data is transferred using asynchronous I/O rings. An event mechanism replaces hardware interrupts for notifications.
CPU	Guest OS may install a `fast' handler for system calls, allowing direct calls from an application
Protection Exceptions	Guest OS must register a descriptor table for exception handlers with Xen. Aside from page faults,
System Calls	Guest OS must run at a lower privilege level than Xen. the handlers remain the same.
Interrupts	Hardware interrupts are replaced with a lightweight event system.

allowing Xen to efficiently perform validation Checks (for example, checking that buffers are contained within a domain's memory reservation). Similar to hardware interrupts, These callbacks can be `held off' at the discretion of the guest OS. to avoid extra costs incurred by frequent wake-up notifications.

Augmenting Smart phones through Computational Clouds:- MCC implements a number of augmentation procedures for leveraging resources and services of cloud datacenters. Examples of the augmentations strategies include; screen augmentation, energy augmentation, storage augmentation and application processing augmentation of SMD [29]. A number of online file storage services are available on cloud server which augments the storage potentials by providing off-device storage services. Examples of the cloud storage services include Amazon S3 and DropBox. Mobile users outsource data storage by maintaining data storage on cloud server nodes.. [30].Similarly, the computing power of the cloud datacenters is utilized by outsourcing computational load to cloud server nodes. Off loading or cyber foraging. Smart mobile devices implement process offloading to utilize the computing power of the cloud. The term cyber foraging is introduced by Satyanarayanan[31].The mechanism of outsourcing computational load to remote surrogates in the close proximity is called cyber foraging [32]. Researchers extend process offloading algorithms for Pervasive Computing [33], Grid Computing [34] and Cluster Computing [35]. In recent years, a number of cloud server based application offloading frameworks are introduced for outsourcing computational intensive components of the mobile applications partially or entirely to cloud datacenters. Mobile applications which are attributed with the features of runtime partitioning are called elastic mobile applications. Elastic applications are partitioned at runtime for the establishment of distributed processing platform.

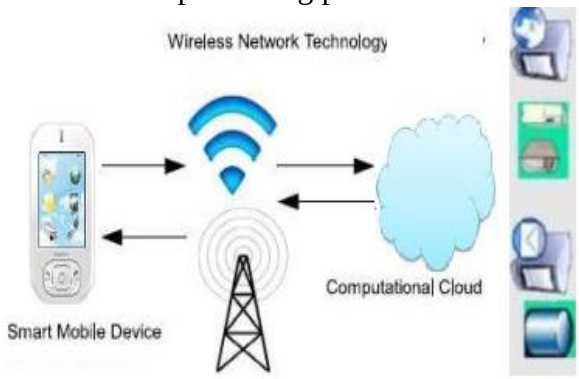

Figure 1: for cloud system architecture

APPROACH FOR SMART PHONE:- The current approaches for SMDs employ a number of strategies for the establishment of runtime distributed application execution platform. This section provides thematic taxonomy for current approaches and

reviews the traditional approaches on the basis of framework nature attributes of the taxonomy. Further, it investigates the advantages and critical aspects of current approaches for SMDs. A classification of application offloading frameworks by using their attributes is shown in Fig. 3. This section analyzes current application offloading frameworks and investigates the implications and critical aspects of current approaches[36].

VIRTUALIZATION OF NETWORK:- Network virtualization enables multiple logical networks to share the physical resources of the underlying network infrastructure. This network model introduces flexibility to the Internet ossification by separating the network architecture functionalities into the following entities:

• Network Infrastructure (NI): provides the physical components required to setup the network (e.g., routers and links). NI efficiently allocates the required network bandwidth and physical resources.

• Virtual Networks (VN): deploy customizable network protocols by leasing the required infrastructure resources from multiple NIs. Each virtual network is a combination of multiple virtual routers and links. When initiating a service, the VN confines to the Service Level Agreements (SLA) with set of NIs and receives the requested resources [36].

• End Users: are similar to the current Internet architecture but have the opportunity to choose from multiple virtual network services.

For any virtual network, the above architectural separation reduces the cost involved in setting up the physical resources and maintaining them. This three-tier architecture promises to introduce flexibility through programmability, improved scalability and reduction in maintenance costs. two virtual networks sharing the network infrastructure resources[36]. Both VNs deploy their customized network services on the shared infrastructure components and establish end-to end connectivity between end users. The VN assumes inherent provision of security features by the hosting NI and is oblivious to the malicious activities of the infrastructure[36].

Security Issues in virtualized Networks:- Network security is an important challenge to be addressed when adapting to new architectural innovations Figure shows the possible combinations in which attacks can compromise different entities in the architecture. For example, (1) indicates the scenario when a malicious VN service launches attacks on the end users[36].

Developing the mobile phone application and cloud servlets:- To build the application Eclipse IDE was used because of its good support for Java development and easy integration with Android SDK and Google App Engine SDK. The build started out by developing the three tests and to execute them in a Java environment. When they were completed the timer class was developed to measure how many milliseconds each test took to execute, Appendix 3. When the tests and timer class were completed, three servlets were developed and uploaded to Google App Engine as separate applications.The list sorting tests required a text file of words to be uploaded and then sorts and outputs the list in alphabetical order. The image transformation test required an image to be uploaded, which are transformed and returned. Because the application already executed all tests automatically the application would only need to load, execute and upload the results, When starting the application a loading screen is displayed while the text list and other features are loaded into the memory of the mobile phone. When this is done a start and an exit button are displayed. The start button executes the tests and displays the progress of which test is being executed. The exit button quits the application. To measure the bandwidth ratio the phone connection type was registered and a class that uploads and downloads the image in a class that uploads and downloads the image in the Assets folder was created [37].

Figure Screenshots from the start screen and from the execution of the application.

The application worked properly but in case something went wrong eventual error messages was saved. In addition, a control test was included which would make sure that the three tests got the right results, for example when the 1500 words were sorted in alphabetical order the Number hundred word had to be the correct one. To increase the security of the servlets and

Prohibit the possibility to send false results, numerical keys were generated and were required to execute the tests on Google App Engine. Every time the application runs it sends a password to the servlet, allowing it to be used. Finally the results from the tests, the information about phone model etcetera were uploaded and saved in a database at Google App Engine [37].

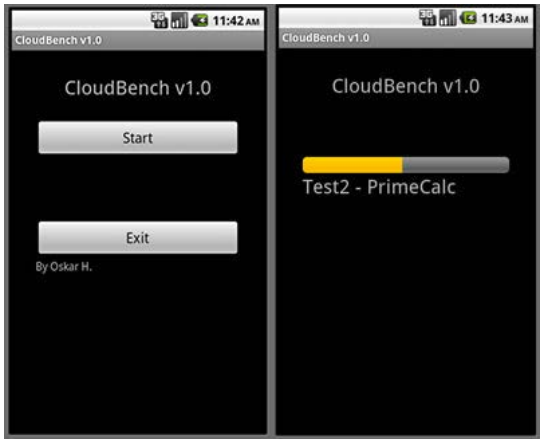

Figure 2: Screenshots from the start screen and from the execution of the application

Application walkthrough:-

Figure 3:The following steps constitute the application:The application icon is pressed on the mobile phone.

2. The loading screen is shown:-

a) The application gathers the mobile phone information.

b) The application gets the text file from the Assets folder and loads it into memory.

c) The application gets the image file from the Assets folder and loads it into memory [37].

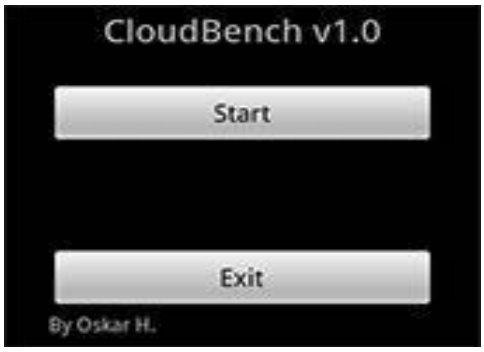

Figure 4: The start and exit button is shown which gives the possibility to start the tests or exit the application

Challenges in Virtual networks:-

Virtual networks introduce unique challenges when compared with the traditional networking requirements. identified with certain level of data transparency between the hosted VNs and the NIs. Our attack scenarios indicate that the underlying infrastructure can introduce biased management practices, monitor confidential information, or launch hidden attacks. Hence the problem of identifying a mechanism to securely process the packets without exposing the input data is required.

• **Global Connectivity:-** To setup end to end network connectivity, the virtual network service should partner with multiple infrastructure providers with varying levels of agreements and requirements.

• **Forwarding Rate:-** High data rate forwarding requirements in the routers imposes significant challenge when extra processing is introduced by the security mechanisms. Most services require certain level of Quality of Service such as low latency with reliable packet processing.

To meet such demands, the computation complexity introduced by the proposed security mechanisms should ensure that the forwarding data rate is not compromised. To address the above challenges, a secure system should provide the following fundamental principle.

Defense Mechanism: Confidentiality:-

The mutual distrust between the participating entities in the network virtualization architecture raises the question of confidentially and privacy of the processed data. Considering. Encrypted Protocol Processing. the possible vulnerabilities as discussed in Section IV, the VN does not

want to expose the data packet (header and payload) when processed by the NI. Encryption techniques are effective to ensure the confidentiality of the data traffic when processed by third party network infrastructures. To avoid biased management practices by ISPs, Bit Torrent protocol versions introduced MSE based protocol encryption that enhances privacy and confidentiality[38][39][40].All processing functions are performed in the encrypted domain and hence the infrastructure is completely oblivious to the data being processed

1) **Trust and Accountability**: Trusted computing ensures consistency in expected behaviors between participating entities.

[41] Proposes a trust management framework that gathers feedback from past experiences of hosting virtual network services and measures the degree of involvement in terms of nodes and links. [42] Proposes to modify the network interface cards to support better detection capabilities using processor extensions and shows inherent assurance of a trusted, accountable platform. Considering the attack space discussed in Section the above solutions lack the dynamics to adapt and protect from attacks. Ideally a monitoring scheme that dynamically tracks the working of the entities in runtime is suitable to ensure effective information integrity provision.

2) **Monitoring**: To identify the biased monitoring practices introduced by ISPs, [43] uses causal inference techniques by passively collecting performance data from clients. To isolate malicious routers, [44] uses a distributed detection technique involving neighboring routers to identify the anomalous behavior of a malicious router.

CONCLUSION:-

Now a day's Various computing technologies are enabled with Internet, wireless sensors, personal computers, and mobile devices are coming together to create machine-to-machine communications. As Smart phone devices carrying the approaches towards capabilities and extensibility of standard desktop workstations and servers, mobile devices are also beginning to face many of the same security threats as desktops and servers experience. In This paper we tried to cover the concept about cloud computing, and mobile cloud computing in terms of Smart mobile computing and explained the various approaches to reach nearer smart phone resources based on availability of resources in the cloud. It has been also discussed that various challenges, issues for compatible applications being distributed for machine to machine. No doubt virtualization and clouding has got a quit significant attention the recent year.During our discussion security issues, vulnerabilities in virtual network also brought into our consideration. Though its mechanism and design is quite different from the running network and internet ,We hope that these observations will provides a vital steps towards more brief ideas which will carry towards its real solution to secure network virtualization in the forth coming year.

References:-

[1] C. A. Waldspurger. Memory resource management in VMware ESX server. In Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI 2002), ACM Operating Systems Review, Winter 2002 Special Issue, pages 181.194, Boston, MA, USA, Dec. 2002.

2] Connectix. Product Overview: Connectix Virtual Server, 2003.
http://www.connectix.com/products/vs.html

[3] Ensim. Ensim Virtual Private Servers, 2003.
http://www.ensim.com/products/materials/
datasheet_vps_051003.pdf.

[4] A. Whitaker, M. Shaw, and S. D. Gribble. Denali: Lightweight Virtual Machines for Distributed and Networked Applications. Technical Report 02-02-01, University of Washington, 2002.

[5] G. W. Dunlap, S. T. King, S. Cinar, M. Basrai, and P. M. Chen. ReVirt: Enabling Intrusion Analysis through Virtual-Machine
Logging and Replay. In Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI 2002), ACM Operating Systems Review, Winter 2002 Special Issue, pages 211.224, Boston, MA, USA, Dec. 2002

[6] T. Gar_nkel, M. Rosenblum, and D. Boneh. Flexible OS Support and Applications for Trusted Computing. In Proceedings of the 9th
Workshop on Hot Topics in Operating Systems, Kauai, Hawaii, May 2003

[7] M. Kozuch and M. Satyanarayanan. Internet Suspend/Resume. In Proceedings of the 4th

IEEE Workshop on Mobile Computing Systems and Applications, Calicoon, NY, Jun 2002.
[8] C. P. Sapuntzakis, R. Chandra, B. Pfaff, J. Chow, M. S. Lam, and M. Rosenblum. Optimizing the Migration of Virtual Computers. In Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI 2002), ACM Operating Systems Review, Winter 2002 Special Issue, pages 377.390, Boston, MA, USA, Dec. 2002.
[9]http://.xensource.com/files/xensource_wp2.pdf
[10] K. A. Fraser, S. M. Hand, T. L. Harris, I. M. Leslie, and I. A. Pratt. The Xenoserver computing infrastructure. Technical Report UCAM-CL-TR-552, University of Cambridge, Computer Laboratory, Jan. 2003.
[11] D. Reed, I. Pratt, P. Menage, S. Early, and N. Stratford. Xenoservers: accounted execution of untrusted code. In Proceedings of the 7th Workshop on Hot Topics in Operating Systems, 1999.
[12] S. Hand, T. L. Harris, E. Kotsovinos, and I. Pratt. Controlling the XenoServer Open Platform, April 2003.
[13] L. Peterson, D. Culler, T. Anderson, and T. Roscoe. A blueprint for introducing disruptive technology into the internet. In Proceedings of the 1st Workshop on Hot Topics in Networks (HotNets-I), Princeton, NJ, USA, Oct. 2002.
[14] S. Oikawa and R. Rajkumar. Portable RK: A portable resource kernel for guaranteed and enforced timing behavior. In Proceedings of the IEEE Real Time Technology and Applications Symposium, pages 111.120, June 1999.
[15] V. Sundaram, A. Chandra, P. Goyal, P. Shenoy, J. Sahni, and H.M.Vin. Application Performance in the QLinux Multimedia Operating System. In Proceedings of the 8th ACM Conference on Multimedia, Nov. 2000.
[16] A. Bavier, T. Voigt, M. Wawrzoniak, L. Peterson, and P. Gunningberg. SILK: Scout paths in the Linux kernel. Technical Report 2002-009, Uppsala University, Department of Information Technology, Feb. 2002.
[17] D. Tennenhouse. Layered Multiplexing Considered Harmful. In Rudin and Williamson, editors, Protocols for High-Speed Networks, pages 143.148. North Holland, 1989.
[18] M. F. Kaashoek, D. R. Engler, G. R. Granger, H. M. Brice.no,R. Hunt, D. Mazi_eres, T. Pinckney, R. Grimm, J. Jannotti, and K. Mackenzie. Application performance and exibility on Exokernel systems. In Proceedings of the 16th ACM SIGOPS Symposium on Operating Systems Principles, volume 31(5) of ACM Operating Systems Review, pages 52.65, Oct. 1997.
[19] I. M. Leslie, D. McAuley, R. Black, T. Roscoe, P. Barham, D. Evers, R. Fairbairns, and E. Hyden. The design and implementation of an operating system to support distributed multimedia applications. IEEE Journal on Selected Areas in Communications,14(7):1280.1297, Sept. 1996.
[20] L. Seawright and R. MacKinnon. VM/370 . a study of multiplicity and usefulness. IBM Systems Journal, pages 4.17, 1979.
[21] J. S. Robin and C. E. Irvine. Analysis of the Intel Pentium's ability to support a secure virtual machine monitor. In Proceedings of the 9th USENIX Security Symposium, Denver, CO, USA, pages 129.144, Aug. 2000.
[22] S. Devine, E. Bugnion, and M. Rosenblum. Virtualization system including a virtual machine monitor for a computer with a segmented architecture. US Patent, 6397242, Oct. 1998.
[23] J. Navarro, S. Iyer, P. Druschel, and A. Cox. Practical, transparent operating system support for superpages. In Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI 2002), ACM Operating Systems Review, Winter 2002 Special Issue, pages 89.104, Boston, MA, USA, Dec. 2002.
[24] R. Kessler and M. Hill. Page placement algorithms for large real-indexed caches. ACM Transaction on Computer Systems, 10(4):338.359, Nov. 1992.
[25] A. Whitaker, M. Shaw, and S. D. Gribble. Denali: Lightweight Virtual Machines for Distributed and Networked Applications. Technical Report 02-02-01, University of Washington, 2002.
[26] A. Whitaker, M. Shaw, and S. D. Gribble. Scale and performance in the Denali isolation kernel. In Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI 2002), ACM Operating Systems Review, Winter 2002 Special Issue, pages 195.210, Boston, MA, USA, Dec. 2002.

[27] S. Hand. Self-paging in the Nemesis operating system. In Proceedings of the 3rd Symposium on Operating Systems Design and Implementation (OSDI 1999), pages 73.86, Oct. 1999.
[28] D. Engler, S. K. Gupta, and F. Kaashoek. AVM: Application-level virtual memory. In Proceedings of the 5th Workshop on Hot Topics in Operating Systems, pages 72.77, May 1995.
[29] S. Abolfazli, Z. Sanaei, and A. Gani, "Mobile cloud computing: A review on smartphone augmentation approaches," in Proc. 1st International Conference on Computing, Information Systems and Communications, 2012.
[30] W. Zheng, P. Xu, X. Huang, and N. Wu, "Design a cloud storage platform for pervasive computing environments," Cluster Computing, vol. 13, pp. 141–151, 2010.
[31] M. Satyanarayanan, "Pervasive computing: Vision and challenges," IEEE Pers. Commun., vol. 8, no. 4, pp. 10–17, 2001.
[32] S. Goyal and J. Carter, "A lightweight secure cyber foraging infrastructure for resource-constrained devices," in Mobile Computing Systems and Applications, 2004. WMCSA 2004. Sixth IEEE Workshop on. IEEE, 2004, pp. 186–195.
[33] J. Oh, S. Lee, and E. Lee, "An adaptive mobile system using mobile grid computing in wireless network," Computational Science and Its Applications-ICCSA 2006, pp. 49–57, 2006.
[34] C. Li and L. Li, "Energy constrained resource allocation optimization for mobile grids," Journal of Parallel and Distributed Computing, vol. 70, no. 3, pp. 245–258, 2010.
[35] Y. Begum and M. Mohamed, "A dht-based process migration policy for mobile clusters," in Information Technology: New Generations (ITNG), 2010 Seventh International Conference on. IEEE, 2010, pp. 934–938.
[36] Security Issues in Network Virtualization for the Future Internet ,Sriram Natarajan and Tilman Wolf, Department of Electrical and Computer Engineering University of Massachusetts, Amherst, MA, USA.
[37]Mobile phones and cloud computing A quantitative research paper on mobile phone application offloading by cloud computing utilization.
[38] Message Stream Encryption, VUZE, http://wiki.vuze.com/w/MessageStream Encryption.
[39] B. B. Brumley and J. Valkonen, "Attacks on message stream encryption," in Proceedings of the 13th Nordic Workshop on Secure IT Systems— NordSec '08, H. R. Nielson and C. W. Probst, Eds., October 2008, pp. 163–173.
[40] C. Gentry, "Fully homomorphic encryption using ideal lattices," in Proceedings of the 41st annual ACM symposium on Theory of computing, ser. STOC '09. New York, NY, USA: ACM, 2009, pp. 169– 178. [Online]. http://doi.acm.org/10.1145/1536414.1536440
[41] L. Mekouar, Y. Iraqi, and R. Boutaba, "Incorporating trust in network virtualization," in Proceedings of the 2010 10th IEEE International Conference on Computer and Information Technology, ser. CIT '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 942–947. [Online]. Available: http://dx.doi.org/10.1109/CIT.2010.174
[42] E. Keller, R. B. Lee, and J. Rexford, "Accountability in hosted virtual networks," in Proc. of the First ACM SIGCOMM Workshop on Virtualized Infrastructure Systems and Architectures (VISA), ser. VISA '09, Barcelona, Spain, Aug. 2009, pp. 29–36.
[43] M. B. Tariq, M. Motiwala, N. Feamster, and M. Ammar, "Detecting network neutrality violations with causal inference," in Proceedings of the 5th international conference on Emerging networking experiments and technologies, ser. CoNEXT '09. New York, NY, USA: ACM, 2009, pp. 289–300. [Online]. Available: http://doi.acm.org/10.1145/ 1658939.1658972
[44] A. T. Mizrak, Y.-C. Cheng, K. Marzullo, and S. Savage, "Detecting and isolating malicious routers," IEEE Transactions on Dependable and Secure Computing, vol. 3, no. 3, pp. 230–244, Jul-Sep 2006.