
Flexible virtual machines in distributed systems and grids

Dohan Kim

**School of Computer Science
University of Windsor
Windsor, Ontario, Canada
kimw@uwindsor.ca**

Table of Contents

Abstract	4
1. Introduction	5
1.1 Virtual Machines.....	5
1.2 Distributed Systems and Grids.....	5
1.2.1 Distributed Systems.....	5
1.2.2 Grids.....	6
1.3 Virtual Machines in Distributed systems and Grids.....	7
1.4 Outline of the Survey.....	7
2. Key Concepts	8
2-1 Category of virtual machine.....	8
2-1-1 OS-level VMs and Virtual Machine Monitor (VMM).....	9
2-1-2 Virtual OS.....	11
2-1-3 Language-level VMs.....	12
2-2 Hardware Virtualization.....	12
2-2-1 CPU Virtualization.....	12
2-2-2 Memory and I/O Virtualization.....	13
2-2-3 Full Virtualization.....	14
2-2-4 Paravirtualization.....	15
2-3 Network Virtualization.....	17
2-3-1 Programmable Network.....	17
2-3-2 Network Components Virtualization.....	18
2-4 Resource Isolation.....	19
3. Design and Implementation	20
3-1 VM Data Management in Grid Computing.....	20
3-2 Process / VM Migration.....	23
3-3 Case Studies.....	24
3-3-1 Grid Computing on OS-level Virtual Machines.....	24
3-3-2 Virtual OS based Distributed systems.....	26
3-4 Testbed Studies.....	27
4. Concluding Remarks	29

Appendix-I	31
Appendix-II	43
Appendix-III	69
Appendix-IV	71
Appendix-V	73
Appendix-VI	75

List of Figures

Figure 1	ISA/ABI Interface (Smith, 2001, page : 7).....	8
Figure 2	Virtual switch and steps of port creation (Jiang c, 2003, page : 5).....	19
Figure 3	Simple Virtual Private Grid File System (Figueiredo c, 2003, 9).....	22
Figure 4	Architecture for a VM-based Grid Service (Figueiredo a, 2003, page : 12)....	25
Figure 5	Virtual Internetworking on Overlay Infrastructure (Jiang c, 2003, page 2)....	28

Abstract

Virtual machine technology, originally developed in the 1960's, is evolving to a new phase by the development of distributed systems and grids. Virtual machine technology has been recently revived as 'flexible', supporting on-demand creation, migration, and dynamic instantiation through real or virtual networks. This survey will cover general virtual machines, with emphasis on 'flexible' virtual machines in distributed systems and grids. This survey will also cover virtual machines in virtual networks (as decoupled from physical machines in physical networks) and their allowance for various services in distributed systems and grids, providing simplified views by raising the level of abstraction.

1 Introduction

1-1 Virtual Machines

According to Goldberg (Goldberg b, 1974), a ‘virtual machine’ (VM) is defined as, “an isolated duplicate of a real existing computer system, in which statistically-dominant subset of the virtual processor’s instructions be executed on host processor in a native way”. Virtual machines have been studied since the late 1960s and are experiencing a resurgence in commercial and research areas (Harris, 2001, King c, 2002). OS-level virtual machines (Goldberg a, 1974, Creasy, 1981, Waldspurger, 2002), in addition to virtual OSes (Dike, 2001, Jiang b, 2004), and language-level virtual machines (Bowles, 1978, Lindholm, 1997, Thomas, 1999) can be deployed in distributed systems and grid environments (Figueiredo a, 2003, Chien a, 2003, Sirer, 1999). OS-level virtual machines allow multiple-guest operating systems to be hosted on the same hardware platform (Bugnion, 1997), as do virtual OSes (Jiang b, 2004), with language-level virtual machines allowing the same application code to be used on any system (Harris, 2001) that supports the appropriate virtual machine.

1-2 Distributed Systems and Grids

1-2-1 Distributed Systems

Tanenbaum (Tanenbaum, 1995) defines a distributed system as, “a collection of independent computers that appear to the users of the system as a single computer”, and clarified two aspects with this definition. The first clarification concerns hardware, where the distributed system is composed of a collection of independent computers, and the second clarification concerns software; the user can think of the distributed system as one single computer (Tanenbaum, 95).

1-2-2 Grids

According to Foster (Foster a, 2001), distributed-computing technology should evolve to accommodate the range of resource types (including the flexibility and control in sharing relationships) to establish dynamic collections of individuals, institutions, and resources; a ‘grid problem’ is defined as “coordinated resource-sharing and problem-solving in dynamic, multi-institutional virtual organizations”. These dynamic collections are called ‘Virtual Organizations’ or VOs (Foster a, 2001, Casanova, 2002). Research and development efforts to build scalable VOs have been conducted in grid communities, but a challenging problem still remains: the achievement of seamless, dynamic, cross-organizational VO sharing (Foster f, 2003). ‘Open grid services architecture’, defined standard mechanisms for the creating, discovering, and naming of ‘grid service instances’, while ‘service’ is defined as a network-enabled entity providing some capability (Foster d, 2002). The benefits of service-oriented grid architectures in combination with web services technologies (Grimshaw b, 1999) have been explained by the former’s ability to make use of advantageous characteristics (such as service description, discovery, and binding of service descriptions to network protocols) of the latter, with these ‘virtualized’

services allowing consistent resource access across multiple heterogeneous environments by location transparency, and also multiple logical resource instances to be mapped onto the same physical resource (Foster d, 2002).

1-3 Virtual Machines in Distributed Systems and Grids

Virtual machines (VMs) can be deployed in distributed systems and grids to provide fault/attack/resource isolation (Bressoud, 1996, Barham, 2003), customization (Whitaker a, 2002), secure logging (King a, 2002), intrusion prevention and detection (Dunlop,2002, Garfinkel, 2003), computation-environment migration (Sapuntzakis, 2002, Osman, 2002, Boyd, 2002), and monitoring/simulating systems (Jiang c, 2003, Sundaraj, 2003, Figueiredo a, 2003).

1-4 Outline of the Survey

This survey covers flexible virtual machines in distributed systems and grids. Section 1 includes the introduction, section 2 covers key concepts to the topic, section 3 covers design and implementation issues such as VM image/data management and migration in distributed systems and grids, including testbed studies for VM-based distributed systems and grid computing. Finally, section 4 presents the concluding remarks. Appendices I to VI contain the bibliography, annotated bibliography, list of researchers, list of upcoming conferences, a cross-referencing graph and an email sent to researchers respectively. Distributed and grid computing by language-level VMs are beyond the scope of this survey.

2 Key Concepts

2-1 Categories of Virtual Machines

A typical computer system has three components; these are hardware, operating system, and application programs. There are two key interfaces in a typical computer system; respectively, these are called ISA and ABI (See Figure 1 below).

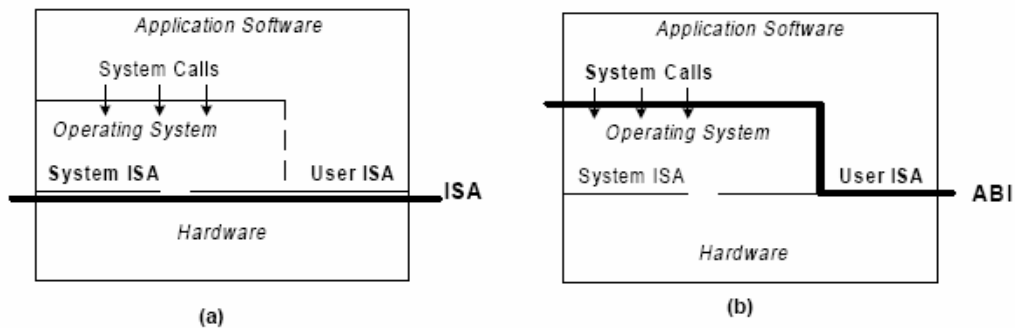


Figure 1 (Smith, 2001, page: 7)

(a) Instruction Set Architecture (ISA) Interface

(b) Application Binary Interface (ABI) Interface

In Figure 1 (a), the ISA interface consists of both the user ISA (non-privileged instruction set architecture) and the system ISA (privileged instruction set architecture). The user ISA is available to both the operating system and application software, whereas the system ISA is only available to the operating system; only privileged operations can be permitted to manipulate the processor, memory and I/O directly (Smith, 2001). As shown

in Figure 1 (b), the main components of ABI interface are the user ISA and ‘system calls’. Between application programs and an operating system, the ‘system calls’ interface is provided to manage and protect hardware resources from unauthorized accesses (Smith, 2001). Two major types of virtual machines are ISA VMs and ABI VMs; ISA VMs manipulate and support both the user ISA and the system ISA, whereas ABI VMs manipulate and support both user ISA and ‘system calls’ (Smith, 2001). OS-level VMs (Waldspurger, 2002) are in the former category but language-level VMs (eg Java Virtual Machine) and virtual OSes (Jiang b, 2004, Dike, 2001) are in the latter.

2-1-1 OS-level VMs and Virtual Machine Monitor (VMM)

OS-level VMs (i.e. classic VMs) are categorized as ISA VMs, with the same ISA execution environments of the entire operating systems (Figueiredo a, 2003). ‘Virtual machine monitor’ (Shriver, 1976) is involved in the OS-level VMs between hardware and guest operating systems (VMs). The original purpose of OS-level virtual machines is to multiplex expensive hardware resources by virtualization of the hardware interface; this virtualization of the hardware layer, called a virtual-machine monitor (VMM), allows multiple, concurrent guest OSes to be hosted on the same hardware platform (Creasy, 1981). ‘Disco’ projects (Bugnion, 1997) and commercial ‘VMware’ (Sugerman, 2001) adapt VMM approaches; ‘Disco’ projects (Bugnion, 1997) adapts VMM to run multiple commodity operating systems on a scalable multiprocessor, whereas ‘VMware’ (Sugerman, 2001) adapts VMM to run several guest OSes on intel-based PC platform. VMM emulates underlying hardware resources for guest OSes, with guest OSes then operating as though they are real hardware; in this case, all I/O and privileged

instructions must be trapped by VMM, with the VMM also ensuring correct scheduling of CPU time slice amongst guest OSes (Creasy, 1981). Other purposes, such as providing isolation and security, were considered for these guest operating systems on VMM; specifically, because of inherent VMM characteristics, it protects other guest operating systems if a malicious user compromises one underlying guest operating system, compelling him or her to break one more level (i.e., VMM) in order to compromise an entire system (Figueiredo a, 2003).

OS-level VMs also provide security monitoring services such as intrusion prevention/detection (Dunlop, 2002, Garfinkel, 2003) and secure logging (King a, 2002) systems. OS-level VMs are useful for intrusion prevention and detection systems; because running doubtful events on real systems (to test attacks) might compromise the system, a better approach is to clone the real system by OS-level VMs in order to test suspicious events (Chen, 2001). Secure logging systems through OS-level VMs (King a, 2002) has been demonstrated. A major shortcoming of current intrusion logging systems can be the fact that an attacker, after he/she takes control over the system, can easily alter the logging system so it cannot be trusted; moving logging software out of the operating system and into the virtual-machine monitor can help to replay the operating system's execution before, during, and after any potential attacker compromises the system (King a, 2002).

Another advantage of OS-level VMs allows running unmodified-legacy applications to migrate and operate seamlessly without residual dependencies, because the entire virtual

machine state can be encapsulated and migrated between VMMs (Osman, 2002, Sapuntzakis, 2002).

2-1-2 Virtual OS

Virtual OS technology (Dike, 2001, Jiang b, 2004,) is similar to OS-level VM technology in that it allows multiple guest operating systems (virtual OSes) to be hosted on the same hardware platform with fault/attack isolation (Jiang b, 2004) but its basic difference lies in the fact there is no hardware emulation layer in the virtual OS technology (Dike, 2001). The virtual OS kernel and its processes run as processes on the host kernel; as a result, a user space virtual machine, using simulated hardware, can be run by the host kernel (Dike, 2001). As shown by Ensim (Ensim, 2003), user Mode Linux (Dike, 2001, Buchacker, 2002) and Linux BSD's Jail (Kamp, 2000), along with most other virtual OS technology, adapt virtualization at the system-call level (Bavier, 2004). Virtualization at the system-call level can be achieved by modification of a kernel code which interacts with hardware (Buchacker, 2002). According to Dike (Dike, 2001), linux virtual OS, running on a linux host, can be implemented by a special tracing process using 'ptrace' linux system-call; processes in user mode will have their system-calls intercepted and virtualized, but in kernel mode processes should be released from the tracing mechanism and directly run into the host kernel. For a linux virtual OS, hardware interactive assembler instructions in virtual OS kernel (such as interrupt, exception handling, and access functions) can be replaced with signal system-calls, which allows multiple user mode kernels on each virtual device to be hosted on a host kernel (Buchacker, 2002).

2-1-3 Language-level VMs

The main purposes of language-level VMs are to provide portability, platform independence, and security (Lindholm, 97). These VMs are mainly deployed for use with application programs rather than operating systems; Java virtual machine (Lindholm, 97) and Mite virtual machine (Thomas, 99) are examples of language-level VMs. These VMs provide non-native instruction sets, exposing only high-level interfaces to the resources of underlying hardware, while allowing separate and independent designing (Harris, 2001) from the application software.

2-2 Hardware Virtualization

2-2-1 CPU Virtualization

CPU virtualization for a virtual machine can be accomplished by timesharing whereby each guest OS, in turn, gains access to a CPU for a certain period of time allowing the schedule policy of VMMs to control context switching amongst OS-level VMs (Creasy, 1981). According to Barham (Barham, 2003), the x86-based architecture for a traditional CPU at privileged levels can be described as a series of concentric rings, with OS code executing in ring 0 (most privileged), ring 3 (least privileged) is used for application purposes, and rings 1 and 2 are seldomly used; for CPU virtualization with VMs in x86-based architecture, VMM should execute in ring 0 while guest OSes (OS-level VMs) should execute in ring 1, thereby preventing guest OSes from directly executing privileged instructions in ring 0, while guest OSes are isolated from running applications.

One method of avoiding CPU-virtualization overhead discussed by Whitaker (Whitaker b, 2002), is to have a guest OS issue a virtual instruction (such as an idle-with-timeout) allowing a guest OS to be removed from scheduler considerations until its timer fires or until a signal arrives, thereby helping a guest OS to avoid wasting its slice of physical CPU by executing OS idle loops

2-2-2 Memory and I/O Virtualization

According to Bugnion (Bugnion, 1997), “a machine address refers to actual hardware memory, while a physical address is a software abstraction used to provide the illusion of hardware memory to a virtual machine”. Robin (Robin, 2000) demonstrated that an extra level of address translation can be used to both virtualize physical memory and control VM physical-to-machine address mappings; physical addresses can be mapped to machine addresses using the TLB (Translation Lookaside Buffer) of the processor while the VMM can protect and manage the page table for each guest OS. A VMM can use the data structure for each VM to control the mapping of physical page numbers to machine page numbers, as whenever a guest OS issues an instruction to access the TLB or its own page table, a VMM can intercept this instruction, preventing the VM from updating actual MMU states; Sugerman (Sugerman, 2001) first introduced ‘shadow page tables’ which can be maintained by a VMM for a processor’s TLB to perform machine-to-physical address mapping, avoiding additional overhead during virtual-to-machine address mapping.

For the purposes of I/O virtualization, Robin(Robin,2000) demonstrated that a VMM should intercept each VMs access to I/O devices and forward them to physical I/O devices in order to virtualize the latter; during this process, one special device driver for each type of device can be used (rather than the real device driver in every I/O device) by first introducing a ‘monitor call’ which directs all command arguments to the VMM into a single trap for simplicity and efficiency.

2-2-3 Full Virtualization

According to Creasy (Creasy, 1981), traditional virtual-machine monitors (VMMs) provide each guest OS (VM) with a full hardware virtualization; virtual hardware exposed to each VM should be functionally the same as the underlying machine so that a virtual-machine monitor can host unmodified multiple operating systems while giving guest OSes (VMs) the illusion they are running directly on physical hardware. For a full hardware virtualization, all hardware-specific instructions by VMs should be intercepted by a VMM; whenever guest OSes (VMs) or applications execute privileged instructions (including hardware instructions) by traps, VMM should intercept these traps prior to a VM interaction with the hardware, and whenever VMs are required to execute non-privileged instructions (such as simple arithmetic operations), those non-privileged instructions are allowed to directly execute on the CPU without VMM intervention (Robin, 2000). Whitaker (Whitaker, 2002) pointed out that traditional mainframe hardware, especially the processor, was designed to be virtualizable, but the Intel IA-32-based processor architecture is not completely virtualizable; some x86 sensitive instructions, which might affect the states of a VMM or other VMs, were not trapped in

user-mode, for example, some x86 instructions (ex. `pushl`, `popl`) access the interrupt-enabled flag in this mode without being trapped. Therefore, full virtualization is not possible in the Intel IA-32-based processor architecture; for full hardware virtualization on Intel-based architecture, inserting manual traps by binary rewriting (thereby emulating full virtualization) was one solution suggested by (Sugerman, 2001).

2-2-4 Paravirtualization

According to Whitaker (Whitaker b, 2002), traditional VMMs with full hardware virtualization demonstrated performance drawbacks because large amounts of memory are consumed by each VM in order for the latter to access its own copy of resources and devices. Xen (Barham, 2003) and Denali (Whitaker a, 2002) systems apply ‘paravirtualization’, which is the virtualization of a subset of the processor’s instruction set with specialized virtual devices to enhance performance. A ‘paravirtualization’ system replaces hardware interrupts with its own event system to provide control transfer between VMM and VMs; for example, Xen systems allow VMs read access to page tables, but a VMM intercepts the write access for updates from VMs by Xen’s trapping mechanism (Barham, 2003) to enhance its performance. An ‘isolation kernel’, similar to a VMM, can also be used as a ‘paravirtualization’ system; the Denali isolation kernel (which provides a simplified interface of an underlying architecture) removes deprecated and rarely used machine instructions and modified some instructions (such as nonvirtualizable instructions in the x86 architecture), then adds particular virtual instructions (thereby enabling guest OSes to be directly executed onto the physical processor in some cases) for the isolation kernel’s instruction set (Whitaker, 2002).

Another example of ‘paravirtualization’ in Denali systems is shown by its replacing of complex BIOS bootstrap functionality of guest OSes with the simple procedure of a VMM loading a VM image into memory (Whitaker a, 2002). For the purpose of minimizing I/O virtualization overhead for each VM, a Denali system drastically reduces the number of I/O devices (supported by guest OSes), keeping only those found in a typical system, such as a network interface card, serial device, keyboard, timer, and a console; thousands of the modified guest OSes (VMs) can be hosted on an ‘isolation kernel’ by this ‘paravirtualization’ approach (Whitaker a, 2002). Using these methods of ‘paravirtualization’, the isolation kernel should be resident in physical memory, while VMs should be paged on demand; whenever page fault is taken by the VM, the isolation kernel verifies the virtual address, allocates new page tables, and initiates a read action from the VM’s swap region (Whitaker a, 2002). For improved performance, an ‘isolation kernel’ should mandate each VM’s access to a subset of virtualized address spaces, with the kernel itself being mapped into those address spaces inaccessible to VMs, thereby avoiding excessive TLB (Translation Look-aside Buffer) flushing onto VM/VMM crossings while providing the means for the sharing of memory between VMs (Peterson a, 2002). An isolation kernel can also choose a select number of active VMs to be in memory; as the remaining VMs are swapped to second storage, this process periodically redistributes physical memory from inactive to active VMs (Whitaker a, 2002). However, if existing native operating systems are to be used in either Xen or Denali systems, drastic porting efforts might be required (Bavier, 2004).

2-3 Network Virtualization

2-3-1 Programmable Network

The goal of a programmable network is to simplify the network services for their deployment (Campbell a, c, 1999). As stated in (Campbell a, 1999), a programmable network decouples control software from communication hardware to virtualize network infrastructures. According to Campbell (Campbell a, 1999), several prototypes of programmable networks have been suggested by a number of research groups. One class of programmable network suggested by Campbell (Campbell c, 1999) is a ‘spawning network’ (whereby a child network operates on a subset of its parents’ network resources, independently performing despite the limitations in their parents’ resource and partitioning models), allowing the creation, deployment, and management of new network topologies, through the virtual network operating system’s ‘life cycle’ (composed of profiling, spawning, and management) process. A virtual network should be defined as a ‘profiling process’ (the selection of topology from the parent link and nodes and the specifying of resource requirements for virtual links, including bandwidth and capacity) before spawning (Campbell c, 1999). The main procedure in the ‘spawning process’ is the dynamic instantiation of profiling scripts (such as setting up topology), the allocation of resources, the creation of virtual network components, and the bootstrapping of network services (Campbell c, 1999). The ‘management process’ in the virtual network operating system’s life cycle depends upon the ‘per-virtual-network’ policy, which allows the management, control, refinement, and monitoring of virtual network

resources (Campbell c, 1999). Campbell (Campbell c, 1999) demonstrated that programmable network can be used to architect, compose and deploy virtual networks by his example of spawning networks.

2-3-2 Network Components Virtualization

Physical network devices can be abstracted as distributed computing objects such as virtual switches (Merwe b, 1997), virtual routers (Campbell b, 1999) and virtual ATM (Merwe a, b, 1997) in a virtual network (Campbell a, 1999). According to Jiang (Jiang d, 2003), virtual network interfaces can be dynamically created, configured or deleted even when the virtual machine is active; when a new request for adding/deleting a virtual network interface arrives, a virtual machine accommodates this new request by renewing VM kernel data structure after proper authentication. A virtual switch is also created for each virtual LAN, with packet forwarding then performed by the former at data link (layer 2) level; the Unix/Linux 'poll' system-call can be used to emulate a physical switch, whereby a UDP Daemon polls the arrival of data and manipulates forwarding or dropping incoming requests (Jiang c, 2003). When the proper VM connect request for virtual LAN arrives, a new port can be allocated for the virtual machine by a virtual switch, enabling a physical connection to be established between the virtual machine host and the virtual switch, as shown in the figure below:

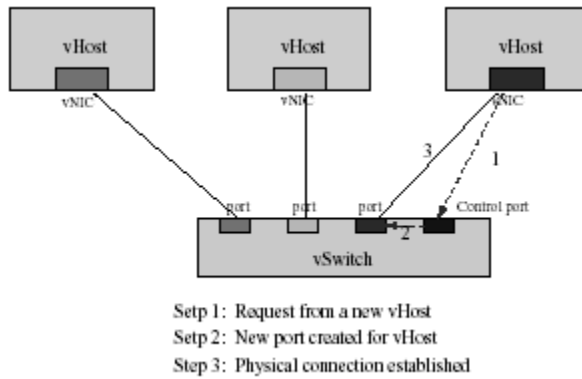


Figure 2 (Jiang c, 2003, page : 5), Virtual switch and steps of port creation

Basic differences between a physical switch and a virtual switch include the lack of hardware constraints to the number of ports possible for the latter, as well as the capability of virtual switches to use various packet queing/forwarding policies which can be dynamically adapted for loss rate, bandwidth, congestion, and delay (Jiang d, 2003). As a result, network component virtualization can avoid the need for a restart when it is used in a dynamic, adaptive VM overlay network (Jiang c, 2003).

2-4 Resource Isolation

By using resource-scheduling algorithms (Rajkumar, 1998), resource isolation for virtual machines can be achieved by resource reservation, allocation, and scheduling; memory allocation limits can be reserved before a virtual machine boots, with other resources allocation such as CPU and network being specified for each virtual machine (Sugarman, 2001). VMs are not allowed to exceed their share of resources; for example, when a process inside a VM needs to perform computational jobs requiring a lot of memory, this VM alone normally would have to swap and experience a low performance, while other

VMs remained unaffected (Dike, 2001). According to (Waldspurger, 2002), the goals of performance isolation and efficient memory utilization often conflict; a possible solution first introduced by (Waldspurger, 2002) to cope with this is to use an ‘idle memory tax’ (defined as reclaiming more idle pages from inactive VMs), which can specify the maximum fraction of idle pages requested from a VM, thereby enabling each VM to use a larger portion of its memory without exceeding full share, while efficiently maintaining memory partition.

3 Design and Implementation

According to (Jiang a, 2003, Sundaraj, 2003), OS-level VMs and Virtual OSES can be deployed in distributed systems and grid environments to provide security, isolation, resource control and site-independence (such as allowing VMs to migrate from resource to resource and instantiate). Figueiredo (Figueiredo a, 2003) insisted that VM-based distributed and grid computing can raise the level of abstraction from operating system users to the OS-level VMs.

3-1 VM Data Management in Grid Computing

Data management in VM-based grid computing involves images (of virtual machines), computation, and data servers (Figueiredo a, 2003). Figueiredo (Figueiredo c, 2003) pointed out that grid computing solutions such as Globus (Foster b, 1996) do not provide common file systems, rather using file-staging techniques as suggested by (Bester, 1999)

to transfer files between user accounts causing limitations to support on-demand transfers. It has been suggested (Figueiredo b, 2001) PVFS or PUNCH (Purdue University Network Computing Hubs) Virtual File Systems, with its logical user accounts, allows data to be transferred on demand between storage (including image servers and data servers) and computation servers. Traditional NFS file systems (Sandberg, 1985) are established by system administrators once, with multiple users accessing only through explicit log-on procedures (Figueiredo b, 2001). PVFS, on the other hand, is both created and terminated dynamically during each client-server session (Figueiredo b, 2001), employing server-side proxies to delegate transactions between NFS clients and servers. These server-side proxies are managed on-demand by grid middleware, differing from other on-demand data-access solutions for grid computing such as Condor (Litzkow, 1992) and Legion (Grimshaw a, 1997) with their ability to be applied on unmodified applications and legacy operating systems (Figueiredo b, 2001).

Virtual Private Grid File Systems (VP/GFS) introduced by (Figueiredo c, 2003) supports on-demand data transfers with private, encrypting channels (using SSH on NFS in grid environment) and integrating server-side, proxy-based PVFS with disk caching by the client-side proxies, eliminating problems such as unnecessary traffic and performance degradation if we transfer including unused data when transferring whole VM images (Schmidt, 2000). The main purpose of client-side proxies in VP/GFS is to cache file system data for enhanced performances (Figueiredo c, 2003) including improved access latency. The following diagram (Figure 3) illustrates simple VP/GFS architecture:

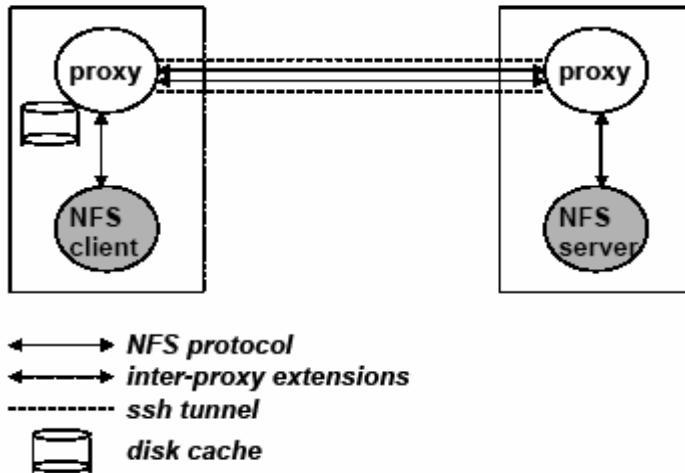


Figure 3 (Figueiredo c, 2003, page: 9), Virtual Private Grid File System Architecture

Another approach to VM image transfer between image and computation servers has been proposed by Sundaraj (Sundaraj, 2003) whereby VNET architecture (a virtual private network) implements a virtual local area network using layer 2 tunneling over a wide area network, allowing a user to transfer VM images to his/her local area as though the virtual machine is on a local area network. Problems associated with layer 3 based VM connectivity, including the growing need for policies such as routable IP address management and firewall forwarding if the number of sites is increased, whereas VNET enables virtual machines to migrate from site to site without an identifiable IP address (Sundaraj, 2003).

3-2 Process / VM Migration

Process migration is defined by Milojevic (Milojevic, 2000) as an “act of transferring a process between two machines” whereby dynamic load distribution, fault resilience, eased system administration, and data access locality are enabled. Process migration at

user-level is deployed in many systems, including Condor (Litzkow, 1992) and MPVM (Casas, 1995), to support cluster computing. Some system-level facilities provided by operating systems (such as inter-process communication) inherently cannot be supported by user-level process migration, whereas object-based process migrations such as those found in Globus (Allen, 2001, Foster b, 1996) and Legion (Grimshaw a, 1997) require programming controls on middleware environments without supporting legacy applications for process migration (Osman, 2002). VM-based process migrations to VMMs have been suggested (Kozuch, 2002, Sapuntzakis, 2002, Osman, 2002) to support legacy applications with a VM by ‘capsules’ (Sapuntzakis, 2002) that can be dynamically instantiated. The advantage of VM migration to VMM for a computing environment is that the latter can encapsulate all volatile execution states of a VM, permitting mobile users to suspend their work in one computer and seamlessly resume their work at another computer (Kozuch, 2002). Migrating all states of a running computing environment (including disks, memory, CPU registers, and I/O devices) across low bandwidth networks has been discussed, with ‘optimized capsules migration’ (copy-on-write disks, which trace only the updates to capsule disks, “ballooning zeros” for any unused memory, hashing, etc) between VMMs (Sapuntzakis, 2002). The mechanism of a ‘ballooning technique’, first introduced by (Waldspurger, 2002), requests from the operating system a large number of unused memory pages, which it then “zeros” to help memory states to be easily compressed. Hashing is used to speed up capsule transfer by checking local storage and examines caches; only the data with different hash values would be transferred to reduce the amount data inside ‘capsule’ (Sapuntzakis, 2002). ‘Remote computational service’ architecture, as discussed by (Schmidt, 2002), can also

be used to help ‘capsule’ migration. ‘Remote computational service’, based on stateless display consoles and cacheable computing sessions, can be connected to session servers via display networks, allowing both the possibilities for active sessions to migrate between session servers, and access to high performance back-end servers which might support clustering and load balancing; users can then access remotely by simply applying low-level, stateless appliance-like consoles while keeping persistent computing sessions (Schmidt, 2002). Figueiredo (Figueiredo a, 2003) insisted that VM-based grid computing can be seamlessly supported by these VM migration technologies.

3-3 Case Studies

3-3-1 Grid Computing on OS-level Virtual Machines

By using a ‘VM life cycle’, the mechanism of grid computing on OS-level virtual machines have been illustrated. Grid computing on an OS-level virtual machine involves a physical, virtual machine O/S image, application image, and user data server (Figueiredo a, 2003). The steps of a ‘VM life cycle’, in grid environments, are as follows:

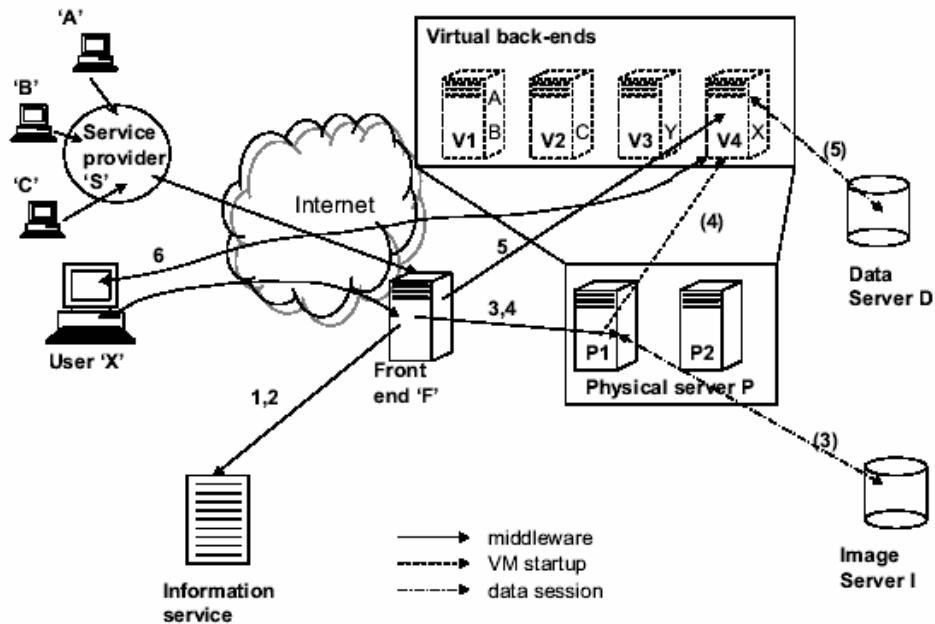


Figure 4 (Figueiredo a, 2003, page: 12), Architecture for a VM-based Grid Service

According to Figure 4, user X (or Service Provider S on behalf of users A, B, and C) queries the availability of resources for use by information services such as Globus MDS, (Foster b, 1996) or GIS (Allen, 2001), using middleware Front end 'F'. Then, grid middleware establishes a data session between physical server P and image server I, either by GridFTP (Foster c, 2002) or on-demand transfers (Figueiredo c, 2003). When the data session is established between physical and image server, the former downloads images from image server and is then able to reduce transfer delay later by caching the VM state; once the download is complete, a physical machine allocates a slice of its resources for a VM image (Figueiredo a, 2003), then instantiates a VM while providing it

with an IP or virtual Ethernet address (Sundaraj, 2003). Virtual back-ends are groups of VMs which are mapped slices of physical machines; data sessions for application downloads are then established between operating systems inside VMs and the application server (Figueiredo a, 2003). These transfers can also be achieved by on-demand transfers (Figueiredo c, 2003). Users can then execute applications with SSH or Globus GRAM (Foster c, 2002) either by interactively using remote display protocols such as VNC (Richardson, 1998), or batch modes (Figueiredo a, 2003).

3-3-2 Virtual OS based Distributed Systems

The hosting of application services by virtual OSES as a distributed-system utility has been suggested (Jiang c, 2003); specifically, Service-On-Demand Architecture (SODA) hosted upon service Hosting Utility Platforms (HUP), providing on-demand creation of application services in distributed systems' environments. These application services, including guest operating systems (virtual OSES), are dynamically created and automatically bootstrapped as a group of virtual service nodes, and each virtual service node is represented as a virtual machine, providing administration isolation in addition to fault and attack isolation (Jiang, c, 2003). The components of Service-On-Demand Architecture (SODA) are middleware entities SODA Agent, SODA Master, SODA Daemon, and Service Switch; initially, the Application Service Provider (ASP) requests service creation to the SODA Agent with a resource requirement, after which the SODA Master checks whether the resource requirement of ASP can be satisfied with the HUP resource availability (Jiang c, 2003). If the resource requirement can be acceptable on HUP, SODA Master consults the SODA Daemon, with the latter downloading

application service images and bootstrapping virtual service nodes; after the service bootstraps, Service Switch will accept client requests and redirect to the appropriate virtual service node (Jiang c, 2003). A similar approach is that available with the Denali (Whitaker a, 2002) and Xenoserver (Hand, 2003) projects, both providing isolation between internet services on shared hardware in distributed/grid environments. These application services can complement web service based grid platform (Foster d, 2002) due to its resemblance to service-oriented architecture (Jiang c, 2003)

3.4 TestBed Studies

According to Jiang (Jiang c, 2003), VM overlay networks can be deployed to test and monitor underlying physical networks and applications running the VMs. Virtual-machine monitor-based overlay supports distributed virtualization (with each node able to provide simultaneously-running multiple services in a multiplex manner) allowing each application to be run as a part of the overlay, and not globally scheduled to run (Peterson, 2002). In Virtual Internetworking on Overlay Infrastructure (VIOLIN) suggested by (Jiang c, 2003), network components such as routers, switches, and end-hosts can be virtualized on top of overlay infrastructure to be user-configurable on demand, easily arranged for different testbeds associated with VM based distributed systems. The components of this architecture consist of virtual end-hosts (i.e. virtual machines in physical hosts) and virtual routers (i.e. virtual machines with multiple virtual interfaces, having the capability of forwarding between each virtual LAN) (Jiang c, 2003). Virtual LANs can be organized by individual virtual switches which connect multiple virtual end-hosts, and are responsible for packet forwarding (at the data link layer); this

architecture creates a VM network for the various services of distributed systems with no modifications of the real internet infrastructure, making the testing of a VM network for different services of distributed systems easier (Jiang c, 2003). The figure below illustrates relations between overlay infrastructure of virtual components, and underlying internet:

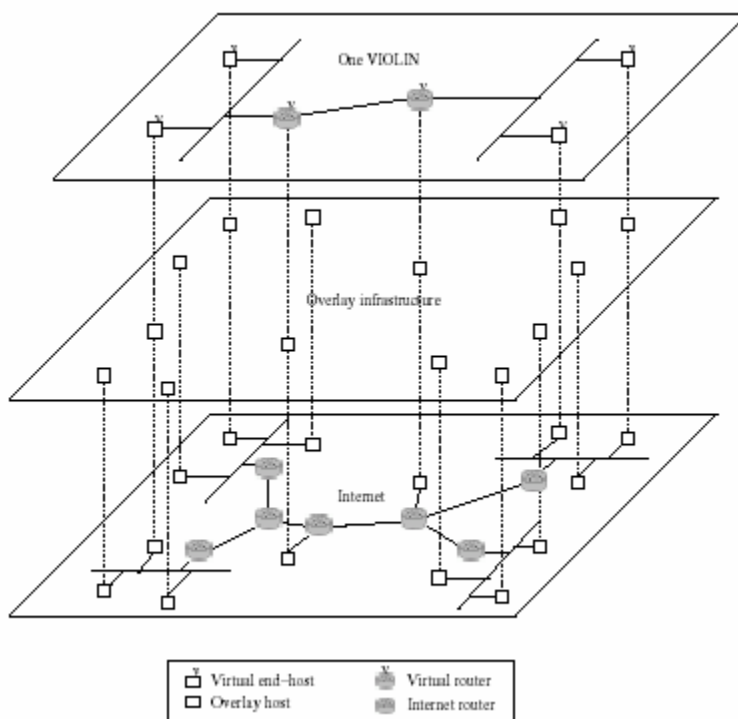


Figure 5 (Jiang c, 2003, page: 2), Virtual Internetworking on Overlay Infrastructure

Sundaraj (Sundaraj, 2003) first suggested Physical, VMD (Virtual Machine Daemon), and VM layers for the monitoring and testing of VM networks; the first is an underlying IP network, while VMD layers are an overlay in this architecture and able to manage VMs, monitoring both the resources provided by underlying physical networks and the

resources requested by VMs in VM layers. According to Jiang (Jiang d, 2003), the major steps for testing VM networks are as follows: first, specified VM and VM networks are required by using a well-defined script language; second, the logical entities in the testbed should be mapped onto virtual machines in VM networks; third, VM networks should perform virtual-node and virtual-topology creation; finally, the test of distributed systems or grid services (either batch-oriented or interactive) can be conducted with experimenters monitoring and managing VM networks at run time. By monitoring the VMD layer, Sundaraj (Sundaraj, 2003) demonstrated that it is possible to adapt communication and computation behavior of VMs in the VM layer, allowing VMD routing rules and topology to be changed for the purpose of efficiently migrating VMs and/or deploying various services in distributed systems on grid.

4 Concluding Remarks

Flexible virtual machines, such as on-demand created, migrating, and dynamically instantiated, can be deployed in distributed systems and grids to provide computational environment migration, customized resource management, isolation, and better security models. The main challenge of virtual machines in distributed systems and grids has been to overcome performance degradation due to virtualization overhead (Chen, 2001). This survey examined, for example, the ‘paravirtualization’ approach, in which both virtualization of a subset of the processor’s instruction set, and specialized virtual devices, enhance performance; this technology allows thousands of lightweight VMs to be run on

a VMM in a single machine at the same time (Barham, 2003, Whitaker a, 2002). The ideas associated with the use of optimized VM migration technologies, especially those used in entire computation-environment migration, have been discussed (Sapuntzakis, 2002, Kozuch, 2002) as well. This survey further examined virtual network-based testbeds to test and monitor a network of virtual machines. Virtual machine technology recently began providing a new abstraction layer in both distributed systems and grids, evolving to envision new computing paradigms.

Appendix-I

Bibliography

- [1]. (Allen, 2001) G. Allen, T. Damlitsch, I. Foster, N. Karonis, M. Ripeanu, E. Seidel, B. Toonen, "Supporting Efficient Execution in Heterogeneous Distributed Computing Environments with Cactus and Globus", *Proc. Of 2001 ACM/IEEE Conference on Supercomputing*, pp:52-52, 2001.
- [2]. (Anderson, 2001) D. Anderson, H. Balakrishnan, F. Kaashoek, R. Morris, "Resilient overlay network", *Proc. Of 18th ACM Symp on Operating Systems Principles (SOSP)*, pp:131-145, 2001
- [3]. (Awadallah, 2002) A. Awadallah and M. Rosenblum, "The vMatrix: A network of virtual machine monitors for dynamic content distribution", *Proc. Of 7th International Workshop on Web Content Caching and Distribution*, webpage : http://klamath.stanford.edu/~aaa/vmatrix/vmatrix_wcw2002.pdf, 2002
- [4]. (Baratloo, 1996) A. Baratloo, M. Karaul, Z. Kedem, P. Wyckoff., "Charlotte: Metacomputing on the Web", *Journal of Future Generation Computer Systems*, vol.15, Issue:5-6, pp: 559-570, 1999
- [5]. (Barham, 2003) P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, A. Warfield, "Xen and the Art of Virtualization", *Proc. Of the 19th ACM Symposium on Operating Systems Principles*, pp:164-177, 2003.
- [6]. (Bavier, 2004) Andy Bavier, L. Peterson, M. Wawrzoniak, S. Karlin, T. Spalink, T. Roscoe, D. Culler, B. Chun, M. Bowman, "Operating System Support for Planetary-Scale Network Services", *Proc. Of the 1st USENIX/ACM Symposium on Networked Systems Design and Implementation*, pp:253-266, 2004.
- [7]. (Bell, 2002) G. Bell, J. Gray, "What's next in high-performance computing?", *Communications of the ACM*, vol. 45, Issue: 2, pp:91-95, 2002.

- [8]. (Bester, 1999) J. Bester, I. Foster, C. Kesselman, J. Tedesco, S. Tuecke ,”GASS: A Data Movement and Access Service for Wide Area Computing Systems”, *Proc. Of the Sixth Workshop on Input/Output in Parallel and Distributed Systems*, pp:78-88, 1999.
- [9]. (Bowles, 1978) Bowles, L.Kenneth, “UCSD: Pascal”, *Byte* 46, pp:170-173 1978
- [10]. (Boyd, 2002) T. Boyd, P. Dasgupta, “Process Migration: A Generalized Approach Using a Virtualizing Operating System”, *ICDCS’ 2002*, pp:385-385, 2002.
- [11]. (Bressoud, 1996) T. Bressoud , F. Schneider, “Hypervisor-based Fault Tolerance”, *ACM Transactions on Computer Systems*, vol.14, Issue:1, pp:1-11, 1996.
- [12]. (Buchacker, 2002) K. Buchacker, V. Sieh, H. Hoxer, “Implementing a User Mode Linux with Minimal Changes from Original Kernel”, *9th International Linux System Technology Conference*, pp: 72–82, 2002
- [13]. (Bugnion, 1997) E. Bugnion, S. Devine, M. Rosenblum, “Disco: running commodity operating systems on scalable multiprocessors”, *Proc. Of the Sixteenth ACM Symposium on Operating Systems Principles*, pp:143-156 , 1997.
- [14]. (Calder, 2003) B. Calder, A. Chien, J. Wang, D. Yang “The Entropia Virtual Machine for Desktop Grids”, *Department of Computer Science and Engineering, Technical Report*, webpage: http://www.cs.ucsd.edu/Dienst/Repository/2.0/Body/ncstrl.ucsd_cse/CS2003-0773/postscript, 2003
- [15]. (Campbell a, 1999) A. Campbell, H. D. Meet, M. Kounavis, K. Miki, J. Vicente, D. Villela, "A Survey of Programmable Networks“, *ACM Computer Communications Review*, vol. 29, issue: 2, pp: 7-24, 1999.
- [16]. (Campbell b, 1999) A. Campbell, J. Vicente, D. Villela, “Virtuosity: Performing Virtual Network Resource Management”, *7th IEEE/IFIP International Workshop on Quality of Service (IWQOS'99)*, pp: 65-76,1999.
- [17]. (Campbell c, 1999) A. T. Campbell, M. E. Kounavis, D. A. Villela, J. Vicente K. Miki, H. G. De Meer, K. S. Kalaichelvan , "Spawning

Networks", *IEEE Network Magazine* vol. 13, Issue: 4, pp: 16-30, 1999.

- [18]. (Casanova, 2002) H. Casanova, "Distributed Computing Research Issues in Grid Computing", *ACM SIGAct News*, vol. 33, Issue:3, pp: 50-70, 2002.
- [19]. (Casas, 1995) J. Casas, D. L. Clark, R. Conuru, S. W. Otto, R. M. Prouty, J. Walpole, "MPVM: A Migration Transparent Version of PVM", *Computing Systems*, vol. 8, Issue: 2, pp:171-216, 1995.
- [20]. (Chen, 2001) P.M. Chen, B.D. Noble, "When virtual is better than real", *Proc. Of 8th Workshop on Hot Topics in Operating Systems*, pp: 133-138, 2001.
- [21]. (Chien a, 2003) A. Chien, B. Calder, S. Elbert, and K. Bhatia, "Entropy: Architecture and Performance of an Enterprise Desktop Grid System", *Journal of Parallel and Distributed Computing*, vol. 63, pp: 597-610, 2003.
- [22]. (Chien b, 1997) A. Chien, S. Pakin, M. Lauria, M. Buchanan, K. Hane, L. Giannini, and J. Prusakova, "High Performance Virtual Machines (HPVM): Clusters with Supercomputing APIs and Performance", *Proc. Of the Eighth SIAM Conference on Parallel Processing for Scientific Computing*, webpage: <http://www-csag.ucsd.edu/papers/csag/external/hpvm-siam97.ps>, 1997.
- [23]. (Creasy, 1981) R.J. Creasy, "The Origin of the VM/370 Time-Sharing System," *IBM Journal of Research and Development*, vol. 25, Issue: 5, pp: 483-490, 1981.
- [24]. (Czajkowski, 1998) K. Czajkowski, I. Foster, N. Karonis, C. Kesselman, S. Martin, W. Smith, S. Teucke, "A resource management architecture for metacomputing systems", *Proc. Of the Fourth Workshop on Job Scheduling Strategies for Parallel Processing*, pp: 62-82, 1998.
- [25]. (Czajkowski, 2001) G. Czajkowski, L. Daynes, "Multitasking without Compromise: A Virtual Machine Evolution", *Proc. Of the OOPSLA '01 conference on Object Oriented Programming Systems Languages and Applications*, pp: 125-138, 2001.
- [26]. (Dike, 2001) J. Dike, "User-mode Linux", *Proc. Of the 5th Annual*

Linux Showcase and Conference, pp:3-14, 2001.

- [27]. (Dincer, 1996) K. Dincer, G.C. Fox. “Building a World-Wide Virtual Machine Based on Web and HPCC Technologies”, *Proc. Of ACM/IEEE conference on Supercomputing, article no. 42*, webpage: <http://www.supercomp.org/sc96/proceedings/SC96PROC/DINCER/INDEX.HTM>, 1996.
- [28]. (Dinda, 1999) P. A. Dinda, D. R. O’Hallaron, “An extensible toolkit for resource prediction in distributed systems”, *Technical Report CMU-CS-99-138, School of Computer Science, Carnegie Mellon University*, webpage: <http://reports-archive.adm.cs.cmu.edu/anon/1999/CMU-CS-99-138.pdf> 1999.
- [29]. (Duffield, 1999) N. G. Duffield, P. Goyal, A. G. Greenberg, P. P. Mishra, K. K. Ramakrishnan, J. E. van der Merive, “A flexible model for resource management in virtual private networks”, *Proc. Of the conference on Applications, technologies, architectures, and protocols for computer communication, pp: 95–108*, 1999.
- [30]. (Ensim, 2003) Ensim, “Ensim Virtual Private Servers”, webpage : http://www.ensim.com/products/materials/datasheet_vps_051003.pdf , 2003.
- [31]. (Figueiredo a, 2003) R. Figueiredo, P. Dinda, and J. Fortes, “A Case for Grid Computing on Virtual Machines”, *Proc. Of IEEE ICDCS 2003, pp: 550-559*, 2003
- [32]. (Figueiredo b, 2001) R. Figueiredo, N. Kapadia, J. Fortes, “The PUNCH Virtual File System: Seamless Access to Decentralized Storage Services in a Computational Grid”, *Proc. Of IEEE International Symposium on High Performance Distributed Computing (HPDC), pp:334-346*, 2001.
- [33]. (Figueiredo c, 2003) R.Figueiredo, “VP/GFS: an Architecture for Virtual Private Grid File Systems”, *Technical Report, ACIS, University of Florida*, webpage : <http://byron.acis.ufl.edu/papers/tr-acis-03-001.pdf>, 2003.
- [34]. (Folliot, 2002) B. Folliot, I. Piumarta, L. Seinturier, C. Baillarguet, C. Khoury, A. Leger and F. Ogel, “Beyond Flexibility and Reflection: The Virtual Virtual Machine Approach”, *IWCC’ 2001, pp:16-25*, 2002

- [35]. (Ford, 1996) B. Ford et al, "Microkernels meet recursive virtual machines", *Proc. Of the Second Symposium on Operating Systems Design and Implementation*, pp:137-151, 1996.
- [36]. (Foster a, 2001) I. Foster, C.Kesselman, S.Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," *International J. Supercomputer Applications*, vol.15, Issue:3, pp:200-222, 2001.
- [37]. (Foster b, 1996) I. Foster, C. Kesselman, "Globus: A Metacomputing Infrastructure Toolkit", *International Journal of Supercomputer Applications and High Performance Computing*, vol.11, Issue:2, pp:115-128, 1996
- [38]. (Foster c, 2002) B. Allcock, J. Bester, J. Bresnahan, A. L. Chervenak, I. Foster, C. Kesselman, S. Meder, V. Nefedova, D. Quesnel, and S. Tuecke, "Data management and transfer in high-performance computational grid environments", *Parallel Computing Journal*, vol. 28, Issue:5, pp: 749-771, 2002.
- [39]. (Foster d, 2002) C. Kesselman, J. Nick, S. Tuecke, "The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration", *Open Grid Service Infrastructure WG, Global Grid Forum*, webpage : <http://www.globus.org/research/papers/ogsa.pdf>, 2002
- [40] (Foster e, 1999) I. Foster, C. Kesselman, C. Lee, R. Lindell, K. Nahrstedt, , A. Roy, "A Distributed Resource Management Architecture that Supports Advance Reservations and Co-Allocation", *Proc. Of IEEE IWQoS'99*, pp:27-36, 1999.
- [41]. (Foster f, 2003) I. Foster, "The Grid: Computing without Bounds," *Scientific Am.*, vol. 288, Issue: 3, pp:78-85, 2003
- [42]. (Gannon, 2002) D. Gannon, R. Bramley, G. Fox, S. Smallen, A. Rossi, R. Ananthakrishnan, F. Bertrand, K. Chiu, M. Farrellee, M. Govindaraju, S. Krishnan, L. Ramakrishnan, Y. Simmhan, A. Slominski, Y. Ma, C. Olariu, N. Rey-Cenvaz, "Programming the Grid: Distributed Software Components, P2P and Grid Web Services for Scientific Applications", *Journal of Cluster Computing*, vol.5, Issue: 3, pp:325-336, 2002.
- [43]. (Ganget, 2002) G. Ganget, D. Engler, M. F. Kaashoek, H. Briceno, R. Hunt, T. Pinckney, "Fast and Flexible Application-Level Networking on Exokernel Systems", *ACM Transactions on Computer Systems*, vol. 20, Issue:1, pp:49-83, 2002.

- [44]. (Garfinkel b, 2003) T. Garfinkel, M. Rosenblum, “A Virtual Machine Introspection-based Architecture for Intrusion Detection”, *10th Annual Network and Distributed System Security Symposium, NDSS’ 2003*, webpage : <http://www.isoc.org/isoc/conferences/ndss/03/proceedings/papers/13.pdf> , 2003
- [45]. (Geist, 1999) A. Geist, J. Dongarra, G. Fagg, A. Gray, “HARNESSE: a next generation distributed virtual machine”, *Journal of Future Generation Computer Systems*, pp: 571-582, 1999.
- [46]. (Goldberg a, 1974) R.P. Goldberg, “Survey of virtual machine research” *IEEE Computer Magazine*, vol.7, Issue: 6, pp:34-45, 1974
- [46]. (Goldberg b, 1974) R.P. Goldberg, G.J. Popek, “Formal Requirements for Virtualizable Third Generation Architectures” *In Communications of the ACM*, vol. 17, Issue: 7, pp: 412-421, 1974
- [48]. (Gough, 2001) J. Gough, “Stacking them up: A Comparison of Virtual Machines”, *Proc.Of the 6th Australasian conference on Computer systems architecture*, pp: 55- 61, 2001.
- [49]. (Grimshaw a, 1997) A.S. Grimshaw, W.A. Wulf, and the Legion team, “The Legion vision of a worldwide virtual computer”, *Magazine of Communications of the ACM*, vol. 40, Issue:1, pp:39-45, 1997
- [50]. (Grimshaw b, 1999) A.S. Grimshaw, A Ferrari, F.C Knabe, M.Humphrey , “Wide-Area Computing: Resource Sharing on a Large Scale”, *IEEE Computer*, vol. 32, Issue : 5 pp: 29-37, 1999.
- [51]. (Hand a, 2003) S. Hand, T. Harris, E. Kotsovinos, I. Pratt, “Controlling the Xenoserver Open Platform”, *Proc. Of OPENARCH '03*, pp: 3-11, 2003.
- [52]. (Hand b, 2003) K.A. Fraser, S. M.Hand, T. L.Harris, I. M. Leslie, I. A. Pratt, , “The Xenoserver computing infrastructure”, *Technical Report, University of Cambridge, Computer Laboratory*, webpage: <http://www.cl.cam.ac.uk/TechReports/UCAM-CL-TR-552.pdf> , 2003.
- [53]. (Harris, 2001) T. L. Harris, “Extensible Virtual Machines”, *PhD thesis, Churchill College, University of Cambridge*, webpage:

<http://www.cl.cam.ac.uk/~tlh20/papers/tim-harris-thesis-tr.ps.gz>, 2001.

- [54]. (Jiang a, 2003) X. Jiang, D. Xu, "SODA: a Service-On-Demand Architecture for Application Service Hosting Utility Platforms", *IEEE International Symposium on High Performance Distributed Computing (HPDC-12)*, pp:174-183, 2003.
- [55]. (Jiang b, 2004) X. Jiang, D. Xu, R. Eigenmann, "Protection Mechanisms for Application Service Hosting Platforms", *to appear in Proc. Of IEEE/ACM Int'l Symposium on Cluster Computing and the Grid (CCGrid 2004)*, Chicago, IL, webpage: <http://www.cs.purdue.edu/homes/dxu/pubs/CCGrid04.pdf>, 2004.
- [56]. (Jiang c, 2003) X.Jiang, D. Xu, "VIOLIN: Virtual Internetworking on OverLay Infrastructure", *Department of Computer Sciences Technical Report CSD TR 03-027*, Purdue University, webpage: <http://www.cs.purdue.edu/homes/dxu/pubs/violin.pdf>, 2003.
- [57]. (Jiang d, 2003) X. Jiang and D. Xu, "vBET: a VM-Based Emulation Testbed", *Proc. Of ACM SIGCOMM 2003 Workshops*, pp: 95-104, 2003.
- [58]. (Kamp, 2000) P.H. Kamp, R. N. M. Watson, "Jails: Confining the Omnipotent Root", *Proc. Of the 2nd International SANE Conference*, webpage: <http://phk.freebsd.dk/pubs/sane2000-jail.pdf>, 2000.
- [59]. (King a, 2002) S.T. King , G.W. Dunlap , S. Cinar, M. Basrai, P.M. Chen, "ReVirt: Enabling Intrusion Analysis through Virtual Machine Logging and Replay", *Proc. Of Symposium on Operating Systems Design and Implementation*, pp:211-224, 2002.
- [60]. (King b, 2002) S.T. King, G.W. Dunlap, P.M. Chen, "Operating System Support for Virtual Machines", *Proc. Of USENIX 2003 Annual Technical Conference*, pp: 71-84, 2003
- [61]. (King c, 2002) S.T. King, P.M. Chen, "Operating System Extensions to Support Host-Based Virtual Machines", *Technical Report University of Michigan*, webpage:

<http://www.eecs.umich.edu/techreports/cse/2002/CSE-TR-465-02.pdf>, 2002.

- [62]. (Kounavis, 1999) M. E. Kounavis A. T. Campbell, H G. De Meer, , K. Miki, J. Vicente, and D. A. Villela, "The Genesis Kernel: A Virtual Network Operating System for Spawning Network Architectures", *Proc. Of 2nd IEEE International Conference on Open Architectures and Network Programming (OPENARCH'99)*, pp:115-127, 1999
- [63]. (Kounavis, 2001) M. E. Kounavis, A. T. Campbell, S. Chou, F. Modoux, J. Vicente, and H. Zhang, "The Genesis Kernel: A Programming System for Spawning Network Architectures", *IEEE Journal on Selected Areas in Communications (JSAC), Special Issue on Active and Programmable Networks, Vol. 19, Issue:3*, pp : 49-73, 2001.
- [64]. (Kozuch, 2002) M. Kozuch, M. Satyanarayanan, "Internet Suspend/Resume", *Fourth IEEE Workshop on Mobile Computing Systems and Applications*, pp :40-46, 2002.
- [65]. (Kreiner, 2002) C. Kreiner, C. Steger, E. Teiniker, R. Weiss "A Novel Co-design Approach based on Distributed Virtual Machines", *Proc. Of the 10th international symposium on Hardware/software co-design*, pp:109- 114, 2002.
- [66]. (Lindholm, 97) T. Lindholm, F. Yellin, "The Java Virtual Machine Specification", *The Java Series, Addison-Wesley, Reading,MA, USA, January 1997*.
- [67]. (Litzkow, 1992) M.J. Litzkow, M.Livny, M. W. Mutka, "Condor technical report", *Technical Report CS-TR-92-1069, University of Wisconsin*, webpage: <http://citeseer.ist.psu.edu/briker91condor.html> 1992.
- [68]. (Merwe a, 1997) J.E. van der Merwe, J.E., Rooney, S. Leslie, I.M. S.A. Crosby, "The Tempest - A Practical Framework for Network Programmability", *IEEE Network magazines, vol.12, Issue:3*, pp:20-28, 1997.
- [69]. (Merwe b, 1997) J. E. van der Merwe I. M. Leslie, "Switchlets and dynamic virtual ATM networks", *Proc. Of the Fifth IFIP/IEEE International Symposium on Integrated Network Management*, pp: 355-368, 1997.

- [70]. (Migliardi a, 1999) M. Migliardi, V. Sunderam, "Heterogeneous Distributed Virtual Machines in the Harness Metacomputing Framework, *Proc. Of Heterogeneous Computing Workshop of IPPS/SPDP99*, pp: 60-72, 1999.
- [71]. (Migliardi b, 1998) M. Migliardi, V. Sunderam, A. Geist, J. Dongarra, "Dynamic Reconfiguration and Virtual Machine Management in the Harness Metacomputing System", *Proc. Of ISCOPE98, Santa Fe', New Mexico (USA)*, pp: 127-134, 1998.
- [72]. (Migliardi c, 1999) Mauro Migliardi and Vaidy Sunderam, "PVM Emulation in the Harness MetaComputing System: A Plug-in Based Approach", *Lecture Notes in Computer Science (1697)*, pp : 117-124, 1999.
- [73]. (Milojicic, 2000) D.S. Milojicic, F. Dougliis, Y. Paindavein, R. Wheeler, S. Zhou, "Process Migration", *ACM Computing Surveys*, vol. 32, Issue:3, pp: 241-299, 2000
- [74]. (Moore, 2002) J. Moore, J. Chase, "Cluster on demand", *Technical report, Duke University*, webpage: <http://www.cs.duke.edu/justin/cod/CS-2002-07.pdf>, 2002.
- [75]. (Osman, 2002) S. Osman, D. Subhraveti, G. Su, J. Nieh, "The design and implementation of Zap: A system for migrating computing environments", *Proc. Of 5th USENIX Symposium on Operating Systems Design and Implementation*, pp: 361-376, 2002.
- [76]. (Peterson a, 2002) L.Peterson, T. Roscoe, "PlanetLab Phase 1: Transition to an Isolation Kernel", *PlanetLab Design Notes*, webpage: <http://www.planet-lab.org/pdn/pdn02-003.pdf>, 2003
- [77]. (Peterson b, 2002) L. Peterson, T. Anderson, D. Culler, and T. Roscoe, "A Blueprint for Introducing Disruptive Technology into the Internet", *Proc. Of ACM HotNets-I*, pp: 59-64, 2002.
- [78]. (Rajkumar, 1998) R. Rajkumar, K. Juvva, A. Molano, S. Oikawa, "Resource Kernels: A Resource-Centric Approach to Real-Time Systems", *Proc. Of the SPIE/ACM Conference on Multimedia Computing and Networking*, pp:476-490, 1998.
- [79]. (Reed, 1999) D. Reed, I. Pratt, P. Menage, S. Early, N. Stratford, "Xenoservers: accounted execution of untrusted code", *Proc. Of the fifth Workshop on Hot Topics in Operating*

Systems, pp:136-141, 1999

- [80]. (Richardson, 1998) T. Richardson, Q. Stafford-Fraser, K. R. Wood, A. Hopper “Virtual Network Computing”, *Magazines of IEEE Internet Computing, vol. 2, Issue:1, pp:33–38, 1998*
- [81]. (Riehle , 2001) D. Riehle, S. Fraleigh, D. Bucka-Lassen, N. Omorogbe, “The architecture of a UML virtual machine”, *Proc. Of the 2001 Conference on Object-Oriented Programming Systems, Languages, and Applications (OOPSLA '01), pp: 327-341, 2001.*
- [82]. (Robin, 2000) J.S. Robin, C.E. Irvine, “Analysis of the intel pentium's ability to support a secure virtual machine monitor”, *Proc. Of the 9th USENIX Security Symposium, webpage: http://www.usenix.org/publications/library/proceedings/sec2000/full_papers/robin/robin.pdf, 2000.*
- [83]. (Rodeh, 1998) O. Rodeh, K. Birman, M. Hayden, D. Dolev, “Dynamic Virtual Private Networks,” *Technical Report, TR98-1695, Dept. of Computer Science, Cornell University, webpage: <http://citeseer.ist.psu.edu/rodeh98dynamic.html>, 1998.*
- [84]. (Sandberg, 1985) R.Sandberg, D.Goldberg, S. Kleiman, D. Walsh, “Design and implementation of the Sun network file system”, *Proc. Of Summer Usenix Conference, pp: 119-130, 1985*
- [85]. (Sapuntzakis, 2002) C. Sapuntzakis, R. Chandra, B. Pfaff, J. Chow, M. Lam, M. Rosenblum, “Optimizing the Migration of Virtual Computers”, *ACM SIGOPS Operating Systems Review, vol. 36, Issue: SI, pp:377-390, 2002.*
- [86]. (Schmidt, 2000) B. K. Schmidt, “Supporting Ubiquitous Computing with Stateless Consoles and Computation Caches”, *PhD thesis, Department of CS, Stanford University, webpage : <http://suif.stanford.edu/papers/schmidt00.ps.gz>, 2000.*
- [87]. (Shriver, 1976) B. D. Shriver , J. W. Anderson , L. J. Waguespack , D. M. Hyams , R. A. Bombet, “An implementation scheme for a virtual machine monitor to be realized on user microprogrammable minicomputers”, *Proc. Of the annual conference, pp: 226-232, 1976*
- [88]. (Sirer , 1999) E.G. Sirer, R. Grimm, A.J Gregory, B.N. Bershad, “Design and implementation of a distributed virtual machine for networked computers”, *ACM Symposium on Operating*

System Principles (SOSP'99), vol. 33, Issue: 5, pp: 202-216, 1999.

- [89]. (Sirer a, 1998) E.G. Sirer, R. Grimm, A.J. Gregory, B.N. Bershad, S. McDirmid, "Distributed Virtual Machines: A System Architecture for Network Computing", *Proc. of the Eighth ACM SIGOPS European Workshop*, pp:13-16, 1998.
- [90]. (Smith, 2001) J.E. Smith, "An overview of virtual machine architectures", *Technical Report, University of Wisconsin*, webpage: http://swig.stanford.edu/~fox/cs241/readings/smith_vm_overview.pdf, 2001.
- [91]. (Sugerman, 2001) J. Sugerman, G. Venkitachalam, B.H. Lim, "Virtualizing I/O Devices on VMware Workstation's Hosted Virtual Machine Monitor", *USENIX Annual Technical Conference*, pp: 1-14, *USENIX Association*, 2001.
- [92]. (Sundaraj, 2003) A. Sundararaj, P. Dinda, "Towards Virtual Networks for Virtual Machine Grid Computing", *Technical Report NWU- CS-03-27, Department of Computer Science, Northwestern University*, webpages : <http://www.cs.northwestern.edu/~ais/nwu-cs-03-27.pdf>, 2003
- [93]. (Sunderam, 2002) V. Sunderam, D. Kurzyniec, "Lightweight self -organizing frameworks for metacomputing". *The 11th International Symposium on High Performance Distributed Computing (HPDC-11)*, pp:113-122, 2002.
- [94]. (Tanenbaum, 95) A.S. Tanenbaum, "Distributed Operating Systems", *Prentice Hall, Inc., ISBN 0-13-143934-0*, 1995.
- [95]. (Thain, 2001) D. Thain, J. Basney, S.C. Son, M. Livny, "The kangaroo approach to data movement on the grid", *Proc. Of the 2001 IEEE International Conference on High-Performance Distributed Computing (HPDC)*, pp: 325–333, 2001
- [96]. (Thomas, 99) R. Thomas, "Mite: a basis for ubiquitous virtual machines", *PhD dissertation, University of Cambridge Computer Laboratory*, webpage: <http://rrt.sc3d.org/download/research/mitethes.pdf> 1999.

- [97]. (UFL LAB, 2003) In-VIGO Project, “Virtualization middleware for computational grid”, *webpage* :
http://invigo.acis.ufl.edu/docs/aboutInVigo.html
- [98]. (Waldspurger, 2002) C. Waldspurger, “Memory Resource Management in VMware ESX Server”, *Proc. Of the 5th symposium on Operating systems design and implementation*, pp:181-194, 2002
- [99]. (Weissman, 2002) J. Weissman, B. Lee, “The Virtual Service Grid: an architecture for delivering high-end network services”, *Concurrency: Practice and Experience*, vol. 14, issue: 4, pp: 287-319, 2002.
- [100]. (Whitaker a, 2002) A. Whitaker, M. Shaw, and S. Gribble, “Denali: Lightweight virtual machines for distributed and networked applications”, *Proc. Of the USENIX Technical Conference*, *webpage: http://www.cs.ucsb.edu/~rich/class/cs595-os/denali_usenix2002.pdf*, 2002
- [101]. (Whitaker b, 2002) A. Whitaker, M. Shaw, S.D. Gribble, “Scale and performance in the Denali isolation kernel”, *ACM SIGOPS Operating Systems Review*, vol. 36, Issue: SI, pp: 195- 209, 2002.
- [102]. (Wolski, 1999) R. Wolski, J. Brevik, C. Krintz, G. Obertelli, N. Spring, A. Su, “Running *EveryWare* on the computational grid”, *Proc. of the 1999 ACM/IEEE Supercomputing Conference*, article no. 6, 1999.
- [103]. (Yang, 2003) K. Yang , X. Guo , A. Galis , B. Yang , D. Liu, “Towards efficient resource on-demand in Grid Computing”, *ACM SIGOPS Operating Systems Review*, vol.37 Issue:2, pp:37-43, 2003

Appendix-II

Annotated Bibliography

(Used a double asterisk for milestone papers)

- [1]. (Awadallah, 2002) A. Awadallah and M. Rosenblum, “The vMatrix: A network of virtual machine monitors for dynamic content distribution”, *Proc. Of 7th International Workshop on Web Content Caching and Distribution*, webpage : http://klamath.stanford.edu/~aaa/vmatrix/vmatrix_wcw2002.pdf, 2002

Keywords: virtual machine, virtual machine monitor, dynamic content distribution, dependencies, VM state encapsulation, replication, transparent mobility, on-demand replication, secure connectivity, availability

The authors demonstrated that a VMM can encapsulate the state of the machine as a VM file which could be instantiated on any VMM running machine. According to the authors, the main problems associated with distribution and replication of dynamic content are dependencies such as libraries, third party modules, OS, server hardware, etc. The authors demonstrated that encapsulation of the state of the machine by VMM helps avoid these dependencies.

- [2]. (Barham, 2003) P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, A. Warfield, “Xen and the Art of Virtualization”, *Proc. Of the 19th ACM Symposium on Operating Systems Principles*, pp:164-177, 2003.

Keywords: virtual machine, virtual machine monitor(VMM), *paravirtualization*, hypervisors, performance isolation, concurrent virtual machines, virtual address translation, VMM scalability, hypercall

In this paper, the authors discussed the merits of using a Xen system. This system deploys *paravirtualization* which virtualizes a subset of the processor's instruction set and its specialized virtual devices to enhance performance. According to this paper, Xen replaces hardware interrupts with its own event system. Xen also allows guest OSes to read access to page tables while mediating write access for an update by its trapping mechanism. The author insisted a Xen system would improve its performance by using these *paravirtualization* mechanisms rather than full virtualization mechanisms.

- [3]. (Bavier, 2004) Andy Bavier, L. Peterson, M. Wawrzoniak, S. Karlin, T. Spalink, T. Roscoe, D. Culler, B. Chun, M. Bowman, "Operating System Support for Planetary-Scale Network Services, *Proc. Of the 1st USENIX/ACM Symposium on Networked Systems Design and Implementation*, pp:253-266, 2004.

Keywords : virtual machine, operating system, distributed virtual machine, resource isolation, resource coordination, resource partitioning, service creation, service termination, virtual servers, virtual services

In this paper, the authors suggested 'distributed virtualization', which enables each service to run a slice of global resources of *PlanetLab's* organization. The authors suggested this slice as a virtual machine form and a distributed set of VMs, can be treated and managed as a single entity. The authors suggested *PlanetLab OS*, which provides virtual machine abstraction at a system-call level and embeds global slice abstraction. The authors suggested slice creations with resource allocation and slice terminations

should be controlled by *node manager*, which is a global privileged service, and the remains should keep the local abstraction.

- [4]. (Boyd, 2002) T. Boyd, P. Dasgupta, “Process Migration: A Generalized Approach Using a Virtualizing Operating System”, *ICDCS 2002*, pp:385-385, 2002.

Keywords : Process migration, virtualized application, virtualized OS, physical binding, virtual binding, full state model, minimal state model, cooperation, resource sharing, distributed state method

In this paper, the authors suggested migration technology using virtualized application and operating systems by inserting additional software functionality. According to the authors, this scheme allows virtual bindings to be separated from physical bindings. The authors insisted that virtualized OS and virtualized software make it possible to migrate processes with minimal state, cooperate run applications and share distributed resources .

- [5]. (Buchacker, 2002) K. Buchacker, V. Sieh, H. Hoxer, “Implementing a User Mode Linux with Minimal Changes from Original Kernel”, 9th International Linux System Technology Conference, pp: 72–82, 2002

Keyword: user mode linux, linux kernel, user mode kernel, system-call, virtualized device, memory mapped I/O, interrupt handling, exception handling, signal handler, OS simulator

The authors suggested underlying machine virtualization at the system-call level. The authors showed that virtualization at the system-call can be achieved by modification of kernel code that interacts with hardware. The authors pointed out hardware interactive

assembler instructions in OS kernels such as interrupt, exception handling and access functions can be replaced with signal system-calls with minimum changes of underlying kernel. The authors insisted that this enables the multiple user mode kernels on each virtual device to host on a host kernel.

- [6]. (Bugnion, 1997) E. Bugnion, S. Devine, M. Rosenblum, “Disco: running commodity operating systems on scalable multiprocessors”, *Proc. Of the Sixteenth ACM Symposium on Operating Systems Principles*, pp:143-156, 1997.

Keywords: scalable multiprocessors, virtual machines, NUMA multiprocessor, virtual machine monitor, hardware virtualization, virtual machine overhead, virtual machine performance bottleneck, scalability, global buffer cache, page migration

In this paper, the authors showed ‘Disco’ virtual machine monitor on scalable multiprocessors. The authors suggested insertion of an additional software layer between hardware and operating systems to run multiple commodity operating systems on a scalable multiprocessor. The authors claimed that the main role of virtual machine monitor in this architecture is to schedule the virtual resources of the virtual machines on the physical resources of the scalable multiprocessors. The authors pointed out the main challenges of virtual machine monitor are to overcome virtualization overhead and additional memory cost for independent virtual machines.

- [7]. (Calder, 2003) B. Calder, A. Chien, J. Wang, D. Yang “The Entropia Virtual Machine for Desktop Grids”, *Department of Computer Science and Engineering, Technical Report*,

webpage: http://www.cs.ucsd.edu/Dienst/Repository/2.0/Body/ncstrl.ucsd_cse/CS2003-0773/postscript, 2003

Keywords : Virtual machine, Grid, Desktop Grids, Desktop distributed computing, desktop control, sandboxing, application security, VM wrapping, OS interception, virtualization

In this paper, the authors introduced Desktop Grids, which exploit the idle cycles on pervasive desktop PC. According to the author, applications are submitted and distributed by the binary level *Entropia Virtual Machine*. The author suggested components of this virtual machines as Desktop Control, Sandbox and application security. The author adapts binary rewriting to provide a light weight interface between application processes and operating system.

[8]. (Campbell a, 1999) A. Campbell, H. D. Meet, M. Kounavis, K. Miki, J. Vicente, D. Villela, "A Survey of Programmable Networks", *ACM Computer Communications Review*, pp: 7-24, 1999.

Keywords : programmable networks, virtual network, control software, communication hardware, network services, resource partitioning, resource isolation, resource coexistence, availability, service deployment

According to the authors, the goal of programmable networking is to simplify the deployment of new network services, and helps to create service and deploy them in networks with an explicit way. The authors insisted that programmable networks provide a foundation for virtual network architectures such as architecting, composing, deploying

virtual network. The authors insisted that separation of communications hardware from control software is the key to make networks programmable.

- [9]. (Campbell c, 1999) A. T. Campbell, M. E. Kounavis, D. A. Villela, J. Vicente K. Miki, H. G. De Meer, K. S. Kalaichelvan, "Spawning Networks", *IEEE Network Magazine* vol. 13, Issue: 4, pp: 16-30, 1999.

Keywords: spawning networks, virtual networks, programmable networks, network services, service deployment, child network, parent network, network customization, life cycle services, inheritance

The authors suggested Spawning Networks, which allows creating new architectures on-demand. The authors insisted that Spawning Networks allows creation, deployment, and management of network architectures automatically. According to the authors, a child network operates on a subset of its parent's network resources and is characterized as a programmable virtual networks. The authors insisted that Spawning Networks provides better network customization and simplify the deployment of new network service.

- [10]. (Chen, 2001) P.M. Chen, B.D. Noble, "When virtual is better than real", *Proc. Of 8th Workshop on Hot Topics in Operating Systems*, pp: 133-138, 2001.

Keywords: virtual machine, virtual machine monitor(VMM), OS kernel, OS security secure logging, intrusion prevention, intrusion detection, environment migration, VMWare, hypervisor

In this paper, the authors demonstrated the reasons why OS level VMs are useful for an intrusion prevention and detection systems. The author stated that running suspicious events on the real systems to test attacks might compromise the system. The author insisted that better approach is to clone the real system by OS-level VMs and test

suspicious events. The author also showed the advantage of OS-level VM-based environment migration. The author indicated that main challenge of this architecture is to overcome performance degradation due to virtualization overhead

- [11]. (Creasy, 1981) R.J. Creasy, "The Origin of the VM/370 Time-Sharing System," IBM Journal of Research and Development, vol. 25, Issue: 5, pp: 483-490, 1981

Keywords: virtual machines, time-sharing system, hardware virtualization, operating systems, multiple operating systems, control program, conventional monitor system VM/370, test systems, virtual memory

In this paper, the author showed the characteristics of VM/370 virtual machine. This paper presents a time-sharing system for multiple operating systems such as CMS, RSCS, or OSNS. The author demonstrated VM/370 virtual machine can virtualize IBM System/370 hardware and allows multiple operating systems to be run as a time-slice way. The authors insisted that test and production systems should coexist with these systems to help experiment for a new release.

- [12]. (Dike, 2001) J. Dike, "User-mode Linux", *Proc. Of the 5th Annual Linux Showcase and Conference*, pp:3-14, 2001.

Keyword: virtual machine, user mode linux, linux kernel, user mode kernel, *ptrace* system-calls, signal system-calls, virtual OS, virtual interrupt handling, signal handler, hardware virtualization

This paper shows design and implementation mechanisms for a Linux virtual machine running on a Linux host. This paper showed virtualization for hardware capability by means of Linux system-calls, which enables user mode kernel. This paper suggested

system-calls issued by user mode Linux processes should be intercepted by a special tracing process using *ptrace* linux system-call. This paper demonstrated the policy between user mode and kernel mode for a user mode linux. According to this paper, processes in user mode should have their system-calls intercepted and virtualized; however, in kernel mode, processes should be released from tracing mechanism and directly run in the host kernel. This paper insisted that this mechanism allows multiple user mode linux to be run on a host linux.

- [13]. *(Figueiredo a, 2003) R. Figueiredo, P. Dinda, and J. Fortes, “A Case for Grid Computing on Virtual Machines”, *Proc. Of IEEE ICDCS’ 2003*, pp: 550-559, 2003

Keywords : virtual machine, virtual machine monitor, classic virtual machine, computational grid, middleware, virtual networking, resource management, data management, mobile VMs, ISA-VMs

In this paper, the authors propose a fundamental change the abstraction level of Grid computing by the operating system users into the OS-level VMs. In this paper, the authors demonstrated software architecture for OS-level (classic) virtual machine grid computing by VM life cycle. The author suggested the components of a virtual grid session in this architecture involved in physical server, virtual machine OS image server, application image server, and user data server. The author insisted that the transfers between these servers can be carried out on-demand. The authors also insisted that VM-based grid deployment can be seamlessly supported by these migration of computing environment technologies (Sapuntzakis, 2002) (Osman, 2002).

- [14]. (Figueiredo b, 2001) R. Figueiredo, N. Kapadia, J. Fortes, "The PUNCH Virtual File System: Seamless Access to Decentralized Storage Services in a Computational Grid", *Proc. IEEE International Symposium on High Performance Distributed Computing (HPDC)*, pp:334-346, 2001.

Keywords: virtual file system, computational grid, NFS, file transfer, security, proxy-based file system, on-demand file transfer, shadow account, file account, self-organizing network

In this paper, the authors suggested PVFS, which is a virtual file system with logical user account. PVFS allows VM images and application data to be transferred on demand between storage (image servers and data servers in this survey's context) and computing servers. Traditional NFS file systems is established once for multiple users by system administrators and requires an explicit user intervention. Meanwhile, PVFS is created and terminated dynamically by client-server sessions and employs server-side proxies to delegate transactions between NFS clients and servers. These server-side proxies are managed on-demand by grid middleware and they differ from on-demand data access solutions for grid computing such as Condor (Litzkow, 1992) and Legion (Grimshaw a, 1997) by its ability to be applied on unmodified applications through NFS systems and legacy operating systems.

- [15]. (Figueiredo c, 2003) R.Figueiredo, "VP/GFS: an Architecture for Virtual Private Grid File Systems", *Technical Report, ACIS, University of Florida, webpage* : <http://byron.acis.ufl.edu/papers/tr-acis-03-001.pdf>, 2003.

Keywords : virtual machine, virtual file system, computational grid, proxy-based file system, on-demand file transfer, client-disk caching, SSH, Globus GSI, Grid File System, grid middleware

In this paper, the authors suggested ‘Virtual Private Grid File System (VP/GFS)’ ,which allows VP/GFS sessions to support on-demand data transfers on private, encrypting channel by SSH on NFS in Grid environment and integrates server-side proxy-based PVFS with disk caching by the client-side proxies. The main purpose of client-side proxies in VP/GFS is to cache file system data for an enhanced performance such as improved access latency.

- [16]. (Ford, 1996) B. Ford et al, “Microkernels meet recursive virtual machines”, *Proc. Of the Second Symposium on Operating Systems Design and Implementation*, pp:137-151, 1996.

Keywords: microkernel, virtual machine, virtual machine monitor, recursive virtual machine, modular OS, extensible OS, scalability, performance, isolation, distributed memory manager

In this paper, the author showed recursive virtual machines by a virtual machine and a microkernel concept. The author defined recursive virtual machines as a ‘virtual machines running on other virtual machines’. The author insisted that traditional monolithic kernel is difficult to achieve recursive virtual machine, because the large source base and modification required is prohibitive. The author suggested modular OS design such that functionality may be implemented at different levels within a nested VMs and parent can re-implement OS functionality on behalf of its child VM processes.

- [17]. (Foster a, 2001) I. Foster, C.Kesselman, S.Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations,"

Keywords; Grid, Virtual Organization, dynamic resource sharing, cross-organizational resource sharing, distributed computing, inter Grid protocol, Grid architecture, Globus, Grid applications

The authors identify that the problem lies in the Grid concept is ‘coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organization’.

The authors demonstrated ‘Grid Problem’ can be defined as flexible, secure, coordinated resource sharing among dynamic collections of individuals, institution, and resources.

The authors define these dynamic collections as ‘Virtual Organization (VO)’. The authors also suggested Grid protocol architecture to define resource sharing frameworks between VO users and resources.

- [18]. ** (Foster d, 2002) C. Kesselman, J. Nick, S. Tuecke, “The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration”, *Open Grid Service Infrastructure WG, Global Grid Forum, 2002. webpage :*
<http://www.globus.org/research/papers/ogsa.pdf>

Keywords : Grid, Grid service, Web Service, OGSA, Globus Toolkit , service virtualization, Grid service semantics, virtual organization, network protocol bindings, transient services

In this paper, the authors proposed the Open Grid Services Architecture which defines Grid service semantics such as creating, naming, and discovering transient Grid Service instances. The author defined a ‘service’ as a network-enabled entity that provides some capability. The authors showed service oriented Grid architecture coupled with Web

Services technologies (Grimshaw b, 1999). The authors insisted that the advantage of service-oriented model is that all components of the environment are virtualized. The author insisted that virtualization of services allows consistent resource access across multiple heterogeneous environments with location transparency and also multiple logical resource instances can be mapped onto the same physical resource.

- [19]. *(Goldberg a, 1974) R.P. Goldberg, "Survey of virtual machine research", *IEEE Computer Magazine*, vol.7, issue: 6 pp:34-45, 1974

Keyword : virtual machine, virtual machine monitor, virtual memory, abstract model, formal requirements, hypervisor, operating system, time sharing, hardware virtualization, sensitive instruction

The authors define a Virtual Machine (VM) as "a hardware-software duplicate of a real existing computer system in which statistically dominant subject of the virtual processor's instruction execute on host processor in native way". According to the authors, traditional Virtual Machines focused on VMM hardware virtualization aspect. The author insisted that virtual hardware exposed to VMM is functionally identical to the underlying machine, which enables the system to host unmodified multiple operating systems. The author claimed that Virtual Machine Monitor gives an illusion such that Guest OSes believe they are running directly on physical hardware by hardware virtualization layer provided by virtual machine monitor (VMM).

- [20]. (Hand b, 2003) K.A. Fraser, S. M.Hand, T. L.Harris, I. M. Leslie, I. A. Pratt, , "The Xenoserver computing infrastructure", *Technical Report, University of Cambridge, Computer Laboratory, webpage: <http://www.cl.cam.ac.uk/TechReports/UCAM-CL-TR->*

552.pdf, 2003.

Keywords : virtual machine, virtual machine monitor, distributed computing infrastructure, broker, service facilities, resource management, resource discovery, resource partitioning, resource accounting, virtualized debugging

In this paper, the authors suggested a public infrastructure for global distributed computing based on Xen virtual machine monitor. The components of this architecture are code execution platform providing resource portioning and accounting, brokers to match the client requirement and server availability and service facilities to provide efficient execution environment considering fault and latency as well. The author insisted that this infrastructure is a flexible system which support system-level, large scale system for competing users and tasks with different code written for a variety of languages and paradigms.

- [21]. (Jiang a, 2003) X. Jiang, D. Xu, "SODA: a Service-On-Demand Architecture for Application Service Hosting Utility Platforms", *IEEE International Symposium on High Performance Distributed Computing (HPDC-12)*, pp:174-183, 2003.

Keywords : virtual machine, computational grid, virtual OS, service-on-demand, virtual service, resource isolation, Open Grid Service Architecture(OGSA), User Mode Linux(UML), service switch, service priming

Authors suggested Service-On-Demand Architecture (SODA) on service Hosting Utility Platform(HUP), which provides on-demand creation of application services in Grid environment and multiple application services will be hosted in this platform.

These application services will be dynamically created and automatically bootstrapped including their guest OSes as a group of virtual service nodes and each virtual service

node is represented as a virtual machine (Virtual OS) to provide administration isolation in addition to fault and attack isolation.

- [22]. (Jiang b, 2004) X. Jiang, D. Xu, R. Eigenmann, "Protection Mechanisms for Application Service Hosting Platforms", *to appear in Proc. Of IEEE/ACM Int'l Symposium on Cluster Computing and the Grid (CCGrid 2004), Chicago, IL*, webpage: <http://www.cs.purdue.edu/homes/dxu/pubs/CCGrid04.pdf>, 2004

Keywords : virtual machine, virtual OS, application service hosting, intrusion detection, intrusion logging, resource isolation, virtual switching, virtual firewalling, virtual server, fault/attack isolation

In this paper, the author showed protection mechanisms for various isolated application services on virtual OS. The author showed two level application service hosting architecture such that application services are run on virtual OSES instead of host OS.

The author insisted that resource isolation, virtual switching and firewalling, and kernelized intrusion detection are key components to protect each application service.

- [23]. (Jiang c, 2003) X.Jiang, D. Xu, "VIOLIN: Virtual Internetworking on OverLay Infrastructure", *Department of Computer Sciences Technical Report CSD TR 03-027, Purdue University*, webpage: <http://www.cs.purdue.edu/homes/dxu/pubs/violin.pdf>, 2003

Keywords : virtual machine, overlay networks, virtual switch, virtual router, User mode linux(UML), virtual LAN(VLAN), virtual internetworking, isolation, software virtualization, application-level overlays

Authors suggested 'VIOLIN' (Virtual Internetworking on Overlay Infrastructure) and showed network component such as routers, switches and end-host can be virtualized on top of overlay infrastructure. The components of this architecture consist of virtual end-

hosts that are virtual machines in a physical overlay host and virtual routers that are also virtual machines with multiple virtual interfaces and have the capability of forwarding between each virtual LAN. Authors suggested Virtual LANs can be organized by each virtual switch which connects multiple virtual end-hosts for a virtual overlay network.

- [24]. (Jiang d, 2003) X. Jiang and D. Xu, “vBET: a VM-Based Emulation Testbed”, *Proc. Of ACM SIGCOMM 2003 Workshops*, pp: 95-104, 2003.

Keywords : virtual machine, virtual machine monitor, virtual OS, virtual network, emulation testbed, , overlay network, virtual distributed environment, resource isolation, linux modules, virtual topology

In this paper, the authors suggested emulation testbed based on virtual machine and virtual network technology. Virtual machine monitor is used in this testbed to virtualize underlying resources and virtual OS is used for a resource /administration isolation.

The components of this testbed such as a router, switch, firewall or application proxy are emulated by virtual machines or application software and then creates a virtual distributed environment. The author insisted that these components can be user-configurable on-demand and easily setup for tesbeds for Internet routing, distributed firewalls, peer-to-peer networks and distributed protocols.

- [25]. (King a, 2002) S.T. King , G.W. Dunlap , S. Cinar, M. Basrai, P.M. Chen, “ReVirt: Enabling Intrusion Analysis through Virtual Machine Logging and Replay”, *Proc. Of Symposium on Operating Systems Design and Implementation*, pp:211-224, 2002.

Keywords : virtual machine, virtual machine monitor, secure logging, intrusion detection, intrusion prevention, replay attacks, virtualization overhead, analyze attacks, user mode linux, cooperative logging

In this paper, the authors demonstrated secure logging systems by OS-level VMs. The authors showed shortcoming of current intrusion logging systems. The authors claimed that an attacker can easily turn off logging after he/she takes over the system, contents of logging after compromise cannot be trusted. The authors insisted that moving logging software out of the operating system and into the virtual machine monitor helps to replay the operating system's execution before, during, and after an attacker compromise the system.

[26]. (King b, 2002) S.T. King, G.W. Dunlap, P.M. Chen, "Operating System Support for Virtual Machines", *Proc. Of USENIX 2003 Annual Technical Conference*, pp: 71-84, 2003

Keywords : virtual machine, virtual OS, Type I VMM, Type II VMM, guest OSes, user mode linux, signal, exception, performance overhead, underlying hardware

In this paper, the authors classified virtual machine monitor as Type I VMM and Type II VMM. According to authors, Type I VMM runs directly on hardware with relative good performance and Type II VMM uses existing OS abstraction to host guest software including guest OSes. The author pointed out Type II VMM is convenient and easier to implement but the performance is relatively low in comparison to Type I VMM. The author suggested a better performance model for Type II VMM by exposing underlying hardware to guest software and fast interception of signals and exception.

- [27]. *(Kozuch, 2002) M. Kozuch, M. Satyanarayanan, "Internet Suspend/Resume", Fourth IEEE Workshop on Mobile Computing Systems and Applications, pp:40-46, 2002.

Keyword : environment migration, virtual machine, virtual machine monitor, job suspend , job resume, distributed file system, file state, mobile computing , mobile users , volatile state

In this paper, the authors suggested Internet Suspend/Resume with virtual machine monitor and distributed file system. The author insisted that distributed file system can keep only persistent state therefore it has a limitation to keep volatile state such as an execution state and it is not possible to suspend/resume their computational jobs when user moves another computer. The authors demonstrated the advantages of VM migration for an environment migration. The author pointed out the drawbacks of traditional user level process migration such as managing open file descriptor. The author insisted that virtual machine monitor can encapsulate all volatile execution state of a VM and allows mobile users to suspend their works from one computer and seamless resume their work at another computer.

- [28]. (Osman, 2002) S. Osman, D. Subhraveti, G. Su, J. Nieh, "The design and implementation of Zap: A system for migrating computing environments", Proc. Of 5th USENIX Symposium on Operating Systems Design and Implementation, pp: 361-376, 2002.

Keywords: virtual machine monitor, virtual machine, process migration, environment migration, transparent migration, user mobility, hardware virtualization, process domain abstraction (pod), network virtualization, checkpoint-restart

The authors showed that the process migration by OS-level VMs in this paper.

The author pointed out process migration at user-level does not support well of some system-level facilities provides by Operating Systems such as inter-process communication. This paper introduced *pods*, which are groups of processes on a thin virtualization layer and can be viewed as a virtualized system. The author insisted that independent and self-contained characteristic of *pods*, transparent migrations of unmodified applications are possible in this context.

- [29]. ** (Peterson, 2002) L. Peterson, T. Anderson, D. Culler, and T. Roscoe, "A Blueprint for Introducing Disruptive Technology into the Internet", *pp: 59-64, Proc. Of ACM HotNets-I, 2002.*

Keywords: overlay network, internet, virtual machine, virtual machine monitor, service oriented network architecture, overlay testbeds, distributed network, slice, topology probing, unbundled management

In this paper, the authors suggested the methodology to design, evaluate, deploy geographically distributed network service using overlay-based testbeds in order to facilitate new service oriented network architecture. The authors suggested the components of overlay as virtual machine monitors which run on each node and a management service in order to control the overlay. The authors insisted that this overlay should support distributed virtualization allows each application to be acquired and run in a slice of the overlay rather than be globally scheduled to run. The authors insisted that the Internet was originally an overlay network as well over the underlying telephony systems and both architectures would evolve by an interaction and an integration to support new service oriented network architecture.

- [30]. (Reumann, 2000) J. Reumann, A. Mehra, K. Shin, D. Kandlur, “Virtual Services: A New Abstraction for Server Consolidation”, *Proc. Of USENIX 2000 Annual Technical Conference*, webpages : <http://www.usenix.org/publications/library/proceedings/usenix2000/general/reumann/reumann.pdf> 2000.

Keywords: virtual service, resource partition, performance isolation, distributed server farm, virtual host, virtual server, dynamic resource binding, *fair shares*, service sharing, hardware multiplex

The author introduced *virtual services* to insulate the interference between competing applications. The author pointed out virtual hosting and virtual server technology, which splits one physical host into several virtual hosts, should be refined to provide performance isolation at the OS-kernel level. The author demonstrated *virtual service* as an operating system abstraction and provides resource partitioning for each service with an application transparent way by a dynamic binding for service activities. The author insisted that this mechanism allows each service to have an illusion that they are running on dedicated physical server.

- [31]. (Robin, 2000) J.S. Robin, C.E. Irvine, “Analysis of the intel pentium's ability to support a secure virtual machine monitor”, *Proc. Of the 9th USENIX Security Symposium*, webpage: http://www.usenix.org/publications/library/proceedings/sec2000/full_papers/robin/robin.pdf, 2000.

Keywords : virtual machine, virtual machine monitor(VMM), VMM security, privileged instructions, non-privileged instruction, protected processor, CPU virtualization, I/O virtualization, network virtualization, Intel processor virtualization

In this paper, the author focused on secure virtual machine monitor for an Intel-based platform. The author classified sensitive instructions and unprivileged instruction on Intel-based platform and showed the strategy to implement secure virtual machine monitor.

The authors define hardware requirements or a secure VMM and analyzed Intel Pentium architecture with respect to these hardware requirements. The author suggested slight modifications for an Intel based processor to support a secure virtual machine monitor.

- [32]. (Sapuntzakis, 2002) C. Sapuntzakis, R. Chandra, B. Pfaff, J. Chow, M. Lam, M. Rosenblum, "Optimizing the Migration of Virtual Computers", *ACM SIGOPS Operating Systems Review*, vol. 36, Issue: SI, pp:377-390, 2002.

Keywords : virtual machine monitor, virtual machine, optimization, *capsules*, COW disks, *ballooning zero*, hash cache, hash-based compression, environment migration, user mobility

In this paper, the author focused on quickly moving the state of a running computer which includes the state in its disks, memory, CPU registers, and I/O devices across low bandwidth network. The authors suggested *capsules* that can be dynamically instantiated as computation caches for arbitrary, legacy applications. The authors showed that legacy applications on a virtual computer can migrate and continue to operate correctly without residual dependencies. The author suggested optimized capsules migration on a virtual computer with copy-on-write disks, *ballooning zeroes* technique and hash.

- [33]. (Schmidt, 2000) B. K. Schmidt, "Supporting Ubiquitous Computing with Stateless Consoles and Computation Caches", *PhD thesis, Department of CS, Stanford University, webpage: <http://suif.stanford.edu/papers/schmidt00.ps.gz>, 2000.*

Keywords : user mobility, fault tolerance, stateless consoles, computation caches, environment migration, SLIM architecture, scalability, remote access, persistent computation, migrating active computations

The author suggested remote computational service architecture based on stateless display consoles and cacheable compute sessions. Author suggested these consoles can be connected to session servers via display network. According to the author, active sessions can migrate between session servers and these session servers provide an access to high performance back-end servers which might support clustering and load balancing. The author also suggested *compute capsules* which represents an active computation state and this allows computing sessions to be migrated around resource pool and remote access to persistent computation.

[34]. (Smith, 2001) J.E. Smith, “An overview of virtual machine architectures”, *Technical Report, University of Wisconsin*, webpage: *webpage : http://swig.stanford.edu/~fox/cs241/readings/smith_vm_overview.pdf*, 2001.

Keywords: virtual machine, System ISA, User ISA, ISA VMs, ABI VMs, Virtual Machine Monitor (VMM), High level language VMs, Classic OS VMs, OS translator, replicated virtual machine

In this paper, the author categorized two major types of virtual machines, which are ISA VMs and ABI VMs. The author showed there are two key interfaces in a typical computer system; respectively, these are ISA and ABI interface. The author demonstrated that the ISA interface consists of both User ISA (non-privileged instruction set architecture) and System ISA (privileged instruction set architecture). The User ISA is available to both the Operating System and application software. The System ISA is only

available to the Operating System; only privileged operations can be permitted to manipulate the processor, memory and I/O directly. The main components of ABI interface are User ISA and System-calls. Between application programs and an operating system, the System-calls interface is provided to manage and protect hardware resources from unauthorized accesses. ISA VMs manipulate and support both User ISA and System ISA, whereas ABI VMs manipulate and support both User ISA and System-calls.

- [35]. (Sugerman, 2001) J. Sugerman, G. Venkitachalam, B.H. Lim, “Virtualizing I/O Devices on VMware Workstation's Hosted Virtual Machine Monitor”, *USENIX Annual Technical Conference*, pp: 1-14, *USENIX Association*, 2001.

Keywords : virtual machine, virtual machine monitor, I/O Virtualization, device virtualization, virtualized NIC, CPU Virtualization, non-virtualized processor, optimizing I/O virtualization, CPU virtualization overhead, hardware abstraction layer

In this paper, the authors stated that VMM should intercept VMs' device accesses from VMs and forward them to physical devices to virtualize I/O devices. The author pointed out that traditional mainframe hardware, especially processor is designed to be virtualizable, but the Intel IA-32 based Processor architecture is not basically full virtualizable, because some x86 instructions do not trap in user-mode such as accessing the interrupt-enabled flag. The author's approach to virtualize Intel based processor is to insert manual traps by binary rewriting for an emulation to a full virtualization.

- [36]. (Sundaraj, 2003) . Sundararaj, P. Dinda, “Towards Virtual Networks for Virtual Machine Grid Computing”, *Technical Report NWU- CS-03-27, Department of Computer Science, Northwestern University, webpages :*

<http://www.cs.northwestern.edu/~ais/nwu-cs-03-27.pdf>
, 2003

Keywords : virtual machine, computational grid, VLANs, VPNs, VNET, overlay networks, bridged networks, adaptive overlay, monitoring VMs, Virtual Machine Monitor(VMM)

In this paper, the authors suggested VNET architecture, which is a virtual private network that implements a virtual local area network using layer 2 tunneling over a wide area network. VNET allows users to transfer VM images to their local area with the illusion that user's virtual machines are on the user's local area network. The authors also suggested three layers architecture to conceptualize VM networks, those are Physical Layer, VMD(Virtual Machine Daemon) layer and VM layer. Physical network is an underlying IP network and VMD layer is an overlay in this architecture and is able to manage VMs in VM layer and monitor both the resource provided by underlying physical network and resource requested by VMs in VM layer. The author pointed out monitoring by VMD layer, it is possible to adapt communication and computation behavior of the VMs in VM layer and change its topology and routing rules to efficiently migrate VMs.

[37]. (Waldspurger, 2002) J. Weissman, B. Lee, "The Virtual Service Grid: an architecture for delivering high-end network services", *Concurrency: Practice and Experience*, vol. 14, issue: 4, pp: 287-319, 2002.

Keywords : virtual machine, virtual machine monitor, memory management, memory virtualization, *ballooning technique*, *idle memory tax technique*, transparent page sharing, content-based page sharing, dynamic reallocation, I/O page remapping

The author introduced VMM-based ESX server mechanisms for a memory management.

A *ballooning technique* is to inflate and deflate memory pages for VM by ESX Server. When the ESX Server wants to reclaim memory for VMs, ESX Server use ballooning technique and inflate memory pages within a VM and allocated to physical pages. When the ESX Server wants to deallocate memory for VMs, the ESX Server deflate balloon by an instruction and deallocate previously-allocated pages. The author also suggested an idle memory tax which specifies the maximum fraction of idle pages that may be requested from a client. The allocation enables client to use a larger portion of its allocated memory, but it should not exceed full share.

- [38]. (Weissman, 2002) J. Weissman, B. Lee, “The Virtual Service Grid: an architecture for delivering high-end network services”, *Concurrency: Practice and Experience*, vol. 14, issue: 4, pp: 287-319, 2002.

Keywords: Grid, Virtual Service, Virtual Service Grid, high performance, cache, replication, dynamic replica, high-end network services, scalable performance, adaptive dynamic replica selection

The author suggested Virtual Service Grid to deliver high performance network services. The author insisted that caching and replication is essential to provide scalable network services. The author demonstrated caching is useful when the service requested is simply a retrieval of server contents, meanwhile, replication is useful for specialized processing is required. The author insisted that Virtual Service Grid provides a scalable performance by dynamic replica management techniques. The author insisted that dynamic replication with a small amount of pre-allocation improved performance significantly for high performance network services.

- [39]. (Whitaker a, 2002) A. Whitaker, M. Shaw, and S. Gribble, "Denali: Lightweight virtual machines for distributed and networked applications", *Proc. Of the USENIX Technical Conference*, webpage: http://www.cs.ucsb.edu/~rich/class/cs595-os/denali_usenix2002.pdf, 2002

Keywords: virtual machine, virtual machine monitor (VMM), isolation kernel, *paravirtualization*, hypervisors, performance isolation, security isolation, microkernel, virtual hosting, VMM scalability

In this paper, the author adapts *paravirtualization* which virtualizes a subset of the processor's instruction set and specialized virtual devices to enhance performance and can be hosted hundreds and thousands guest OSes on a VMM in a single machine. The authors suggested Denali system using an *isolation kernel* which is similar to VMM but it has basic difference such as modified OSes should be ported for an isolation kernel to improve performance. Isolation kernel has its own memory management mechanism and interrupts to improve performance. Isolation kernel is resident in physical memory, but VMs are paged on demand. Whenever page fault is taken, the isolation kernel verifies virtual address and allocates new page tables and initiates a read action from the VM's swap region. The isolation kernel system periodically redistributes physical memory from inactive VMs to active VMs. The author stated thousands of the guest OSes can be hosted on an *isolation kernel*.

- [40]. (Yang, 2003) K. Yang, X. Guo, A. Galis, B. Yang, D. Liu, "Towards efficient resource on-demand in Grid Computing", *ACM SIGOPS Operating Systems Review*, vol.37 Issue:2, pp:37-43, 2003

Keywords : Grid Computing, Resource on Demand (RoD), Efficiency, Quality of Service (QoS), Active Networks (AN), Policy-based Management (PBM), Grid resource allocation, Grid resource management, virtual Grid resources, physical Grid resources

In this paper, the authors insisted one attribute of Grid Computing is its Resource on-Demand (RoD) capability, which transparently provides the Grid resources needed for Grid applications or for services with QoS (Quality of Service). The authors claimed sophisticated policy-based management methods are required in order to establish efficient Resource On Demand (ROD) in Grid environments; providing efficient QoS mechanisms inside the Grid networking environments is factor in achieving a key this goal.

Appendix-III

List of Researchers

Dr. Renato J. Figueiredo

Assistant Professor
Department of Electrical and Computer Engineering
University of Florida
P.O. Box 116200, 336 Larsen Hall, Gainesville, FL, 32611
Phone: (352)392-6430
Homepage: <http://www.acis.ufl.edu/~renato>
E-mail: renato@acis.ufl.edu

Dr. Jos'e A. B. Fortes

Professor and BellSouth Eminent Scholar
Electrical and Computer Engineering
University of Florida
P.O. Box 116200, 339 Larsen Hall, Gainesville, FL, 32611-6200
Phone : (352)392-9265
Homepage : <http://www.acis.ufl.edu/fortes>

Dr. Michael A. Kozuch

Intel Research Pittsburgh
417 S. Craig Street
Pittsburgh, PA 15213
Homepage : <http://info.pittsburgh.intel-research.net/people/makozuch/>

Dr. Larry L. Peterson
Professor

Department of Computer Science
35 Olden Street
Princeton, NJ 08544 Office: CS 219
Phone: 609-258-6077
Homepage : <http://www.cs.princeton.edu/~llp/>
E-mail: llp@cs.princeton.edu

Dr. Ian Foster

Professor of Computer Science
The University of Chicago
1100 E. 58th Street
Ryerson Hall
Room 155
Chicago, IL 60637
Phone: (773)702-3487
Homepage : <http://www-fp.mcs.anl.gov/~foster/>
E-mail: foster@cs.uchicago.edu

Dr. A. Campbell

Department of Electrical Engineering
and Center for Telecommunications Research
1312 Seeley W. Mudd Bldg.
Columbia University
New York, NY 10027-6699
Phone: (212) 854-0871, (212) 932 9421
Homepage: <http://www.ee.columbia.edu/people/campbell.php3>
E-mail : campbell@comet.columbia.edu

Appendix-IV

Forthcoming conferences

(VM' 04)

3rd virtual machine research and technology symposium

Date: May 6-7, 2004

Location: SANJOSE, CALIFORNIA, USA

Sponsored by USENIX in cooperation with ACM SIGPLAN

Program Committee :

Henri Bal, *Vrije Universiteit, The Netherlands*

Robert Berry, *IBM, UK*

Hans Boehm, *HP Labs, USA*

Michal Cierniak, *Intel, USA*

Stephen Fink, *IBM, USA*

Etienne Gagnon, *University of Quebec at Montreal, Canada*

John Gough, *Queensland University of Technology Australia*

Sam Midkiff, *Purdue University, USA*

David Tarditi, *Microsoft, USA*

David Ungar, *Sun Microsystems, USA*

Matt Welsh, *University of California, Berkeley, USA*

Saul Wold, *Sun Microsystems, USA*

Web Page : <http://www.usenix.org/events/vm04/>

(SIGCOMM 2004)

ACM SIGCOMM Conference 2004

ACM SIGCOMM Special Interest Group on Data Communications

Location : Portland, Oregon, USA

Date : Aug.30 - Sept. 3, 2004

Web page : <http://www.acm.org/sigcomm/sigcomm2004/>

(ICDCS 2004)

The 24th International Conference on Distributed Computing Systems

Sponsor : IEEE Computer Society

Location : Tokyo, japan

Date : March 23-26, 2004

Web page : <http://www.cis.ohio-state.edu/icdcs04/>

(CCGRID2004)

4th IEEE/ACM International Symposium on Cluster Computing and the Grid

Location : Chicago, Illinois, USA

Date : April 19-22, 2004

Web page : <http://www-fp.mcs.anl.gov/ccgrid2004/>

(NSDI, 04)

First Symposium on Networked Systems Design and Implementation

Location : San Francisco, California, USA

Date : March 29-31, 2004

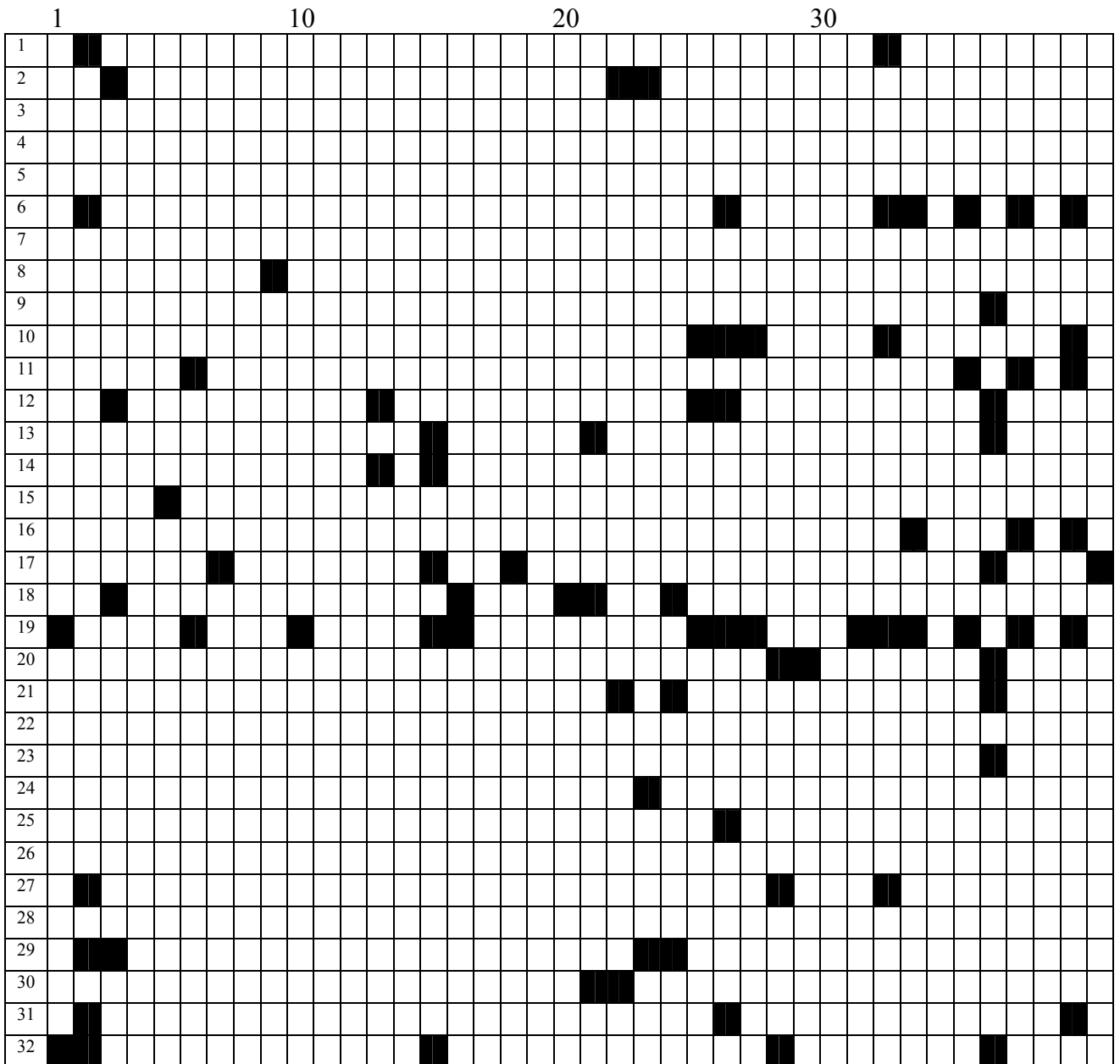
Sponsor : USENIX, ACM SIGCOMM, ACM SIGOPS

Webpage : <http://www.usenix.org/events/nsdi04/>

Appendix V : Cross Referencing Graph

(X as referenced by Y; X = papers in horizontal, Y= papers in vertical)

Milestone Papers : [6], [10], [17],[18], [19] Important Papers: [13], [18],[19],[27],[29]



Appendix VI : Email sent to researchers

UNIVERSITY
of WINDSOR

student webmail

Mailbox **Compose** **Rules** **Settings** **External** **Feedback** **Log Out**



From: "Ananth I. Sundararaj" <ais@cs.northwestern.edu>
Subject: Re: I would like to ask your opinion about VMs in P2P Network
Date: Thu, 1 Apr 2004 19:43:11 -0600 (CST)
To: Dohan Kim <kimw@uwindsor.ca>

Hi Dohan!

How are you doing? VNET is now publically available from
<http://virtuoso.cs.northwestern.edu>

The release contains a detailed README and all the source code. I do not know the exact scenario in which it might be of use to you.

But please do let me know if you run into problems while using it.

I look forward to your comments and suggestions on the same. As detailed in the paper, we are now working on a second generation VNET that will support arbitray topologies and routing.

In the next iteration the VNET daemons will form a peer-to-peer overlay. I will keep you informed about the upcoming releases.

Regards,
Ananth

On Fri, 30 Jan 2004, Dohan Kim wrote:

> Dear Mr.Sundararaj,
>
> I am a masters student at University of Windsor,
> Windsor,Ontario, Canada.
> I am doing a survey on "flexible virtual machine in
> distributed systems and Grids"
> I read your paper "Towards Virtual Networks for Virtual
> Machine Grid Computing", and I am really interested in your
> research.
> I am also researching virtual machine in P2P Network. I am
> wondering if your 'VNET' can be applied on scalable P2P
> network. Could you give me some information about this?
>
> I really appreciate your time to read this email.
>

> Best wishes,
> Dohan Kim
>
> University of Windsor, Canada
>

