The Virtualization Landscape

N.D. Jebessa
University of Amsterdam
http://www.science.uva.nl/~ndj/

October 19, 2012

Abstract

The state of the art in virtualization research is presented here. This should be useful for someone who is starting with research in the area. The motivation for this review is the absence of a fairly comprehensive summary of the subject with pointers to relevant work. The paper discusses the meaning of virtual machines, historical perspectives as well as references to major players in industry and academia. Popular implementations are categorized in a VM taxonomy. Virtualized resources such as servers, devices, storage, networks that comprise virtual computer systems are discussed from a research point of view. Major topics in virtualization research are also covered.

1 Overview

Modern computer systems are complex structures containing numerous closely interacting components in hardware and software [118] as well as networks and data. This complexity gives rise to challenging problems in several domains. David Wheeler famously commented that 'all problems in computer science can be solved by another level of indirection' [120], often deliberately misquoted with 'abstraction' substituted for 'indirection'. Abstraction layers [30] and structured organization [126] have long been used by hardware, OS and application software designers to work independently. As J.E. Smith puts it [118], virtualization acts as an interconnection technology, interjected between abstraction layers near the HW/SW interface hence forming a virtual machine [48]. Virtualization is a key technology that is being used to

address computer systems problems like security, performance optimization, system administration, and reliability, among others [37]. Computing models like Infrastructure-as-a-Service (IaaS) clouds extensively employ processor, memory, storage and network virtualization techniques. The same technology is increasingly being used in system design, testing [84], grids [38, 108] and mobile computing environments [113].

2 Virtual Computer Systems

2.1 Formal Definition

Even in the 1960s and 1970s, there was extensive research in virutal machines (see the excellent survey by Goldberg [48]) and machines like the IBM 370 supported virtualization at the architecture level [27]. A formal machine model to test whether an architecture supports virtual machines is developed by Popek and Goldberg [98]. They define a virtual machine to be *an efficient, isolated duplicate of the real machine*. The notions of *equivalence*, *efficiency* and *resource control* are explained through the idea of a *virtual machine monitor*. The formal technique employed by Popek and Goldberg also extends to *recursive virtualization*.

2.2 A Bit of History

Research efforts like Multics [26] (MIT, 1962) show the significance of time sharing and resource multiplexing for efficient use of the expensive mainframes of the day. By the end of the 60s, virtual machines [47] were being explored in industry and academia [48, 107]. The IBM VM/370 [27] operating system (1972) running on System/370 hardware was an early multi-user environment providing users with their own copies of the underlying hardware through the use of virtual machine technology.
Multitasking operating systems and cheaper hardware in the 1980s and 1990s devalued virtual machine monitors, as consumers could get access to minicomputers and PCs instead of expensive mainframes. This essentially meant that computer architectures need not support virtualization and, unsurprisingly, popular x86 processors from Intel and AMD didn't support virtualization until late 2000s [130, 94, 2, 1] through VT-x and AMD-V, respectively. In the mid 1990s, Bressoud and Schneider [17] were one of the first to re-

introduce virtual machine monitors, particularly for fault-tolerant computing. They used the word *hypervisor* to mean a software layer that implements same ISA virtual machines [118]. In this paper, hypervisor and VMM (virtual machine monitor or manager) are used interchangeably[1]. Bugnion, *et al.* introduced Disco [18] where the virtual machine monitor idea was used to run multiple operating systems in a multiprocessor system. This idea and the researchers led to the founding of VMware in 1998 which subsequently was able to virtualize x86 processors using binary translation [116] techniques. The company has since been a major player in the virtualization market. The now popular Xen virtual machine monitor [11] was introduced in 2003. The ubiquitous use of VM technology in mobility, manageability, security and cost reduction through server consolidation must have convinced processor vendors like Intel to bring back support of virtualization in hardware [94]; without which sophisticated software techniques had to be used. Dong, *et al.* [33] explain how Xen was extended to use hardware support. A comparison of software/hardware techniques for virtualization is given by Adams and Agesen [2].

Currently there are several examples of virtual computer systems ranging from virtual machines on desktops[2] to virtual computing resources delivered over the Internet via the cloud model [6, 76]. Key market players in the cloud domain include Amazon Web Services, Rackspace (the co-creator of the OpenStack cloud OS), Savvis, Salesforce.com, Terremark, Joyent, Citrix (the company behind XenServer, XenDesktop and CloudStack open source cloud computing), Microsoft (with Azure), VMware (with its vCloud product, Cloud Foundry open source PaaS and companies like Bluelock providing virtual datacenter solutions based on vCloud), IBM (with SmartCloud) and Google (with its Docs, Compute Engine, App Engine and Cloud Storage services among others). In addition to commercial virtual infrastructure providers, open source projects like OpenStack, Eucalyptus, Nimbus and OpenNebula are used to build private/hybrid/public clouds. Most of these cloud platforms are hypervisor agnostic in the sense that they support a number of hypervisors - KVM/QEMU, Xen, LXC, OpenVZ, UML (User Mode Linux [31]), VirtualBox, VMware ESX and GSX, and Hyper-V, via

[1]The hypervisor is sometimes taken to be the kernel of a VMM, i.e. the core or privileged part. See http://www.ok-labs.com/blog/entry/is-there-a-difference-between-a-virtual-machine-monitor-and-a-hypervisor/ for more.

[2]VMware, VirtualBox and Parallels among others.

a virtualization library like libvirt[3]. In addition to virtualization, enabling technologies for the cloud as we know it include structured storage, web services, virtual appliances, computational market dynamics [132, 51] and the Internet itself.

3 Virtual Machines: Architecture and Taxonomy

Smith and Nair [118] explain the architecture of virtual machines (VMs) that support individual processes (process VMs) or a complete system (system VMs). Further looking at the purpose of the VM like flexible hardware usage, software isolation, instruction set architecture (ISA) translation, the authors develop a taxonomy of virtual machines. The same authors describe virtual machines as versatile platforms for systems and processes in their 2005 book [117]. A 1973 paper by Goldberg [47] presents a model which represents the addressing of resources by processes executing on a virtual machine. The model is then used to describe and characterize various VM designs.
The taxonomy presented in [118] divides VMs into either process or system VMs. ISA[4] simulation is the major basis of their taxonomy. In the two major categories (i.e. process and system VMs), the authors further distinguish VMs according to whether they use the same ISA or a different one.

3.1 Same ISA Process VMs

Multiprogramming is so common in contemporary operating systems that few consider the OS itself as a VM [118]. The OS, while performing resource management and multiplexing, gives each process its own address space, registers and file structure, essentially serving as a process virtual machine.
Other examples of process VMs include *emulators* (like Intel's IA32-Execution Layer [10] which allows IA-32 binaries to be run on Itanium hardware and the Wine[5] emulator which is used to run Windows applications on Linux, BSD, Solaris and Mac OS X) and *dynamic binary translators* which are used to improve performance of emulation by way of block instruction translation

[3] `http://libvirt.org`
[4] Instruction Set Architecture
[5] WineHQ homepage - http://www.winehq.org/

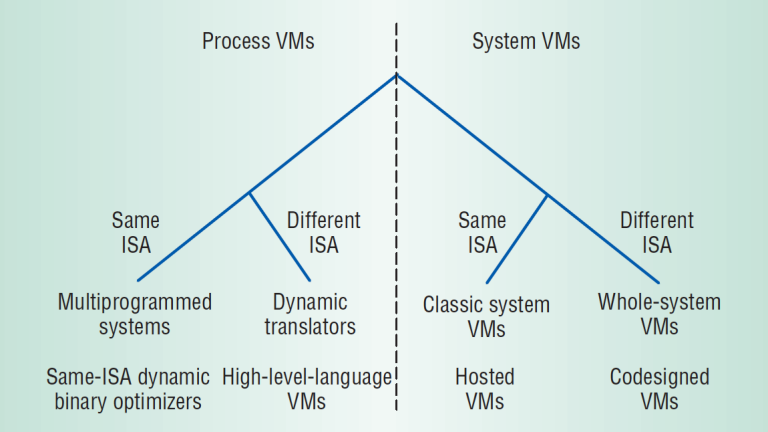

Figure 1: VM taxonomy [118]

and software caching. This may include binary optimization as in the case of Dynamo [7], for example.

3.2 Different ISA Process VMs

High-level language (HLL) virtual machines aim to provide portability in the write-once, run-anywhere model. The VM supports a virtual ISA that code is compiled to. An interpreter in the VM runs the intermediate code on a target platform. The Common Language Infrastructure (CLI) [87] and Java [80] are popular HLL VMs. In [86] the author discusses UCSD p-System for Pascal and its legacy in the likes of languages like Java. p-System was developed in the early 70s as a multiplatform OS that implemented a virtual "p-machine". A decade earlier, UNCOL [32] aimed to serve as language between problem-oriented languages and machine code.

3.3 Same ISA System VMs

Classic system VMs [118] (also known as *Type 1* [48], *native* or *bare metal* hypervisors) run directly on the underlying hardware. As such, such a VMM controls the hardware and manages guest operating systems which run on a layer above the VMM itself. Recent examples include Microsoft Hyper-V, VMware ESXi and Citrix XenServer.
Hosted VMs [118] (also known as *Type 2* [48] hypervisors) run on top of

an existing operating system, installed much like an application program. Notable examples are Oracle VirtualBox, Microsoft Virtual PC and VMware Workstation.

3.4 Different ISA System VMs

Whole-system VMs [118] like QEMU [13] allow an unmodified guest operating system to be run on a possibly different target architecture. Early versions of Virtual PC [128] allowed Windows OS compiled for an Intel processor to run on Apple's then PowerPC-based systems.
While most system VMs are implemented on hardware developed for some standard ISA (like the x86 or ARM), *codesigned VMs* [118] implement new ISAs targeted at improving performance, power efficiency or both. Such a VM would appear to be part of the hardware implementing it. A well-known example of codesigned VMs is the Transmeta Crusoe [29] processor, which used a VLIW implementation of the x86 architecture for power saving. Another one is AS/400 [119] (now part of IBM's Power Systems) which supported an object-based instruction set.

4 Virtualized Resources

4.1 Servers

Virtualization allows for the creation of *virtual servers*. Virtual web server providers extensively use virtual server solutions. Noteworthy solutions include FreeBSD Jails, Solaris Zones, products from Ensim[6], Linux V-server[7], Parallels Virtuozzo Containers, OpenVZ and LXC.
FreeBSD Jails [64] provides the ability to partition the operating system environment, while maintaining the simplicity of the UNIX "root" model. In this OS-level virtualization, a 'jail' is a virtual environment with own files, processes, user and superuser accounts, completely isolated and functionally indistinguishable from a real FreeBSD environment.
Solaris Zones [100] aims at server consolidation with a focus on the support of multiple isolated application environments rather than multiple operating system instances.

[6] `http://www.ensim.com`
[7] `http://www.linux-vserver.org`

In [139] the authors present FVM, an OS-level virtual machine for Windows using a technique called namespace virtualization wherein isolation is provided by renaming resources at the system call interface.

4.2 Devices

I/O device virtualization is not an easy task. Sugerman, *et al.* [123] describe how VMware Workstation virtualizes I/O. They use a hosted architecture where a user-level program (VMApp) uses the host to virtualize the I/O devices while a separate, privileged VMM provides high performance CPU virtualization. They report on performance of hosted I/O virtualization vs. native I/O and suggest optimization techniques to reduce CPU utilization while performing I/O. The authors in [67] suggest extensions to a host operating system (UMLinux) so as to reduce virtualization overhead.
Fraser, *et al.* [39] propose a next-generation architecture that aims to solve three problems in PCs namely dependability (due to lack of isolation of driver and OS code), maintainability (device driver code needs to be rewritten for each OS) and manageability (troubleshooting and maintenance of driver related problems is difficult and time consuming). They use the Xen VMM to demonstrate a 'Safe Device Architecture' wherein devices and drivers are isolated from the OS and each other. This allows a single driver to be used under any number of operating systems.
LeVasseur, *et al.* employ virtual machines to reuse unmodified drivers and improve system dependability [77]. In addition to reuse, running unmodified drivers with their original OS in VMs provides isolation from faults caused by defective or malicious drivers. The authors report network performance within 3-8% of native Linux and a 0.12% increase in CPU utilization per VM. The techniques they used can simplify new OS development efforts.
Whitaker, *et al.* present the μDenali VMM [133] as a modification to the Denali isolation kernel [134, 135]. They describe μDenali as an extensible and programmable VMM. Two techniques are described: *extension* (the ability to dynamically assemble a virtual machine out of either default or custom-built virtual hardware elements) and *interposition* (a device in a certain VM being bound to virtual hardware in another VM via an event-routing framework). This way the μDenali VMM allows simplified creation of VM-based services like the authors' implementation of Internet Suspend/Resume [113, 133], a network intrusion detection virtual appliance, and a continuous rejuvenation framework for the Apache web server.

A number of authors have examined the performance overhead due to virtualized devices [89, 22] and suggested software and hardware-assisted techniques to optimize performance [88, 81, 1, 102, 136, 19].

Xenoprof is a system-wide statistical profiling tool for the Xen VMM introduced in [89] where it is used to analyze performance overheads of network I/O due to applications running inside Xen VMs. Using different workloads and Xen configurations, the authors identify key areas for Xen optimization efforts as well as the use of Xenoprof to resolve performance bugs in Xen-based environments. A similar work by Cherkaskova and Gardner [22] discusses a lightweight monitoring framework for Xen to measure the CPU usage of VMs running inside Xen as well as the CPU overhead due to I/O processing in each VM.

In addition to profiling [89] and monitoring [22] software and hardware mechanisms have been proposed to optimize I/O performance. As server environments extensively use network and disk in I/O devices, most experiments [89, 88] use such devices while a few others have explored graphics [19].

Liu, *et al.* [81] propose a device virtualization model called *VMM-bypass* I/O. Time-critical operations like Infiniband network I/O benefit hugely from such a model as the VMM is not involved while operations are being carried out. The authors' prototype implementation, Xen-IB, nearly achieves raw performance.

Another interesting approach is the concept of a *self-virtualized device* suggested by Raj and Schwan [102]. This improves I/O performance by offloading virtualization functionality to the device itself. The authors show the feasibility of such an approach through SV-NIC, a self-virtualizing NIC that exports virtual network interfaces to guest VMs. Commercial solutions with similar approaches can now be found from networking vendors like Cisco[8]. Processor vendors like Intel [1] support I/O virtualization at the hardware level.

The CDNA (Concurrent Direct Network Access) [136] architecture aims to improve network efficiency and performance through a novel hardware/software partitioning of tasks like traffic multiplexing, interrupt delivery and memory protection; essentially eliminating software bottlenecks.

Lagar-Cavilla, *et al.* describe VMGL [19], a cross-platform OpenGL graphics virtualization solutions that is independent of the VMM and GPU. With sci-

[8]See Cisco's Virtual Interface Card that exports multiple PCIe endpoints with support for Single-Root I/O Virtualization (SR-IOV)

entific applications increasingly using GPUs as accelerators and with promising results in GPU sharing in virtualized environments [103], VMs hosting such applications can benefit from such a solution.

4.3 Storage

Storage system design is a complex endeavor specially in an enterprise or data center with hundreds of computers and thousands of disks. Manual design of a seamlessly integrated storage infrastructure can be a cumbersome task leading to cost inefficiency, sub-optimal performance and lack of flexibility. When logical volumes, storage area networks (SANs) and RAID infrastructure are incorporated, this becomes even more complex. In such a case, experts opt for automated design tools like MINERVA [3], a suite of tools that uses declarative specifications of application requirements and device capabilities; constraint-based formulations of the various sub-problems; and optimization techniques to explore the search space of possible solutions. While tools like MINERVA achieve better performance than would have been possible through manual designs of storage experts, the problem becomes quite involved when one considers distributed systems and the networking aspect of storage. To this end, the authors in [12] employ the end-to-end principle [110] for scalable network storage.

Façade [82] is a virtual store controller with performance guarantees to independent, competing clients. With a similar aim of QoS as Façade, Stonehenge [54] virtualizes a cluster-based storage system. While many storage virtualization solutions focus only on the capacity storage attribute, Stonehenge is able to virtualize attributes like bandwidth and latency as well, multiplexing multiple *virtual disks* in what the authors call *multi-dimensional* storage virtualization.

Pfaff, *et al.* [97] go beyond limitations of virtual disks and introduce Ventana, a virtualization aware file system. The authors argue that the low-level interface of virtual disks is *coarse-grained*, forcing all-or-nothing operations, and *opaque*, offering no practical means of sharing. Ventana is a distributed file system that provides the powerful versioning, security and mobility properties of virtual disks that make them so compelling while overcoming the limitation of coarse-grained versioning and opacity for cooperative sharing. The design abstractions are *branches* as in version control systems; *views* that resemble the mounting concept in UNIX; *access control* with ACLs for files, branches and versions; and *disconnected operation* hence supporting a

simple model of mobility.

4.4 Networks

Network virtualization is a very important enabler for virtual infrastructures. Virtual private networks (VPNs), virtual local-area networks (VLANs) and overlay networks are commonly used mechanisms in this regard.
Ferguson and Huston [35] describe different approaches to build a VPN. RFC 2764 [46] describes a framework for VPNs running across IP backbones. It discusses the various types of VPNs, their respective requirements, and proposes specific mechanisms that could be used to implement each type of VPN using existing or proposed specifications. Secure VPN protocols include *IPSec*; Microsoft *MPPE* used with the Point-to-Point Tunneling Protocol; *SSL/TLS* as used in OpenVPN; and *SSH* as used in OpenSSH.
As VPNs aim to emulate the services provided by a private network over the shared Internet, the reliability of an end-to-end connection poses interesting technical challenges. The authors in [56] present a polynomial time restoration algorithm for fast recovery from link failures between VPN endpoints connected using a tree structure.
An *overlay network* is a network consisting of nodes implementing a network abstraction on top of the network provided by the underlying substrate. Jannotti, *et al.* [57] present a multicast protocol called Overcast as a novel use of overlay networks. Their solution is a reliable, scalable and application-level multicasting system. Narada [23] is a somewhat similar solution for efficient multicasting by shifting the role from routers to end systems. The authors report low performance penalties. An interesting contribution of their paper is a self-organizing and self-improving protocol on top of a dynamic, unpredictable and heterogeneous Internet without relying on a native multicast medium.
Resiliency in overlay networks has been studied early by Andersen, *et al.* [5] and later by Banerjee, *et al.* [8] among others. RONs (Resilient Overlay Networks) allow distributed Internet applications to fast detection and recovery from path outages and periods of degraded performance. Banerjee, *et al.* [8] present Probabilistic Resilient Multicast (PRM), a fast multicast data recovery scheme that achieves high delivery ratios with low overheads. The authors report better performance (i.e. low overheads and low recovery latency) than protocols like Overcast [57]. Their solution can also augment application-layer solutions like Narada [23].

Sundararaj and Dinda [125] introduce a layer 2 virtual network tool called VNET as part of their Virtuoso middleware for grid computing using virtual machines. Related work in this area include the Collective [111] middleware which creates virtual appliance networks (VANs); the SODA [108] virtual network (VIOLIN [59]); IP over P2P (IPOP) [41, 42] used by WOW [40] which is a distributed system that combines virtual machine, overlay networking and peer-to-peer techniques to create scalable wide-area networks of virtual workstations for high-throughput computing; ViNe [129]; VTL [73] and OverQoS [122] to mention a few.

4.5 Remote Display

Visual output from VMs running remotely can be transferred through protocols like VNC [104], Microsoft RDP and THINC [9], among others. The performance of thin client computing is discussed in the SLIM [114] stateless thin-client architecture and the paper by Lai and Nieh [71] discusses the limits of wide-area thin-client computing. Speculative remote display is one of the techniques suggested to improve performance of remote display. The authors in [74] report on their experience using client-based speculative remote display.

5 Some Topics in VM Research

5.1 VM Introspection and Monitoring

The black box nature of virtual machines and their isolation calls for an insight into what's happening inside a VM. One early work in this area is state sampling of interactive VM/370 users [127] using a tool called VM/Monitor to understand delays caused by resource contention. The techniques are used to detect bottlenecks and system tuning.
Recent research includes parallel program topology inference [49], application performance improvement [124], optical network reservations [75], VM scheduling [79], autonomic virtual applications [138], network measurement [50], the AntFarm process tracker [60],the Geiger page cache monitor [61] and architectural approaches like*process out-grafting* [**?**] for process-level execution monitoring outside the VM.
Different approaches to fill the semantic gap between the internals of a run-

ning VM and the underlying VMM have been proposed. While most of the aforementioned papers assume no knowledge about the VM (i.e. the VM is treated as a black box), researchers have proposed gray box approaches as well. Applications include internal VM state reconstruction [101] and VM migration [137] including virtual Trusted Platform Modules (TPM) [**?**, 28].

5.2 Power Management

Virutal machine technology has been extensively used to save power consumption in data centers. Server consolidation allows only the required amount of physical servers hosting virtual servers to be turned on. Research in this area includes scheduling (e.g. temperature-aware workload placement in data centers [90], coordinated power management [92] and power budgeting [93].

5.3 Migration of Virtual Machines

The ability to migrate whole system state between physical machines in the same network or over the Internet is an important enabler. Muthitacharoen, *et al.* [91] introduce LBFS, a network file system designed for low-bandwidth networks. LBFS avoids the same data being sent over the network when the same data can already be found in the server's file system or the client's cache. Kozuch, *et al.* [69] discuss efficient state transfer in ISR (Internet Suspend/Resume [113]).
The paper by Sapuntzakis, *et al.* [112] discusses optimization of virtual computer migration based on VMware. Zap [96] supports legacy and networked application migration through a thin virtualization layer on top of Linux. Boyd and Dasgupta [15] show the use of process migration for off-the-shelf applications on top of a virtualizing operating system by injecting functionality into running processes and the OS, specifically Windows 2000.
Clark, *et al.* [24] demonstrate live migration of virtual machines using a prototype implementation for the Xen VMM. Nelson, *et al.* [95] describe techniques for fast transparent migration of VMs running unmodified, mainstream operating systems. Related work in this area includes that of *Nomad* by Huang, *et al.* [55] for migrating network resources and Braford, *et al.* in WAN migration of VMs. Kumar and Schwan [70] present *Netchannel*, a transparent VMM-level mechanism for efficient and seamless access to I/O

devices. Netchannel allows virtual device migration, device hot swapping as well as remote access to devices.

5.4 Security and Reliability

Madnick and Donovan [83] present one of the earliest papers in the use of VMs for security and isolation. They argue that such an approach provides substantially better security than conventional multiprogramming OS-based approach due to the inherent design of VMM-based systems. The authors give a probabilistic model to compare the security of VMM mechanisms (e.g. VM/370) vs. OS mechanisms (e.g. OS/360).
Another early work is a verifiable protection system by Popek and Kline [99] where the authors describe the UCLA-VM operating system with influencing concepts such as program verification, security kernels, virtual machines and virtual memory.
Almost two decades after the previous two papers were published, Bressoud and Schneider [17] re-introduced the hypervisor concept for fault tolerance.
Chen and Noble [20] argue that the OS and applications that run on a real machine should relocate into a VM. The authors present secure logging, intrusion prevention and migration, and environment migration as use cases. Their paper 'When Virtual is Better Than Real' is later supported by Garfinkel and Rosenblum [45] in their paper 'When Virtual is Harder than Real'. The authors acknowledge the flexibility of virtual machines but note that their inherent difference with real machines with regard to mobility, connectivity and patch cycle calls for a different approach in security architecture.
ReVirt [34] is a VM-based logger and replay system with a reasonable overhead. Livewire [44] uses a somewhat similar approach for VMM-based intrusion detection through an introspection-based architecture. Their approach uses a VMM to isolate the IDS from the monitored VM.
King, *et al.* [66] describe the design and implementation of a time-traveling virtual machine (TTVM) for debugging operating systems. Using the checkpointing, replay and logging mechanism of the TTVM, the authors are able to perform reverse debugging on operating systems so as to find non-deterministic bugs in device drivers, bugs that require long runs to trigger and stack corrupting bugs. The authors implement time travel with gdb (TTVM-gdb) and realize, with reasonable overhead, commands such as reverse breakpoint, revers watchpoint and reverse step.

SecVisor [115] is a tiny hypervisor that provides lifetime kernel code integrity for commodity operating systems. The hypervisor ensures that only user-approved code can execute in privileged mode over the entire system lifetime. This protects the kernel from code injection attacks such as kernel rootkits. The authors use both software memory virtualization and hardware support to implement the hypervisor (1739 and 1112 LOC each, with 2 hypercalls and very minimal effort to port existing kernels). The small size of SecVisor makes it possible to perform formal verification if necessary, as has been done for the seL4 kernel [68]. Lycosid [62] employs VMM-based statistical methods to detect and identify hidden processes such as stealth rootkits.

Chen, *et al.* present Overshadow [21], a virtual machine based system that protects the privacy and integrity of application data even in the event of OS compromise. Overshadow provides cryptographic isolation to the application through a technique called *multi-shadowing* which leverages the extra level of indirection offered by memory virtualization in a VMM to provide a one-to-many mapping between guest addresses to actual machine addresses. The application sees a cleartext view of its memory pages whereas the OS sees an encrypted view, a technique called *cloaking*.

Trusted Computing Base (TCB) of an application in today's layered architectures include the BIOS, firmware, bootloader, the OS kernel and the application code. This complicates the task of securing applications. McCune, *et al.* propose a secure execution architecture that aims to address the limitation of current CPU-based isolation technologies, namely AMD's Secure Virtual Machine (SVM) technology and Intel's Trusted Execution Technology (TXT). Their approach [85] guarantees a minimal TCB for security-sensitive code of an application, i.e. only the CPU, memory and the interface between them, with an optional Trusted Platform Module (TPM) which can itself be virtualized [14]. While the work of McCune, *et al.* focuses on hardware support for TCB minimization, hypervisor based techniques have also been proposed by Lie and Litty [78]. Earlier work by Hohmuth *et al.* [53] compares small kernels versus virtual machine monitors. The Nizza [52] secure system architecture proposed by Härtig, et al. for minimal TCB uses trusted wrappers to reuse legacy OS with VMMs providing the required isolation. Subsequent work by the authors includes that of Bastei [36] and the ongoing Genode[9] OS framework.

[9] `http://www.genode.org`

Systems security researchers in the past few years have utilized virtualization technologies to such an extent that some researchers claim that the use of virtual machines to protect computers has reached a level of orthodoxy [16]. The authors admit that virtualization is an enabler for security solutions but emphasize that it's not a security provider by itself without considering the design and management of the systems, processes and people surrounding it. Roscoe, *et al.* [106] also question whether hypervisors are a disruptive force in OS research. They argue that hypervisors are just re-implementing existing interfaces (e.g. the traditional UNIX interface and a PC-like application binary interface (ABI)) which in the short term is relevant for industry and academic research. Beyond current applications and virtualization research, the authors suggest future OS research directions in avenues like engineering methodologies, formal methods, programming language and architecture support as well as metrics to evaluate systems.

A fairly comprehensive survey of virtual machine based security systems is given in [140]. Notable uses include *VM-based intrusion detection systems* like Livewire [44]; *VM-based intrusion prevention systems* like NSA's NetTop, IntroVirt [63] and sHype [109]; *VM-based honeypots* like Potemkin [131] and Collapsar [58]; *VM-based trusted computing platforms* like Terra [43] and *VM-based logging and replay* like ReVirt [34] and *VM-based malicious software* like SubVirt [65].

The design and implementation of the the hypervisors themselves is also widely researched. In addition to Terra [43], the VAX VMM and sHype [109], numerous other secure hypervisor approaches have been proposed. NOVA [121] described by the authors as a *microhypervisor* due to their motivation in the small TCB of microkernel-based systems is one of many proposed but not widely used hypervisors. Another effort in this domain is Xoar [25], a modification to the Xen VMM that borrows modularity and isolation principles from microkernels. Palacios [72] is a small, non-paravirtualized, configurable and embeddable VMM which makes use of x86 virtualization support. It supports Linux, Kitten and MINIX, among other host operating systems.

The management of VM images is another venue of research. An interesting area is offline patching of dormant images as in Nüwa [141] that leverages IBM's Mirage image library [4]. Richter, *et al.* discuss a privacy-aware search model called Nanuk [105] to search inside VMs.

6 Conclusion

The various types of virtualizable resources have been discussed in this paper. While this is in no way exhaustive, the topics covered should be enough for someone who is starting with virtualization-related research. Further looking at the papers cited here (and their references), one should be able to have a holistic view of the virtualization landscape. One can also look at the joint ACM/USENIX International Conference on Virtual Execution Environments (VEE), courses taught by seasoned researchers in the field[10] as well as books such as Smith and Nair's *Virtual Machines* [117].

References

[1] D. Abramson, J. Jackson, S. Muthrasanallur, G. Neiger, G. Regnier, R. Sankaran, I. Schoinas, R. Uhlig, B. Vembu, and J. Wiegert. Intel virtualization technology for directed I/O. *Intel Technology Journal*, 10(3):179–192, Aug. 2006.

[2] K. Adams and O. Agesen. A Comparison of Software and Hardware Techniques for x86 Virtualization. In *Proceedings of the 12th international conference on Architectural support for programming languages and operating systems*, ASPLOS-XII, pages 2–13, New York, NY, USA, 2006. ACM.

[3] G. A. Alvarez, J. Wilkes, E. Borowsky, S. Go, T. H. Romer, R. Becker-Szendy, R. Golding, A. Merchant, M. Spasojevic, and A. Veitch. Minerva: An automated resource provisioning tool for large-scale storage systems. *ACM Transactions on Computer Systems*, 19(4):483–518, Nov. 2001.

[4] G. Ammons, V. Bala, T. Mummert, D. Reimer, and X. Zhang. Virtual machine images as structured data: the mirage image library. In *Proceedings of the 3rd USENIX conference on Hot topics in cloud computing*, HotCloud'11, Berkeley, CA, USA, 2011. USENIX Association.

[5] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris. Resilient overlay networks. In *Proceedings of the 18th ACM Symposium on Oper-*

[10] See P. Dinda - *Resource Virtualization*, R. Figuereido - *Virtual Computers*

ating Systems Principles (SOSP '01), volume 35, pages 131–145, New York, NY, USA, Dec. 2001. ACM.

[6] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. A view of cloud computing. *Commun. ACM*, 53(4):50–58, Apr. 2010.

[7] V. Bala, E. Duesterwald, and S. Banerjia. Dynamo: a transparent dynamic optimization system. In *PLDI '00: Proceedings of the ACM SIGPLAN 2000 conference on Programming language design and implementation*, volume 35, pages 1–12, New York, NY, USA, May 2000. ACM.

[8] S. Banerjee, S. Lee, B. Bhattacharjee, and A. Srinivasan. Resilient Multicast Using Overlays. *Networking, IEEE/ACM Transactions on*, 14(2):237–248, 2006.

[9] R. A. Baratto, L. N. Kim, and J. Nieh. THINC: a virtual display architecture for thin-client computing. In *Proceedings of the twentieth ACM symposium on Operating systems principles*, volume 39 of *SOSP '05*, pages 277–290, New York, NY, USA, Dec. 2005. ACM.

[10] L. Baraz, T. Devor, O. Etzion, S. Goldenberg, A. Skaletsky, Y. Wang, and Y. Zemach. IA-32 Execution Layer: a two-phase dynamic translator designed to support IA-32 applications on Itanium®-based systems. In *Proceedings of the 36th annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 36, Washington, DC, USA, 2003. IEEE Computer Society.

[11] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the art of virtualization. In *Proceedings of the nineteenth ACM symposium on Operating systems principles*, SOSP '03, pages 164–177, New York, NY, USA, 2003. ACM.

[12] M. Beck, T. Moore, and J. S. Plank. An end-to-end approach to globally scalable network storage. In *SIGCOMM '02: Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*, volume 32, pages 339–346. ACM Press, Oct. 2002.

[13] F. Bellard. QEMU, a Fast and Portable Dynamic Translator. pages 41–46.

[14] S. Berger, R. Cáceres, K. A. Goldman, R. Perez, R. Sailer, and L. van Doorn. vTPM: virtualizing the trusted platform module. In *Proceedings of the 15th conference on USENIX Security Symposium - Volume 15*, Berkeley, CA, USA, 2006. USENIX Association.

[15] T. Boyd and P. Dasgupta. Process Migration: A Generalized Approach Using a Virtualizing Operating System. In *ICDCS '02: Proceedings of the 22 nd International Conference on Distributed Computing Systems (ICDCS'02)*, pages 385+, Washington, DC, USA, 2002. IEEE Computer Society.

[16] S. Bratus, M. E. Locasto, A. Ramaswamy, and S. W. Smith. VM-based security overkill: a lament for applied systems security research. In *Proceedings of the 2010 workshop on New security paradigms*, NSPW '10, pages 51–60, New York, NY, USA, 2010. ACM.

[17] T. C. Bressoud and F. B. Schneider. Hypervisor-based fault tolerance. In *SOSP '95: Proceedings of the fifteenth ACM symposium on Operating systems principles*, volume 29, pages 1–11, New York, NY, USA, Dec. 1995. ACM Press.

[18] E. Bugnion, S. Devine, K. Govil, and M. Rosenblum. Disco: Running Commodity Operating Systems on Scalable Multiprocessors. *ACM Transactions on Computer Systems*, 15(4):412–447, 1997.

[19] A. L. Cavilla, N. Tolia, M. Satyanarayanan, and E. de Lara. VMM-Independent Graphics Acceleration. In *Proceedings of the 3rd International Conference on Virtual Execution Environments (VEE'07)*, pages 33–43, San Diego, California, USA, 2007.

[20] P. M. Chen and B. D. Noble. When Virtual Is Better Than Real. In *HOTOS '01: Proceedings of the Eighth Workshop on Hot Topics in Operating Systems*, Washington, DC, USA, 2001. IEEE Computer Society.

[21] X. Chen, T. Garfinkel, C. E. Lewis, P. Subrahmanyam, C. A. Waldspurger, D. Boneh, J. Dwoskin, and D. R. K. Ports. Overshadow: a

virtualization-based approach to retrofitting protection in commodity operating systems. In *ASPLOS XIII: Proceedings of the 13th international conference on Architectural support for programming languages and operating systems*, pages 2–13, New York, NY, USA, 2008. ACM.

[22] L. Cherkasova and R. Gardner. Measuring CPU overhead for I/O processing in the Xen virtual machine monitor. In *Proceedings of the annual conference on USENIX Annual Technical Conference*, ATEC '05, page 24, Berkeley, CA, USA, 2005. USENIX Association.

[23] Y.-H. Chu, S. G. Rao, S. Seshan, and H. Zhang. A case for end system multicast. *IEEE Journal on Selected Areas in Communications*, 20(8):1456–1471, Oct. 2002.

[24] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield. Live migration of virtual machines. In *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation - Volume 2*, NSDI'05, pages 273–286, Berkeley, CA, USA, 2005. USENIX Association.

[25] J. M. Cobleigh, G. S. Avrunin, and L. A. Clarke. Breaking up is hard to do: An evaluation of automated assume-guarantee reasoning. *ACM Trans. Softw. Eng. Methodol.*, 17(2), May 2008.

[26] F. J. Corbató and V. A. Vyssotsky. Introduction and overview of the multics system. In *Proceedings of the November 30–December 1, 1965, fall joint computer conference, part I*, AFIPS '65 (Fall, part I), pages 185–196, New York, NY, USA, 1965. ACM.

[27] R. J. Creasy. The Origin of the VM/370 Time-Sharing System. *IBM Journal of Research and Development*, 25(5):483–490, 1981.

[28] B. Danev, R. J. Masti, G. O. Karame, and S. Capkun. Enabling secure VM-vTPM migration in private clouds. In *Proceedings of the 27th Annual Computer Security Applications Conference*, ACSAC '11, New York, NY, USA, 2011. ACM.

[29] J. C. Dehnert, B. K. Grant, J. P. Banning, R. Johnson, T. Kistler, E. Klaiber, and J. Mattson. The transmeta code morphing software: Using speculation, recovery, and adaptive retranslation to address real-life challenges. pages 15–24. IEEE Computer Society, 2003.

[30] E. W. Dijkstra. The structure of the THE-multiprogramming system. *Commun. ACM*, 11(5):341–346, May 1968.

[31] J. Dike. A user-mode port of the Linux kernel. In *In Proceedings of the 5th Annual Linux Showcase and Conference*, Oakland, California, Nov. 2001.

[32] W. B. Dobrusky and T. B. Steel. Universal computer-oriented language. *Commun. ACM*, 4(3), Mar. 1961.

[33] Y. Dong, S. Li, A. Mallick, J. Nakajima, K. Tian, X. Xu, F. Yang, and W. Yu. Extending Xen with Intel Virtualization Technology. *Intel Technology Journal*, 10(03):193–204, 2006.

[34] G. W. Dunlap, S. T. King, S. Cinar, M. A. Basrai, and P. M. Chen. ReVirt: enabling intrusion analysis through virtual-machine logging and replay. *SIGOPS Oper. Syst. Rev.*, 36(SI):211–224, Dec. 2002.

[35] P. Ferguson and G. Huston. What is a VPN?, 1998.

[36] N. Feske and C. Helmuth. Design of the Bastei OS Architecture. Technical report, Technische Universität Dresden, Dec. 2006.

[37] R. Figueiredo, P. A. Dinda, and J. Fortes. Guest Editors' Introduction: Resource Virtualization Renaissance. *Computer*, 38(5):28–31, May 2005.

[38] R. J. Figueiredo, P. A. Dinda, and J. A. B. Fortes. A Case For Grid Computing On Virtual Machines. In *ICDCS '03: Proceedings of the 23rd International Conference on Distributed Computing Systems*, Washington, DC, USA, 2003. IEEE Computer Society.

[39] K. Fraser, S. Hand, R. Neugebauer, I. Pratt, A. Warfield, and M. Williamson. Reconstructing I/O. Technical report, 2004.

[40] Ganguly, A., Agrawal, A., Boykin, P., Figueiredo, and R. WOW: Self-organizing Wide Area Overlay Networks of Virtual Workstations. *Journal of Grid Computing*, 5(2):151–172, June 2007.

[41] A. Ganguly, A. Agrawal, P. Boykin, and R. Figueiredo. Ip over p2p: enabling self-configuring virtual ip networks for grid computing. In

Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International, page 10 pp., april 2006.

[42] A. Ganguly, P. Boykin, D. Wolinsky, and R. Figueiredo. Improving peer connectivity in wide-area overlays of virtual workstations. *Cluster Computing*, 12(2):239–256, June 2009.

[43] T. Garfinkel, B. Pfaff, J. Chow, M. Rosenblum, and D. Boneh. Terra: a virtual machine-based platform for trusted computing. In *Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP '03)*, pages 193–206, New York, NY, USA, 2003. ACM Press.

[44] T. Garfinkel and M. Rosenblum. A Virtual Machine Introspection Based Architecture for Intrusion Detection. In *In Proc. Network and Distributed Systems Security Symposium*, pages 191–206, 2003.

[45] T. Garfinkel and M. Rosenblum. When virtual is harder than real: security challenges in virtual machine based computing environments. In *HOTOS'05: Proceedings of the 10th conference on Hot Topics in Operating Systems*, page 20, Berkeley, CA, USA, 2005. USENIX Association.

[46] B. Gleeson, A. Lin, J. Heinanen, G. Armitage, and A. Malis. A Framework for IP Based Virtual Private Networks, 2000.

[47] R. P. Goldberg. Architecture of virtual machines. In *Proceedings of the workshop on virtual computer systems*, pages 74–112, New York, NY, USA, 1973. ACM.

[48] R. P. Goldberg. Survey of Virtual Machine Research. *IEEE Computer Magazine*, pages 34–45, June 1974.

[49] A. Gupta and P. A. Dinda. Inferring the topology and traffic load of parallel programs running in a virtual machine environment. In *Proceedings of the 10th international conference on Job Scheduling Strategies for Parallel Processing*, JSSPP'04, Berlin, Heidelberg, 2005. Springer-Verlag.

[50] A. Gupta, M. Zangrilli, A. I. Sundararaj, A. I. Huang, P. A. Dinda, and B. Lowekamp. Free Network Measurement for Adaptive Virtualized Distributed Computing. In *Proceedings of the 20th International*

Parallel and Distributed Processing Symposium (IPDPS'06), Rhodes Island, Greece, 2006.

[51] S. Hand, T. Harris, E. Kotsovinos, and I. Pratt. Controlling the XenoServer Open Platform, 2003.

[52] H. Härtig, M. Hohmuth, N. Feske, C. Helmuth, A. Lackorzynski, F. Mehnert, and M. Peter. The Nizza Secure-System Architecture. In *In IEEE CollaborateCom 2005*, 2005.

[53] M. Hohmuth, M. Peter, H. Härtig, and J. S. Shapiro. Reducing TCB size by using untrusted components: small kernels versus virtual-machine monitors. In *EW 11: Proceedings of the 11th workshop on ACM SIGOPS European workshop*, pages 22+, New York, NY, USA, 2004. ACM.

[54] L. Huang, G. Peng, and T. C. Chiueh. Multi-dimensional storage virtualization. *SIGMETRICS Perform. Eval. Rev.*, 32(1):14–24, June 2004.

[55] W. Huang, J. Liu, M. Koop, B. Abali, and D. Panda. Nomad: migrating OS-bypass networks in virtual machines. In *VEE '07: Proceedings of the 3rd international conference on Virtual execution environments*, pages 158–168, New York, NY, USA, 2007. ACM Press.

[56] G. F. Italiano, R. Rastogi, and B. Yener. Restoration algorithms for virtual private networks in the hose model. In *Proceedings.Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, pages 131–139. IEEE, 2002.

[57] J. Jannotti, D. K. Gifford, K. L. Johnson, F. M. Kaashoek, and James. Overcast: Reliable Multicasting with an Overlay Network. pages 197–212.

[58] X. Jiang and D. Xu. Collapsar: a VM-based architecture for network attack detention center. In *SSYM'04: Proceedings of the 13th conference on USENIX Security Symposium*, page 2, Berkeley, CA, USA, 2004. USENIX Association.

[59] X. Jiang and D. Xu. VIOLIN: Virtual Internetworking on Overlay Infrastructure. In *Parallel and Distributed Processing and Applications*, pages 937–946. 2005.

[60] S. T. Jones, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau. Antfarm: tracking processes in a virtual machine environment. In *Proceedings of the annual conference on USENIX '06 Annual Technical Conference*, ATEC '06, Berkeley, CA, USA, 2006. USENIX Association.

[61] S. T. Jones, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau. Geiger: monitoring the buffer cache in a virtual machine environment. *SIGARCH Comput. Archit. News*, 34(5), Oct. 2006.

[62] S. T. Jones, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau. VMM-based hidden process detection and identification using Lycosid. In *VEE '08: Proceedings of the fourth ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*, pages 91–100, New York, NY, USA, 2008. ACM.

[63] A. Joshi, S. T. King, G. W. Dunlap, and P. M. Chen. Detecting past and present intrusions through vulnerability-specific predicates. In *SOSP '05: Proceedings of the twentieth ACM symposium on Operating systems principles*, volume 39, pages 91–104, New York, NY, USA, Dec. 2005. ACM Press.

[64] P.-H. Kamp and R. N. M. Watson. Jails: Confining the Omnipotent Root. In *Proceedings of the 2nd International System Administration and Networking Conference (SANE 2000)*, Maastricht, Netherlands, May 2000.

[65] S. T. King, P. M. Chen, Y. M. Wang, C. Verbowski, H. J. Wang, and J. R. Lorch. SubVirt: Implementing malware with virtual machines. In *SP '06: Proceedings of the 2006 IEEE Symposium on Security and Privacy*, pages 314–327, Washington, DC, USA, 2006. IEEE Computer Society.

[66] S. T. King, G. W. Dunlap, and P. M. Chen. Debugging Operating Systems with Time-Traveling Virtual Machines. pages 1–15.

[67] S. T. King, G. W. Dunlap, and P. M. Chen. Operating system support for virtual machines. In *Proceedings of the annual conference on USENIX Annual Technical Conference*, page 6, Berkeley, CA, USA, 2003. USENIX Association.

[68] G. Klein, K. Elphinstone, G. Heiser, J. Andronick, D. Cock, P. Derrin, D. Elkaduwe, K. Engelhardt, R. Kolanski, M. Norrish, T. Sewell, H. Tuch, and S. Winwood. seL4: formal verification of an OS kernel. In *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*, SOSP '09, pages 207–220, New York, NY, USA, 2009. ACM.

[69] M. Kozuch, Bressoud, M. Kozuch, M. Satyanarayanan, M. Satyanarayanan, T. Bressoud, T. Bressoud, Y. Ke, and Y. Ke. Efficient State Transfer for Internet Suspend/Resume. Technical report, Intel Research Laboratory at Pittburgh, 2002.

[70] S. Kumar and K. Schwan. Netchannel: a VMM-level mechanism for continuous, transparentdevice access during VM migration. In *Proceedings of the fourth ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*, VEE '08, New York, NY, USA, 2008. ACM.

[71] A. Lai and J. Nieh. Limits of wide-area thin-client computing. In *Proceedings of the 2002 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, SIGMETRICS '02, pages 228–239, New York, NY, USA, 2002. ACM.

[72] J. Lange, K. Pedretti, T. Hudson, P. Dinda, Z. Cui, L. Xia, P. Bridges, A. Gocke, S. Jaconette, M. Levenhagen, and R. Brightwell. Palacios and Kitten: New High Performance Operating Systems for Scalable Virtualized and Native Supercomputing. In *IPDPS '10: Proceedings of the 24th IEEE International Parallel and Distributed Processing Symposium*, Washington, DC, USA, Apr. 2010. IEEE Computer Society.

[73] J. R. Lange and P. A. Dinda. Transparent network services via a virtual traffic layer for virtual machines. In *HPDC '07: Proceedings of the 16th international symposium on High performance distributed computing*, pages 23–32, New York, NY, USA, 2007. ACM.

[74] J. R. Lange, P. A. Dinda, and S. Rossoff. Experiences with client-based speculative remote display. In *USENIX 2008 Annual Technical Conference on Annual Technical Conference*, ATC'08, Berkeley, CA, USA, 2008. USENIX Association.

[75] J. R. Lange, A. I. Sundararaj, and P. A. Dinda. Automatic dynamic run-time optical network reservations. In *Proceedings of the High Performance Distributed Computing, 2005. HPDC-14. Proceedings. 14th IEEE International Symposium*, HPDC '05, Washington, DC, USA, 2005. IEEE Computer Society.

[76] A. Lenk, M. Klems, J. Nimis, S. Tai, and T. Sandholm. What's inside the Cloud? An architectural map of the Cloud landscape. In *Proceedings of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing*, volume 0 of *CLOUD '09*, pages 23–31, Washington, DC, USA, May 2009. IEEE Computer Society.

[77] J. Levasseur, V. Uhlig, J. Stoess, and S. Götz. Unmodified Device Driver Reuse and Improved System Dependability via Virtual Machines. pages 17–30.

[78] D. Lie and L. Litty. Using hypervisors to secure commodity operating systems. In *Proceedings of the fifth ACM workshop on Scalable trusted computing*, STC '10, New York, NY, USA, 2010. ACM.

[79] B. Lin and P. A. Dinda. VSched: Mixing Batch And Interactive Virtual Machines Using Periodic Real-time Scheduling. In *SC '05: Proceedings of the 2005 ACM/IEEE conference on Supercomputing*, Washington, DC, USA, 2005. IEEE Computer Society.

[80] T. Lindholm and F. Yellin. *Java(TM) Virtual Machine Specification, The (2nd Edition)*. Prentice Hall PTR, 2 edition, Apr. 1999.

[81] J. Liu, W. Huang, B. Abali, and D. K. Panda. High performance VMM-bypass I/O in virtual machines. In *ATEC '06: Proceedings of the annual conference on USENIX '06 Annual Technical Conference*, page 3, Berkeley, CA, USA, 2006. USENIX Association.

[82] C. R. Lumb, A. Merchant, and G. A. Alvarez. Façade: Virtual Storage Devices with Performance Guarantees. In *FAST '03: Proceedings of the 2nd USENIX Conference on File and Storage Technologies*, pages 131–144, Berkeley, CA, USA, 2003. USENIX Association.

[83] S. E. Madnick and J. J. Donovan. Application and analysis of the virtual machine approach to information system security and isolation.

In *Proceedings of the workshop on virtual computer systems*, pages 210–224, 1973.

[84] P. S. Magnusson. The Virtual Test Lab. *IEEE Computer*, 38(5):95–97, 2005.

[85] J. M. McCune, B. Parno, A. Perrig, M. K. Reiter, and A. Seshadri. How low can you go?: recommendations for hardware-supported minimal TCB code execution. *SIGOPS Oper. Syst. Rev.*, 42(2), Mar. 2008.

[86] W. W. McMillan. The soul of the virtual machine. *Spectrum, IEEE*, 48(7):44–59, July 2011.

[87] E. Meijer and J. Gough. Technical Overview of the Common Language Runtime, 2000.

[88] A. Menon, A. L. Cox, and W. Zwaenepoel. Optimizing network virtualization in Xen. In *ATEC '06: Proceedings of the annual conference on USENIX '06 Annual Technical Conference*, page 2, Berkeley, CA, USA, 2006. USENIX Association.

[89] A. Menon, J. R. Santos, Y. Turner, G. J. Janakiraman, and W. Zwaenepoel. Diagnosing performance overheads in the xen virtual machine environment. In *Proceedings of the 1st ACM/USENIX international conference on Virtual execution environments*, VEE '05, pages 13–23, New York, NY, USA, 2005. ACM.

[90] J. Moore, J. Chase, P. Ranganathan, and R. Sharma. Making scheduling "cool": temperature-aware workload placement in data centers. In *ATEC'05: Proceedings of the USENIX Annual Technical Conference 2005 on USENIX Annual Technical Conference*, page 5, Berkeley, CA, USA, 2005. USENIX Association.

[91] A. Muthitacharoen, B. Chen, and D. Mazières. A low-bandwidth network file system. In *Proceedings of the eighteenth ACM symposium on Operating systems principles*, SOSP '01, pages 174–187, New York, NY, USA, 2001. ACM.

[92] R. Nathuji and K. Schwan. VirtualPower: coordinated power management in virtualized enterprise systems. In *Proceedings of twenty-first*

ACM SIGOPS symposium on Operating systems principles, volume 41 of *SOSP '07*, pages 265–278, New York, NY, USA, Oct. 2007. ACM.

[93] R. Nathuji, K. Schwan, A. Somani, and Y. Joshi. VPM tokens: virtual machine-aware power budgeting in datacenters. *Cluster Computing*.

[94] G. Neiger, A. Santoni, F. Leung, D. Rodgers, and R. Uhlig. Intel Virtualization Technology: Hardware support for efficient processor virtualization. *Intel Technology Journal*, 10(3):167–177, Aug. 2006.

[95] M. Nelson, B. H. Lim, and G. Hutchins. Fast transparent migration for virtual machines. In *ATEC '05: Proceedings of the annual conference on USENIX Annual Technical Conference*, page 25, Berkeley, CA, USA, 2005. USENIX Association.

[96] S. Osman, D. Subhraveti, G. Su, and J. Nieh. The Design and Implementation of Zap: A System for Migrating Computing Environments.

[97] B. Pfaff, T. Garfinkel, and M. Rosenblum. Virtualization Aware File Systems: Getting Beyond the Limitations of Virtual Disks. pages 353–366.

[98] G. J. Popek and R. P. Goldberg. Formal requirements for virtualizable third generation architectures. In *SOSP '73: Proceedings of the fourth ACM symposium on Operating system principles*, volume 7, New York, NY, USA, Oct. 1973. ACM Press.

[99] G. J. Popek and C. S. Kline. A verifiable protection system. In *Proceedings of the international conference on Reliable software*, pages 294–304, New York, NY, USA, 1975. ACM Press.

[100] D. Price and A. Tucker. Solaris Zones: Operating System Support for Consolidating Commercial Workloads. pages 241–254.

[101] B. Prosnitz. Blackbox No More: Reconstruction of Internal Virtual Machine State. Technical report, 2007.

[102] H. Raj and K. Schwan. High performance and scalable I/O virtualization via self-virtualized devices. In *Proceedings of the 16th international symposium on High performance distributed computing*, HPDC '07, New York, NY, USA, 2007. ACM.

[103] V. T. Ravi, M. Becchi, G. Agrawal, and S. Chakradhar. Supporting gpu sharing in cloud environments with a transparent runtime consolidation framework. In *Proceedings of the 20th international symposium on High performance distributed computing*, HPDC '11, pages 217–228, New York, NY, USA, 2011. ACM.

[104] T. Richardson, Q. Stafford-Fraser, K. R. Wood, and A. Hopper. Virtual network computing. *IEEE Internet Computing*, 2(1):33–38, Jan. 1998.

[105] W. Richter, G. Ammons, J. Harkes, A. Goode, N. Bila, E. De Lara, V. Bala, and M. Satyanarayanan. Privacy-sensitive VM retrospection. In *Proceedings of the 3rd USENIX conference on Hot topics in cloud computing*, HotCloud'11, Berkeley, CA, USA, 2011. USENIX Association.

[106] T. Roscoe, K. Elphinstone, and G. Heiser. Hype and virtue. In *Proceedings of the 11th USENIX workshop on Hot topics in operating systems*, pages 1–6, Berkeley, CA, USA, 2007. USENIX Association.

[107] M. Rosenblum and T. Garfinkel. Virtual machine monitors: current technology and future trends. *Computer*, 38(5):39–47, May 2005.

[108] P. Ruth, X. Jiang, D. Xu, and S. Goasguen. Virtual distributed environments in a shared infrastructure. *Computer*, 38(5):63–69, 2005.

[109] R. Sailer, E. Valdez, T. Jaeger, R. Perez, L. V. Doorn, J. L. Griffin, S. Berger, R. Sailer, E. Valdez, T. Jaeger, R. Perez, L. Doorn, J. Linwood, and G. S. Berger. sHype: Secure Hypervisor Approach to Trusted Virtualized Systems. In *IBM Research Report RC23511*, 2005.

[110] J. H. Saltzer, D. P. Reed, and D. D. Clark. End-to-end arguments in system design. *ACM Trans. Comput. Syst.*, 2(4):277–288, Nov. 1984.

[111] C. Sapuntzakis, D. Brumley, R. Chandra, N. Zeldovich, J. Chow, M. S. Lam, and M. Rosenblum. Virtual Appliances for Deploying and Maintaining Software. In *LISA '03: Proceedings of the 17th USENIX conference on System administration*, pages 181–194, Berkeley, CA, USA, 2003. USENIX Association.

[112] C. P. Sapuntzakis, R. Chandra, B. Pfaff, J. Chow, M. S. Lam, and M. Rosenblum. Optimizing the migration of virtual computers. In *In*

Proceedings of the 5th Symposium on Operating Systems Design and Implementation, pages 377–390, 2002.

[113] M. Satyanarayanan, M. A. Kozuch, C. J. Helfrich, and D. R. O'Hallaron. Towards seamless mobility on pervasive hardware. *Pervasive Mob. Comput.*, 1(2), July 2005.

[114] B. K. Schmidt, M. S. Lam, and J. D. Northcutt. The interactive performance of SLIM: a stateless, thin-client architecture. In *Proceedings of the seventeenth ACM symposium on Operating systems principles*, SOSP '99, New York, NY, USA, 1999. ACM.

[115] A. Seshadri, M. Luk, N. Qu, and A. Perrig. SecVisor: a tiny hypervisor to provide lifetime kernel code integrity for commodity OSes. *SIGOPS Oper. Syst. Rev.*, 41:335–350, Oct. 2007.

[116] R. L. Sites, A. Chernoff, M. B. Kirk, M. P. Marks, and S. G. Robinson. Binary translation. *Commun. ACM*, 36(2):69–81, Feb. 1993.

[117] J. Smith and R. Nair. *Virtual Machines: Versatile Platforms for Systems and Processes*. Morgan Kaufmann, June 2005.

[118] J. E. Smith and R. Nair. The architecture of virtual machines. *Computer*, 38(5):32–38, May 2005.

[119] F. G. Soltis. *Inside the as/400*. Twenty Ninth Street Press, 1996.

[120] D. Spinellis. Another Level of Indirection. In A. Oram and G. Wilson, editors, *Beautiful Code: Leading Programmers Explain How They Think*, chapter 17. O'Reilly and Associates, Sebastopol, CA, 2007.

[121] U. Steinberg and B. Kauer. NOVA: a microhypervisor-based secure virtualization architecture. In C. Morin and G. Muller, editors, *EuroSys '10: Proceedings of the 5th European conference on Computer systems*, EuroSys '10, pages 209–222, New York, NY, USA, Apr. 2010. ACM.

[122] L. Subramanian, I. Stoica, H. Balakrishnan, and R. Katz. OverQoS: An Overlay based Architecture for Enhancing Internet QoS, 2004.

[123] J. Sugerman, G. Venkitachalam, and B.-H. Lim. Virtualizing I/O Devices on VMware Workstation's Hosted Virtual Machine Monitor. In

Proceedings of the General Track: 2002 USENIX Annual Technical Conference, Berkeley, CA, USA, 2001. USENIX Association.

[124] A. Sundararaj, A. Gupta, and P. Dinda. Increasing application performance in virtual environments through run-time inference and adaptation, 2005.

[125] A. I. Sundararaj and P. A. Dinda. Towards Virtual Networks for Virtual Machine Grid Computing. In *In Proceedings of the 3rd USENIX Virtual Machine Research And Technology Symposium (VM*, pages 177–190, 2004.

[126] A. S. Tanenbaum. *Structured Computer Organization (5th Edition)*. Prentice Hall, 5 edition, June 2005.

[127] W. H. Tetzlaff. State sampling of interactive VM/370 users. *IBM Syst. J.*, 18(1), Mar. 1979.

[128] E. Traut. Building the virtual pc. *BYTE*, 22(11):51–52, Nov. 1997.

[129] M. Tsugawa and J. A. B. Fortes. A virtual network (ViNe) architecture for grid computing. *Parallel and Distributed Processing Symposium, International*, 0:123+, 2006.

[130] R. Uhlig, G. Neiger, D. Rodgers, A. L. Santoni, F. C. M. Martins, A. V. Anderson, S. M. Bennett, A. Kagi, F. H. Leung, and L. Smith. Intel virtualization technology. *Computer*, 38(5):48–56, May 2005.

[131] M. Vrable, J. Ma, J. Chen, D. Moore, E. Vandekieft, A. C. Snoeren, G. M. Voelker, and S. Savage. Scalability, fidelity, and containment in the potemkin virtual honeyfarm. *SIGOPS Oper. Syst. Rev.*, 39(5):148–162, 2005.

[132] C. A. Waldspurger, T. Hogg, B. A. Huberman, J. O. Kephart, and W. S. Stornetta. Spawn: A Distributed Computational Economy. *IEEE Trans. Softw. Eng.*, 18(2):103–117, Feb. 1992.

[133] A. Whitaker, R. Cox, M. Shaw, and S. Gribble. Constructing services with interposable virtual hardware, 2004.

[134] A. Whitaker, R. S. Cox, M. Shaw, and S. D. Gribble. Rethinking the design of virtual machine monitors. *Computer*, 38(5):57–62, 2005.

[135] A. Whitaker, M. Shaw, and S. D. Gribble. Scale and performance in the Denali isolation kernel. *SIGOPS Oper. Syst. Rev.*, 36(SI):195–209, Dec. 2002.

[136] P. Willmann, J. Shafer, D. Carr, A. Menon, S. Rixner, A. L. Cox, and W. Zwaenepoel. Concurrent Direct Network Access for Virtual Machine Monitors. *High Performance Computer Architecture, 2007. HPCA 2007. IEEE 13th International Symposium on*, pages 306–317, Feb. 2007.

[137] T. Wood, P. Shenoy, and Arun. Black-box and Gray-box Strategies for Virtual Machine Migration. pages 229–242.

[138] J. Xu and J. A. B. Fortes. Towards autonomic virtual applications in the in-vigo system. In *Proceedings of the Second International Conference on Automatic Computing*, ICAC '05, pages 15–26, Washington, DC, USA, 2005. IEEE Computer Society.

[139] Y. Yu, F. Guo, S. Nanda, L.-c. Lam, and T.-c. Chiueh. A featherweight virtual machine for windows applications. In *Proceedings of the 2nd international conference on Virtual execution environments*, VEE '06, New York, NY, USA, 2006. ACM.

[140] X. Zhao, K. Borders, and A. Prakash. Virtual Machine Security Systems.

[141] W. Zhou, P. Ning, X. Zhang, G. Ammons, R. Wang, and V. Bala. Always up-to-date: scalable offline patching of VM images in a compute cloud. In *Proceedings of the 26th Annual Computer Security Applications Conference*, ACSAC '10, New York, NY, USA, 2010. ACM.