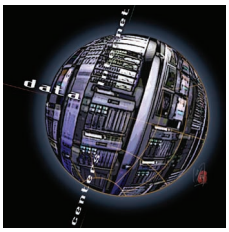
SoftUDC: A Software-Based Data Center for Utility Computing



Utility computing aims to aggregate server, network, and storage systems into a single, centrally managed pool of resources. SoftUDC, a virtual machine monitor, lets applications and administrative domains share physical resources while maintaining full functional isolation.

Mahesh Kallahalla
Mustafa Uysal
Ram Swaminathan
David E. Lowell
Mike Wray
Tom Christian
Nigel Edwards
Chris I. Dalton
Frederic Gittler
HP Labs

Typical enterprise IT environments consist of numerous independent and distributed servers, networks, and storage devices. To improve efficiency, customers increasingly seek to bring these disparate systems under one umbrella of control. *Utility computing* facilitates this effort by aggregating such systems into a single, centrally managed pool of resources.

To adequately address user expectations, a utility computing environment should provide several core features.

- *Unified control.* IT administrators should have a single point of control from which to manage their infrastructure. This central console simplifies administration and enables automation of many tasks.
- *Freedom from physical configuration.* Administrators should be able to deploy applications and change system configurations without physically rewiring their infrastructure. Further, they should not have to overhaul or redesign the infrastructure to use it.
- *Resource sharing.* To improve hardware utilization, multiple applications should share and oversubscribe physical resources.
- *Resource isolation.* Users should have complete administrative control and the illusion that they alone own their resources. Despite the underlying sharing of resources, one administrative domain's network and disk traffic should be invisible to other domains.

To provide these features, we propose a software-based utility data center that virtualizes server, network, and storage resources. SoftUDC relies on a *virtual machine monitor* that runs on each server. The VMM supports running multiple virtual machines, abstracts each virtual machine's view of its storage and networks, and binds virtual machines across nodes into isolated *virtual farms*.

The SoftUDC control system spans all the VMMs, providing a unified console for data center resources and functions. From this console, an administrator can deploy services and modify virtual farms without reconfiguring the physical infrastructure. SoftUDC also automates many common administrative tasks such as performing routine maintenance, deploying new applications, and dynamic load balancing.

SoftUDC adds a *gatekeeper* to each server's VMM that mediates all I/O and network traffic the virtual servers generate. This component enforces access control and provides communications and

Published by the IEEE Computer Society
0018-9162/04/$20.00 © 2004 IEEE

I/O security. The gatekeeper could be implemented in hardware that virtualizes the network and storage, either in the VMM or the underlying host OS, if the VMM is running on one.

Our prototype uses the Xen VMM[1] and implements gatekeeper components in the VMM as well as in a special management virtual machine running on top of it. Figure 1 presents an overview of the SoftUDC software stack on a server, with a distinguished virtual machine hosting the management OS.

SERVER VIRTUALIZATION

SoftUDC presents an abstraction of a virtual machine with its own OS image and application suite. Each virtual machine exists in a separate protection domain and is thus isolated from OS crashes, user errors, or transient failures occurring on other virtual machines. This provides a level of isolation comparable to that found in separate physical servers.

Each SoftUDC physical server supports an arbitrary number of virtual machines, subject to availability of sufficient physical resources to provide acceptable performance. Each physical server also runs a management OS in a separate virtual machine that participates in the management and operation of the server, VMM, storage, and network infrastructure. All virtual machines, including those that run the management OS, run as unprivileged tasks and cannot directly execute privileged instructions.

Management API

The management OS uses a distinct management API, accessible directly or through a secure connection, to carry out tasks that require privileged operations. In addition to managing resource allocation, the API provides access to services that create, quiesce, and destroy virtual machines. Managed resources include processors, memory, network bandwidth, and I/O bandwidth. Resources can be oversubscribed: The sum of all resources allocated to all virtual machines can exceed the total resources available on the system, enabling efficient support of applications with resource requirements that are complementary over time.

The VMM provides resource measurement and monitoring functions to enable performance monitoring of virtual machines. It also offers resource control functions to enable allocation of resources to virtual machines. Resource allocation typically is managed to optimize cost, performance, quality of service, and power consumption. Analytical models map user-level or business-performance metrics to physical resource management actions so that resource management decisions can produce predictable changes in visible performance characteristics.

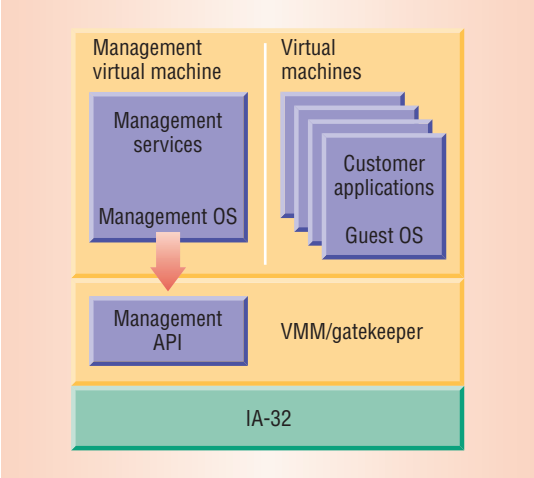


Figure 1. SoftUDC implementation. Each physical server supports an arbitrary number of virtual machines and runs a management OS in a separate virtual machine that participates in the management and operation of the server, network, and storage infrastructure.

Virtual machine migration

SoftUDC can migrate virtual machines from one physical server to another, treating physical servers as a single, unified pool.[2] Virtual machine migration can acquire resources available on a different physical server or increase resource utilization by coalescing multiple virtual machines making modest resource demands on one physical server. It can also free up a particular physical server for software or hardware maintenance or shut it down to conserve energy.

Data encryption

SoftUDC assumes that those who have physical access to a machine, such as the data center owner, are trusted. To relax this constraint, Trusted Computing Group (www.trustedcomputinggroup.org/home) technology can be used to provide a hardware root of trust: the Trusted Platform Module inside the machine. A TPM makes it possible to establish a server's integrity without owner interference. Users can verify that the server is running a particular BIOS and SoftUDC platform—including the VMM, underlying host OS, and gatekeeper—and that they have not been modified. Thus, users can trust the SoftUDC platform to enforce its advertised security properties. They can protect their data by encryption, independently of the actions and policies of the physical server's owner, and the owner cannot decrypt the data.

NETWORK VIRTUALIZATION

Network virtualization gives users the impression of having their own virtual private local area network (LAN), known as a VNET, within which they can use any media access control (MAC) or IP address. VNETs must provide security comparable

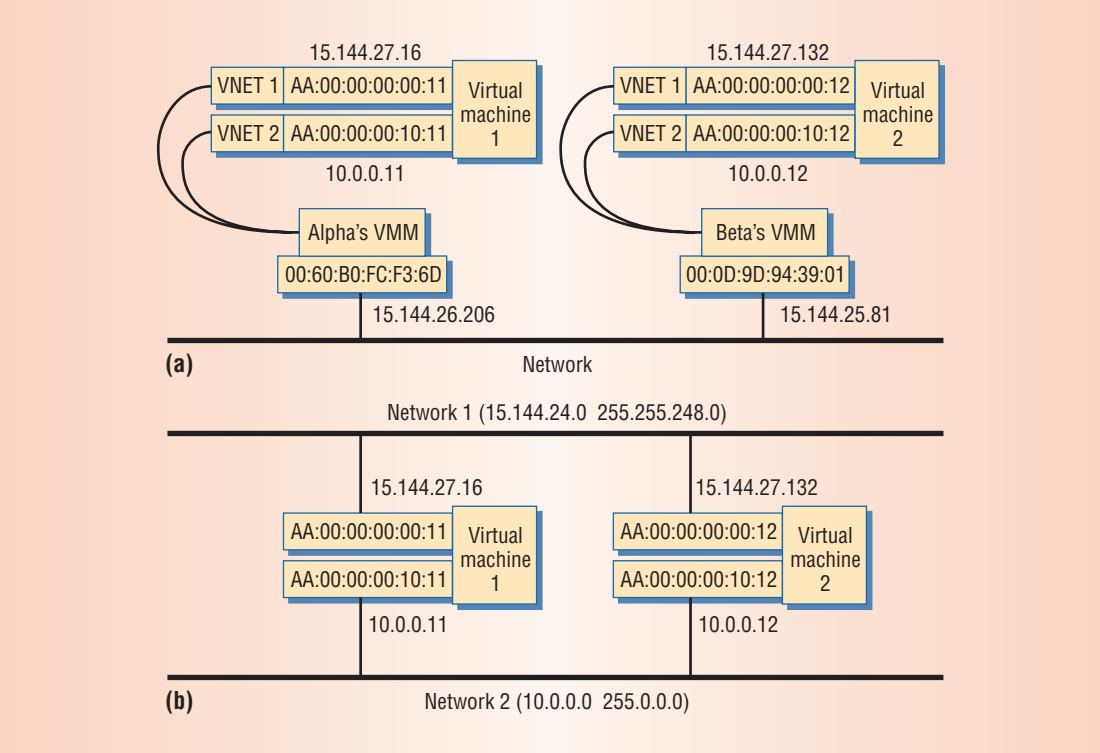


Figure 2. SoftUDC network virtualization. (a) The physical network configuration and (b) the virtual network view.

to a hardware VLAN. They are decoupled from the underlying network topology, and they maintain network connectivity during virtual machine migration.

Virtual network interface

Virtual machines access networking via a *virtual interface network* (VIF), which mimics an Ethernet device. The VMM forwards outbound network packets to its physical network interface and dispatches incoming network packets to appropriate VIFs. If this was all the VMM did, network packets from virtual machines would use the physical network. Their VIFs would need suitable MAC and IP addresses, and other machines on the physical network could observe virtual machine packets in transit.

Network traffic from virtual machines is distributed to virtual interfaces via Ethernet bridging in a special virtual machine containing a kernel module. The kernel module uses the EtherIP protocol to encapsulate outbound Ethernet traffic destined for a VNET in an IP packet and forwards the packet to the network. The kernel module decapsulates inbound EtherIP traffic to produce an Ethernet frame, which it delivers to VIFs on its VNET. An EtherIP header field stores a packet's VNET identification for transport across the network. If a VIF requires direct access to the physical network, the kernel module delivers its traffic to the network without encapsulation.

We encapsulate Ethernet frames rather than IP traffic primarily because doing so allows virtual machines to use any network protocol transparently. In addition, handling the Ethernet frame is simpler than extracting an IP packet.

The kernel module must direct encapsulated VNET traffic to a suitable IP address. This care-of address is based on the Ethernet frame and VNET destination MAC address. If the MAC is a broadcast or multicast address, the care-of address is the local VNET multicast address. If the MAC is a unicast address, the care-of address is the real IP address of the machine hosting the addressed VIF.

We designed the Virtual Address Resolution Protocol (VARP) to use in discovering VIF care-of addresses, which change during virtual machine migration. Broadcasting a VARP reply for all VIFs maintains network connectivity by updating the VARP caches of any systems communicating with the virtual machine.

Network isolation

SoftUDC uses VNETs to isolate networks. The VMM encapsulates a packet and sends it to another VMM or a virtual router on the same VNET. The receiving VMM unwraps the packet and delivers it to the target. In the case of a virtual machine, the target can consume the packet, or, in the case of a virtual router or dual-homed virtual machine, it can forward the packet. Figure 2 illustrates how SoftUDC virtualizes the physical network to provide a different view to machines in the virtual farm.

Entities that share the same physical network but are not part of the same virtual farm should not be able to read the data in a packet or send packets to entities in that virtual farm. The IPsec Encapsulated Security Payload protocol encapsulates VNET

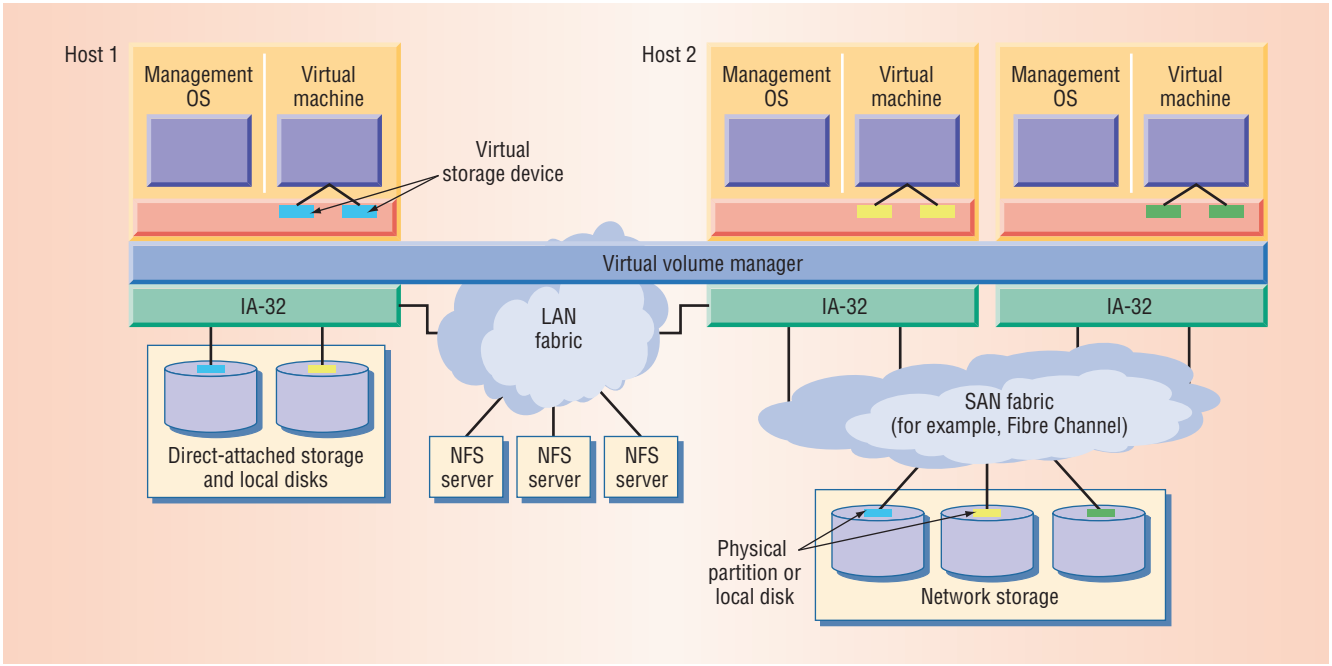


Figure 3. SoftUDC storage virtualization. The virtual volume manager abstracts a virtual storage device to the OS running on the servers from the shared pool of storage devices.

EtherIP packets to provide message authentication and ensure confidentiality. IPsec encryption ensures that only the intended target can read the data. IPsec authentication ensures that only entities within the virtual farm can send data.

Isolating network traffic from different farms also requires preventing one farm from using all available bandwidth on the physical network. We use Xen's scheduling and I/O throttling facilities to limit the bandwidth available to a virtual machine.

Firewall

SoftUDC provides a virtual firewall that the administrator can incorporate into a virtual farm. With this ordinary Linux-based firewall that runs in a separate multihomed virtual machine, designers can configure demilitarized zones and exercise control of the network traffic flow into, out of, and through their virtual network.

SoftUDC network virtualization facilitates the creation of multitier virtual networks. Two virtual servers can reside on the same physical machine and yet be in completely different networks. This makes it possible to isolate sensitive virtual machines in separate networks behind virtual firewalls.

Wide-area VNETs

Because VNET transport and VARP both use multicast, VNET packets normally are restricted to the LAN segment, and VARP cannot discover VIFs hosted farther away. In a network supporting multicast routing, routing the VNET multicast address to the relevant LANs can remedy this problem, but this solution often does not support multicast routing.

SoftUDC therefore uses a special-purpose daemon, *vnetd,* to support wide-area VNETs. One vnetd runs on each LAN segment hosting VIFs, and the vnetds are all connected to one another. Each vnetd forwards local VNET multicasts to its peers and resends forwarded multicasts locally. These daemons also forward VARP requests and replies.

Vnetds are transparent to VMMs. Encapsulated VNET traffic normally goes directly to the hosting machine rather than through the vnetds because VARP forwarding ensures that VMMs discover the hosting machine's address even if the VIF is remote. With these techniques, SoftUDC can host wide-area VNETs without requiring changes to the network infrastructure or using special hardware.

STORAGE VIRTUALIZATION

The goal of storage virtualization is to provide an abstraction of a storage pool consisting of all available storage devices in a data center. This abstract storage pool has two key properties.

- *Device transparency.* Because the storage devices incorporated in the storage pool are transparent to the clients, virtualization can incorporate file-based and block-based storage devices into the same storage pool.
- *Location transparency.* The location of data in the storage pool is invisible to virtual machines and applications. The data can be located in network-attached storage (including block and file storage), direct-attached storage, or local disks in the physical servers.

SoftUDC's *virtual volume manager* implements server-based storage virtualization that enables sharing of storage devices in the data center. As Figure 3 shows, the virtual volume manager runs in the gatekeeper and provides an abstraction of a

The virtual volume manager relies primarily on data migration to provide location transparency.

virtual storage device to the OS running on the servers from the shared pool of storage devices. It controls the mapping of VSDs onto the physical storage devices that are part of this pool. The mapping is attribute-based: Attributes such as required protection level (used to select RAID storage parameters) and desired performance (used to configure the striping parameters) determine the appropriate physical storage devices for a given VSD.

Traditionally, server-based storage virtualization only aggregates the storage area network (SAN) resources to which a server is attached. In contrast, SoftUDC's virtual volume manager can use any storage device in the data center, including storage directly attached to other servers. The virtual volume manager provides the necessary routing and redirection capabilities and uses both the SAN and LAN fabrics to carry I/O traffic.

SoftUDC uses a large pool of storage devices for striping, or increasing parallelism, and it can use both the LAN fabric and I/O adapters to increase a server's available I/O bandwidth. SoftUDC can transparently replicate VSD contents across multiple, possibly remote, storage devices to enhance storage system resilience.

Data migration

The virtual volume manager relies primarily on data migration to provide location transparency. When a virtual machine migrates to other nodes, the virtual volume manager migrates the VSDs along with it. In this case, only the VSD's access points migrate—no physical data moves. This form of migration can also pass large amounts of data across virtual machines by changing the VSD mappings.

In addition to migrating VSD access points, the virtual volume manager can move data transparently between physical devices. It can move disk data to retire obsolete storage devices, balance disk access load, and handle changes in VSD attributes. The virtual volume manager can continue to access VSD data while it is moving; it uses I/O request throttling to limit the effect of data movement on application performance.[3,4]

Performance isolation

Using VSD gives the illusion of private storage. However, the virtual volume manager uses shared physical storage devices to implement the VSD abstraction. While multiplexing workloads onto a shared infrastructure can improve resource utilization, providing performance isolation for individual workloads is difficult in practice. SoftUDC seeks to solve this problem by

- carefully providing sufficient resources for the shared workloads,[5] and
- using I/O request throttling to provide isolation against overloads and transient changes.[3]

Because virtualized devices consume additional bandwidth from the LAN fabric and CPU cycles to execute remote I/Os, the performance isolation mechanism also must take into account the fabric topology and transient changes in the hardware environment. It is an open question whether a unified mechanism for performance isolation can be developed that satisfies all these conditions.

Storage isolation

Separating VSDs from one another—storage isolation—requires enforcement of VSD access controls. This does not mean that a VSD is exclusively available to only one virtual machine; virtual machines can share VSDs as the infrastructure design requires. Because the data in a VSD travels to and from a virtual machine via the network, enforcement is by authenticating the two virtual volume manager servers at each end point. In addition, transparent encryption ensures the confidentiality of data being stored in a VSD.

SoftUDC uses IPsec to enforce this same requirement for network isolation. The current implementation assumes that an authenticated virtual volume manager server is trusted to preserve the integrity and confidentiality of the data it is storing. If this is not the case, clients must encrypt the data before it is passed into the virtual volume manager.

In addition to providing virtual block devices, the virtual volume manager can virtualize file storage on network file system (NFS) servers into a single name space using an NFS proxy that redirects NFS accesses from guest operating systems to appropriate back-end NFS servers.

AUTOMATED MANAGEMENT

The predominant cost of operating data centers is the personnel required to administer them.[6] SoftUDC greatly reduces such expenses by leveraging the common resource virtualization layer to automate most forms of data center management operations. These operations include routine maintenance such as repairing failed components, upgrading system software, reallocating and rede-

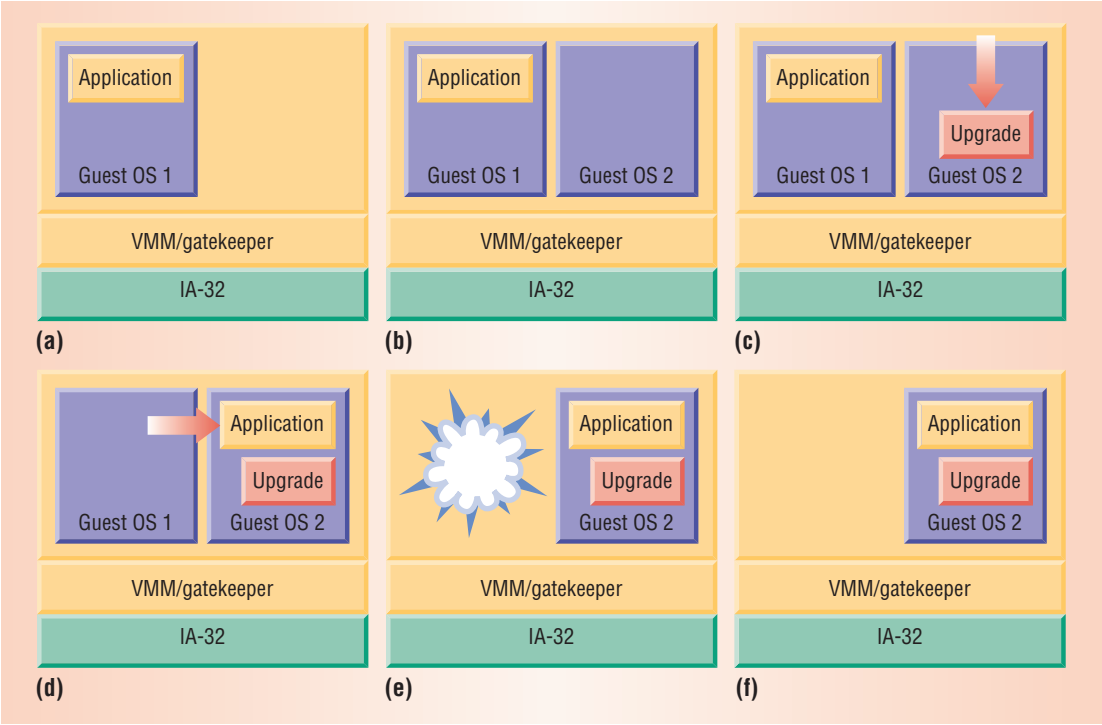


Figure 4. Online OS maintenance on a single node. When (a) the OS needs maintenance, SoftUDC (b) starts a second OS, (c) performs maintenance on the second OS instance, (d) migrates applications to the upgraded OS, (e) shuts down the first OS instance, and (f) completes the maintenance.

ploying resources to accommodate changing user needs or workloads, and incorporating and reconfiguring resources to accommodate growth.

Online maintenance

SoftUDC eliminates most planned downtime by performing maintenance on live systems without stopping applications. It also reduces the time required for administration by automating most forms of maintenance and by parallelizing software maintenance over many nodes at once. SoftUDC supports automatic online upgrading, patching, or reconfiguration of system software including guest operating systems, management operating systems, VMMs, and firmware. It also performs online hardware addition, replacement, removal, and reconfiguration.

Maintenance primitives. SoftUDC's support for online maintenance is constructed from several virtualization primitives including the ability to run multiple virtual machines on a hardware node, support for migrating a running virtual machine from node to node, and support for migrating running applications between virtual machines. SoftUDC's ability to adjust network and storage virtualization also accommodates on-the-fly migration.

OS maintenance. Figure 4 depicts an online OS maintenance operation on a single node. First, SoftUDC uses a copy-on-write clone of the first operating system's boot image to initiate a second OS instance on the node.

As the original OS and application run, SoftUDC brings the second OS to the upgraded state by installing an upgrade to the OS, applying a patch, setting up a new version of the application, or reconfiguring other software. It can reboot the second OS instance as necessary.

After updating the second OS instance, SoftUDC uses one of several possible techniques to migrate the applications from the first OS instance. For example, it can use either a transparent process migration system, such as Zap,[7] or cluster-style failover.[8] For applications like Web servers, SoftUDC can simply redirect application request traffic to an instance of the application already running in the second instance. It adjusts the underlying storage and network virtualization to ensure that the application sees no change to its network or application storage as a result of migration. When the application is safely running in the upgraded environment, SoftUDC shuts down the original OS and virtual machine.

Because SoftUDC's OS maintenance mechanism is compatible with unmodified commodity operating systems, its kernel does not require special maintenance support. In addition, performing maintenance in this manner works without a spare node. As a result, maintenance can proceed in parallel across the many nodes of a cluster or farm rather than in serial updates as most "rolling upgrade" scenarios require.[9]

Hardware, VMM, and firmware maintenance. As with OS maintenance, SoftUDC provides online, automated maintenance for hardware, VMM, and firmware. However, this maintenance must be performed on an entire node. As a result, it requires using spare nodes and cannot be parallelized over entire clusters or farms.

Maintenance begins with selecting the first node or set of nodes to be upgraded or repaired. For

The SmartFrog framework automates and controls online maintenance operations.

hardware maintenance, the administrator selects the node; for VMM and firmware updates, SoftUDC makes the selection.

SoftUDC shifts all load off the target node or nodes by migrating the virtual machines to nodes with sufficient spare resources to accommodate them. It can also bring additional nodes online, if needed.

For hardware maintenance, SoftUDC shuts down the newly idled node and flashes a light on its front panel to indicate to the administrator which node among the data center's racks of machines is ready for maintenance. For VMM or firmware updates, SoftUDC deploys the update on the node or nodes and reboots.

Once the update is completed and the nodes have booted, SoftUDC migrates virtual machines back to them. It then selects the next one or more nodes to undergo maintenance and performs the same procedure on them.

Automation. Many of the system's software elements provide control interfaces to support the automation of maintenance. For example, SoftUDC uses appropriate control routines in the VMM to remotely instantiate virtual machines or migrate virtual machines to other nodes. Similarly, a control API in the process migration layer can perform OS maintenance, as well as in the network and storage virtualization layers.

The SmartFrog (www.hpl.hp.com/research/smartfrog) framework automates and controls online maintenance operations. It provides a configuration language for describing updates to the system. The framework also provides daemons that run in all the relevant places—guest OS and management OS for the VMM—that invoke the control routines of the underlying software, enabling remote automatic deployment of updates.

Resource management

SoftUDC builds on earlier results from automated resource management projects at HP Labs. For example, the Hippodrome project[5] demonstrated the first fully automated design loop for allocating and configuring self-managing storage systems. Similarly, SmartFrog explores the automation of deploying and managing distributed software components, applications, and services. We are extending these tools to incorporate the richer mechanisms provided in the resource virtualization layer and using them to automate management operations in SoftUDC.

Virtual farm creation and deployment. We use SmartFrog to automate virtual farm creation and deployment. The framework provides access controls in the gatekeeper on each physical node that will host part of the new virtual farm. The SmartFrog daemon running in the management OS on each physical node starts the farm's virtual machines that will run on that node. This daemon instantiates additional daemons on the newly created virtual machines to install application packages and to instantiate the requested applications and services.

Dynamic load balancing. Once the system is running, dynamic load balancing is the key to minimizing power use and dealing with dynamic workloads, node failures, and oversubscribed hardware. SoftUDC's support for virtual machine migration provides automatic load balancing.

When SoftUDC notices that the virtual machine load is decreasing, it migrates virtual machines onto a smaller set of physical nodes to free up physical resources. It either uses the resulting idle nodes to support other virtual machines with a high load or turns them off to save power.

As the load increases, SoftUDC watches for physical nodes that appear to be close to running out of capacity. It migrates virtual machines off those nodes to other nodes with spare cycles, powering up nodes as needed. SoftUDC can also migrate virtual machines when a node failure creates a demand spike on the remaining physical nodes underlying a virtual farm.

SoftUDC's storage subsystem uses a similar load-balancing strategy. Monitoring agents running in the virtual volume manager detect the load that different virtual machines post on different storage components. When Hippodrome determines that moving data to different locations can optimize system performance or power, the virtual volume manager can transparently affect the data migration.

DATA CENTER IMPLEMENTATIONS

Egenera's BladeFrame (www.egenera.com) is a hardware approach to building a data center that can be partitioned into multiple isolated farms. However, this product, like HP's discontinued Utility Data Center, relies on the availability of support in the hardware to enforce the partitions. In contrast, SoftUDC leverages existing hardware to implement virtualization and makes no assumptions about the hardware infrastructure or fabric topology, such as the existence of virtual LAN support in the network or a SAN.

Several cluster management offerings, including VirtualCenter from VMware (www.vmware.com), simplify the task of managing machines in the data

center by providing a single console spanning all of them. In addition, SoftUDC automates common farm administration tasks and aggregates disjoint storage and networking resources.

Processor virtualization plays a key role in achieving SoftUDC's goals. Several products and projects aim to increase virtualization performance and flexibility including Xen,[1] Denali,[10] and VMware's ESX Server. In contrast, SoftUDC focuses on the secure delivery of virtual farms by linking the virtualization software on individual nodes and exploits the security achieved by running network and storage virtualization code in the VMM.

SoftUDC is a cost-effective and flexible solution to the quest for utility computing. It has no specialized hardware and is built upon existing hardware with new software. SoftUDC's main underlying characteristic is careful virtualization of servers, networking, and storage. It aggregates these resources into a single, centrally managed pool, allows administrators to deploy applications and modify their environment without physically rewiring servers, and facilitates sharing of physical resources while maintaining full isolation. ■

References

1. P. Barham et al., "Xen and the Art of Virtualization," *Proc. 19th ACM Symp. Operating Systems Principles* (SOSP 03), ACM Press, 2003, pp. 164-177.
2. C.P. Sapuntzakis et al., "Optimizing the Migration of Virtual Computers," *Proc. 5th Symp. Operating Systems Design and Implementation* (OSDI 02), ACM Press, 2002, pp. 377-390.
3. M. Karlsson, C. Karamanolis, and X. Zhu, "Triage: Performance Isolation and Differentiation for Storage Systems," *Proc. 12th Int'l Workshop Quality of Service* (IWQoS 04), IEEE Press, 2004, pp. 67-74.
4. C. Lu, G.A. Alvarez, and J. Wilkes, "Aqueduct: Online Data Migration with Performance Guarantees," *Proc. Conf. File Systems and Technology* (FAST 02), Usenix Assoc., 2002, pp. 219-230.
5. E. Anderson et al., "Hippodrome: Running Circles around Storage Administration," *Proc. Conf. File Systems and Technology* (FAST 02), Usenix Assoc., 2002, pp. 175-188.
6. E. Lamb, "Hardware Spending Spatters," *Red Herring,* June 2001, pp. 32-33.
7. S. Osman et al., "The Design and Implementation of Zap: A System for Migrating Computing Environments," *Proc. 5th Symp. Operating Systems Design and Implementation* (OSDI 02), ACM Press, 2002, pp. 361-376.
8. Y. Huang et al., "NT-SwiFT: Software Implemented Fault Tolerance on Windows NT," *Proc. 2nd Usenix Windows NT Symp.,* Usenix Assoc., 1998, pp. 47-56.
9. D.E. Lowell, Y. Saito, and E.J. Samberg, "Devirtualizable Virtual Machines Enabling General, Single-Node, Online Maintenance," to be published in *Proc. 11th Int'l Conf. Architectural Support for Programming Languages and Operating Systems* (Asplos XI), ACM Press, 2004.
10. A. Whitaker, M. Shaw, and S.D. Gribble, "Scale and Performance in the Denali Isolation Kernel," *Proc. 5th Symp. Operating Systems Design and Implementation* (OSDI 02), ACM Press, 2002, pp. 195-209.

Mahesh Kallahalla *is a researcher in the Mobile Software Lab at DoCoMo USA Labs. He worked on the SoftUDC project while a researcher in the Storage Systems Department at HP Labs in Palo Alto, Calif. His research interests include operating systems and storage systems, with a focus on mobile systems, security, and automatic management. Kallahalla received a PhD in electrical engineering from Rice University. He is a member of the IEEE Computer Society, the ACM, and Usenix. Contact him at kallahalla@docomolabs-usa.com.*

Mustafa Uysal *is a researcher in the Storage Systems Department at HP Labs. His research interests are in the design, implementation, and analysis of large-scale storage systems and distributed, data-intensive computing. Uysal received a PhD in computer science from the University of Maryland. He is a member of the IEEE Computer Society, the ACM, and Usenix. Contact him at mustafa.uysal@hp.com.*

Ram Swaminathan *is a researcher in the Internet Systems and Storage Lab at HP Labs. His research interests include design and analysis of algorithms, game theory, cryptography, and security. Swaminathan received a PhD in computer science from Purdue University. He is a permanent member of the Center for Discrete Mathematics and Theoretical Computer Science and a member of ACM SIGACT. Contact him at ram.swaminathan@hp.com.*

David E. Lowell *is a researcher in the Linux Systems and Networks Department at HP Labs. His*

research interests include operating systems, distributed systems, fault tolerance, and OS services for reliability. Lowell received a PhD in computer science from the University of Michigan. He is a member of the ACM. Contact him at david.lowell@hp.com.

***Mike Wray** is a researcher in the Linux Systems and Networks Department at HP Labs Bristol, UK. His research interests include network virtualization and the management of large-scale virtual systems. Wray received an MSc in mathematics from the University of Warwick. He is a member of the IEEE Computer Society and the ACM. Contact him at mike.wray@hp.com.*

***Tom Christian** is a principal scientist in the Linux Systems and Networks Department at HP Labs. His research interests include virtual machine environments, utility computing, and distributed system management. Christian received a BS in applied mathematics from the University of Colorado. Contact him at tom.christian@hp.com.*

***Nigel Edwards** is a researcher in the Application Systems Department at HP Labs Bristol. His research interests include distributed systems, security, and operating systems. Edwards received a PhD in reconfigurable distributed systems from the University of Bristol. He is a member of the ACM and the IEE. Contact him at nigel.edwards@hp.com.*

***Chris I. Dalton** is a research engineer in the Trusted Systems Lab at HP Labs Bristol. His research interests include pragmatic approaches to system and network security, including virtualization. Dalton received a BEng in electronic and electrical engineering from Imperial College London. Contact him at cid@hp.com.*

***Frederic Gittler** is a security architect in the Trusted Systems Lab at HP Labs. His research interests include security architecture and governance. Gittler is a graduate of Ecole Nationale Supérieure des Télécommunications and Ecole Nationale de la Statistique et de l'Administration Economique. Contact him at frederic.gittler@hp.com.*