WHITE PAPER

The Architecture of VMware ESXi

vmware®

Table of Contents

The Architecture of VMware ESXi

Introduction

VMware® ESXi is the next-generation hypervisor, providing a new foundation for virtual infrastructure. This innovative architecture operates independently from any general-purpose operating system, offering improved security, increased reliability, and simplified management. The compact architecture is designed for integration directly into virtualization-optimized server hardware, enabling rapid installation, configuration, and deployment.

Functionally, ESXi is equivalent to ESX 3, offering the same levels of performance and scalability. However, the Linux-based service console has been removed, reducing the footprint to less than 32MB of memory. The functionally of the service console is replaced by new remote command line interfaces in conjunction with adherence to system management standards. Because ESXi is functionally equivalent to ESX, it supports the entire VMware Infrastructure 3 suite of products, including VMware Virtual Machine File System, Virtual SMP, VirtualCenter, VMotion, VMware Distributed Resource Scheduler, VMware High Availability, VMware Update Manager, and VMware Consolidated Backup.

Components of ESXi

The VMware ESXi architecture comprises the underlying operating system, called VMkernel, and processes that run on top of it. VMkernel provides means for running all processes on the system, including management applications and agents as well as virtual machines. It has control of all hardware devices on the server, and manages resources for the applications. The main processes that run on top of VMkernel are:

- Direct Console User Interface (DCUI) — the low-level configuration and management interface, accessible through the console of the server, used primarily for initial basic configuration.
- The virtual machine monitor, which is the process that provides the execution environment for a virtual machine, as well as a helper process known as VMX. Each running virtual machine has its own VMM and VMX process.
- Various agents used to enable high-level VMware Infrastructure management from remote applications.
- The Common Information Model (CIM) system: CIM is the interface that enables hardware-level management from remote applications via a set of standard APIs.

Figure 1 shows a diagram of the overall ESXi architecture. The following sections provide a closer examination of each of these components.

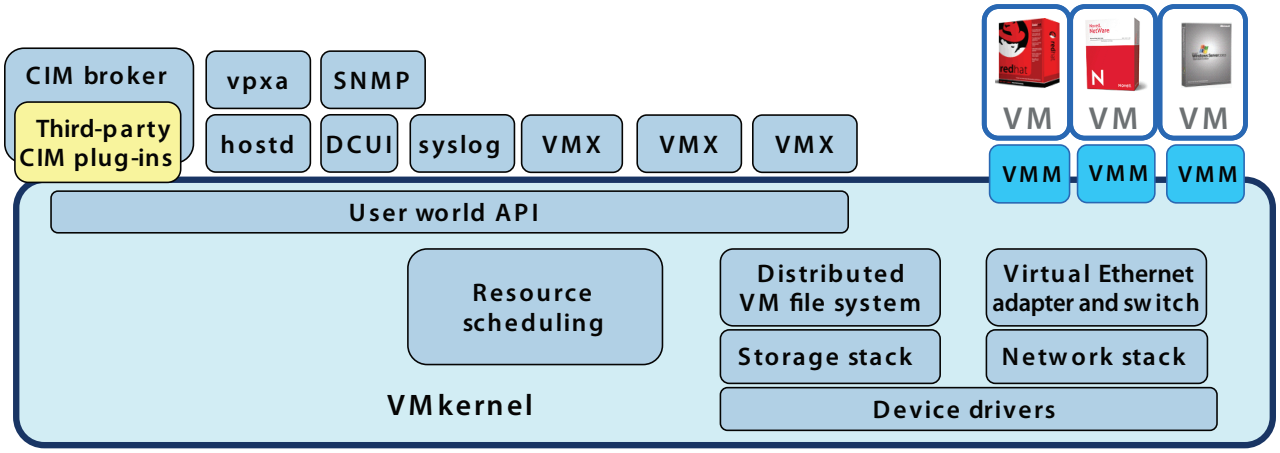

Figure 1: The streamlined architecture of VMware ESXi eliminates the need for a service console.

VMkernel

VMkernel is a POSIX-like operating system developed by VMware and provides certain functionality similar to that found in other operating systems, such as process creation and control, signals, file system, and process threads. It is designed specifically to support running multiple virtual machines and provides such core functionality as:

- Resource scheduling
- I/O stacks
- Device drivers

Some of the more pertinent aspects of the VMkernel are presented in the following sections.

File System

VMkernel uses a simple in-memory file system to hold the ESXi configuration files, log files, and staged patches. For familiarity, the structure of the file system is designed to be the same as that used in the service console of ESX. For example, ESXi configuration files are found in `/etc/vmware` and log files are found in `/var/log/vmware`. Staged patches are uploaded to `/tmp`.

This file system is independent of the VMware VMFS file system used to store virtual machines. Just as with ESX, a VMware VMFS datastore may be created on a local disk in the host system or on shared storage. If the only VMFS datastores used by the host are on external shared storage, the ESXi system does not actually require a local hard drive. By running diskless setups, you can increase reliability by avoiding hard drive failures and reduce power and cooling consumption.

Remote command line interfaces provide file management capabilities for both the in-memory file system and the VMware VMFS datastores. Access to the file system is implemented via HTTPS `get` and `put` Access is authenticated via users and groups configured locally on the server and is controlled by local privileges.

Because the in-memory file system does not persist when the power is shut down, log files do not survive a reboot. ESXi has the ability to configure a remote syslog server, enabling you to save all log information on an external system.

Users and Groups

Users and groups can be defined locally on the ESXi system. They provide a way to distinguish users that access the system via the Virtual Infrastructure Client, the remote command line interfaces, or the VIM API.

Groups can be used to combine multiple users, just as in other operating systems. Groups can be used, for example, to set privileges for many users at once. There are a few system users and groups, which are predefined in order to identify certain processes running in the VMkernel.

Administrative privileges can be set individually for each user or group. User and group definitions are stored on the file system in the files `/etc/passwd`, `/etc/shadow`, and `/etc/group`, and as in other operating systems, passwords are generated using standard `crypt` functions.

User Worlds

The term "user world" refers to a process running in the VMkernel operating system. The environment in which a user world runs is limited compared to what would be found in a general-purpose POSIX-compliant operating system such as Linux. For example:

- The set of available signals is limited.
- The system API is a subset of POSIX.
- The `/proc` file system is very limited.
- A single swap file is available for all user world processes. If a local disk exists, the swap file is created automatically in a small VFAT partition. Otherwise, the user is free to set up a swap file on one of the attached VMFS datastores.

In short, a user world is not intended as a general-purpose mechanism to run arbitrary applications but provides only enough of a framework for processes that need to run in the hypervisor environment.

Several important process run in user worlds. These can be thought of as native VMkernel applications and are described in the following sections.

Direct Console User Interface

The Direct Console User Interface (DCUI) is the local user interface that is displayed only on the console of an ESXi system. It provides a BIOS-like, menu-driven interface for interacting with the system. Its main purpose is initial configuration and troubleshooting. One of the system users defined in VMkernel is `dcui`, which is used by the DCUI process to identify itself when communicating with other components in the system.

The DCUI configuration tasks include:

- Set administrative password
- Configure networking, if not done automatically with DHCP

Troubleshooting tasks include:

- Perform simple network tests
- View logs
- Restart agents
- Restore defaults

The intention is that the user carries out minimum configuration with the DCUI, then uses a remote management tool, such as the VI Client, VirtualCenter, or the remote command line interfaces, to perform all other configuration and ongoing management tasks.

Anyone using the DCUI must enter an administrative-level password, such as the root password. Initially, the root password is blank. VMware strongly recommends that you set this password before connecting the server to any untrusted network. For example, turn on the server without any network cable attached, set the password, attach the server to the network, then select the option for obtaining IP information via DHCP. Alternatively, if the server will be on a trusted network, you can set the administrator password using the VI Client. You can give additional local users the ability to access the DCUI by making them a part of the `localadmin` group. This approach provides a way to grant access to the DCUI without handing out the root password, but obviously you would grant this right only to trusted accounts.

Other User World Processes

Agents used by VMware to implement certain management capabilities have been ported from running in the service console to running in user worlds.

- The `hostd` process provides a programmatic interface to VMkernel and is used by direct VI Client connections as well as the VI API. It is the process that authenticates users and keeps track of which users and groups have which privileges. It also allows you to create and manage local users.
- The `vpxa` process is the agent used to connect to VirtualCenter. It runs as a special system user called `vpxuser`. It acts as the intermediary between the `hostd` agent and VirtualCenter.
- The agent used to provide VMware HA capabilities has also been ported from running in the service console to running in its own user world.
- A `syslog` daemon also runs as a user world. If you enable remote logging, that daemon forwards all the logs to the remote target in addition to putting them in local files.
- A process that handles initial discovery of an iSCSI target, after which point all iSCSI traffic is handled by the VMkernel, just as it handles any other device driver. Note that the iSCSI network interface is the same as the main VMkernel network interface.

In addition, ESXi has processes that enable NTP-based time synchronization and SNMP monitoring.

Open Network Ports

A limited number of network ports are open on ESXi. The most important ports and services are the following:

- 80 — This port serves a reverse proxy that is open only to display a static Web page that you see when browsing to the server. Otherwise, this port redirects all traffic to port 443 to provide SSL-encrypted communications to the ESXi host.
- 443 (reverse proxy) — This port also acts as a reverse proxy to a number of services to provide SSL-encrypted communication to these services. The services include the VMware Virtual Infrastructure API (VI API), which provides access to the RCLIs, VI Client, VirtualCenter Server, and the SDK.
- 427 (service location protocol) — This port provides access for the service location protocol, a generic protocol to search for the VI API.
- 5989 — This port is open for the CIM server, which is an interface for Third-party management tools.
- 902 — This port is open to support the older VIM API, specifically the older versions of the VI Client and VirtualCenter.

Consult the *ESX Server 3i Configuration Guide* for the complete list of open ports.

System Image Design

ESXi is designed for distribution in various formats, including directly embedded in the firmware of a server or as software to be installed on a server's boot disk. Figure 2 shows a diagram of the contents of the ESXi system image. Regardless of whether the image exists on flash memory or on the hard drive of a computer, the same components are present:

- A 4MB bootloader partition, which runs upon system boot up.
- A 48MB boot bank, which contains the 32MB core hypervisor code, along with a second alternate boot bank of the same size. The reason for two boot banks is explained below.
- A 540MB store partition, which holds various utilities, such as the VI Client and VMware Tools images.
- A 110MB core dump partition, which is normally empty but which can hold diagnostic information in case of a system problem.

The ESXi system has two independent banks of memory, each of which stores a full system image, as a fail-safe for applying updates. When you upgrade the system, the new version is loaded into the inactive bank of memory, and the system is set to use the updated bank when it reboots. If any problem is detected during the boot process, the system automatically boots from the previously used bank of memory. You can also intervene manually at boot time to choose which image to use for that boot, so you can back out of an update if necessary.

At any given time, there are typically two versions of VI Client and two versions of VMware Tools in the store partition, corresponding to the hypervisor versions in the two boot banks. The specific version to use is determined by which boot bank is currently active.

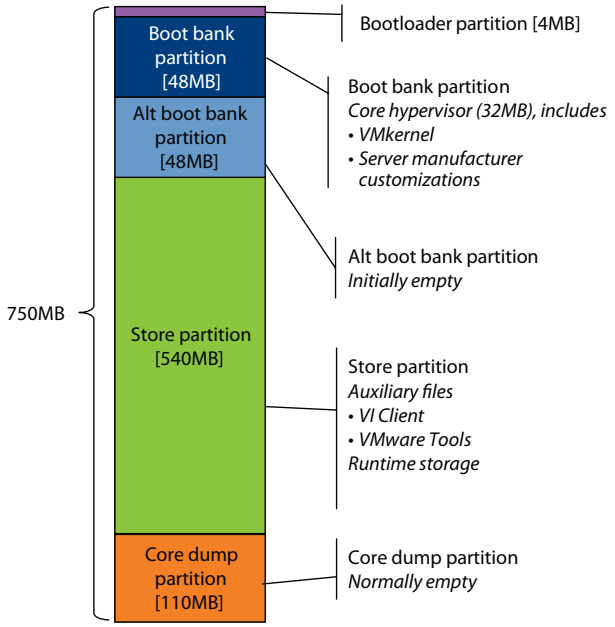

Figure 2: Contents of the ESXi system image

The core hypervisor code also can contain custom code provided by server vendors (OEMs) that provides additional functionality, such as hardware monitoring and support information. These customizations would be present, for example, if ESXi had been obtained in embedded form from the server manufacturer or if a custom version of ESXi was installed onto the hard drive. Any update to an existing ESXi installation automatically incorporates the proper update to this custom code.

Startup and Operation

When the system boots for the first time, the VMkernel discovers devices and selects appropriate drivers for them. It also discovers local disk drives and, if the disks are empty, formats them so they can be used to store virtual machines.

During this initial boot, the VMkernel automatically creates the configuration files using reasonable default values (for example, using DHCP to obtain network identity information). Users can adjust the defaults with the direct console user interface or with the standard VMware management tools: VMware VirtualCenter and the VI Client. In the embedded version of ESXi, the configuration is stored in a specific part of the memory module that is both readable and writable. On subsequent reboots, the system reads the configuration from this persistent memory. In the rest of the boot process, the system is initialized and the resident file system is built in memory. The hardware drivers are loaded, the various agents are started, and finally the DCUI process is started.

Once the system is up and running, all further routine operations occur in much the same way as they do in ESX 3. Because ESXi no longer includes a service console, many of the management activities performed on the ESX platform are no longer necessary; they were required only to configure and manage the service console itself. Other management tasks previously done in the service console are now performed in one of the following ways:

- Using the VI Client, which provides a Windows-based graphical user interface for interactive configuration of the platform. The VI Client has been enhanced to provide capabilities that were previously available only in the service console.
- Using the remote command line interfaces, new interfaces that enable scripting and command-line–based configuration of the platform from a Linux or Windows-based server, via an encrypted and authenticated communication channel.
- Using external agents that leverage well-defined APIs, such as the VI API and the CIM management standard.

In addition, you can manage ESXi using VirtualCenter, just as you would any ESX 3 system. You can have a mixed environment of ESX 3 and ESXi systems. VirtualCenter presents both types of systems in the VI Client user interface in essentially the same way; certain features unique to ESXi management appear for hosts equipped with that version.

Management Model for ESXi

The architecture of ESXi brings along with it a new management model. The core tenets of this model are: a compute infrastructure based on stateless, interchangeable devices; centralized management, including administration and policy; communication with the system using well-defined and standardized APIs instead of unstructured interactive sessions that are difficult to lock down and audit. The following section describes some aspects of this management model in more detail.

State Information

The state of an ESXi system is fully described by a handful of configuration files. These files control such functions as configuration of virtual networking and storage, SSL keys, server network settings, and local user information Although these configuration files are all found in the in-memory file system, they are also periodically copied to persistent storage. For example, in ESXi Embedded, there is a small part of the server firmware that is designated as read-write. In case of sudden power loss, you can reboot the server and it is restored to the exact configuration of the last copy. Nothing else is required to maintain state, so the internal hard disk can even be eliminated from the server.

You can also download a backup file that contains all the state information. This allows you to replicate the state of an ESXi system onto another similar system. You can create backups of your server configuration, and if a server fails catastrophically, you can easily replace it with an identical unit, then bring that new unit to the same state by restoring the backup file.

Common Information Model

The Common Information Model (CIM) is an open standard that defines how computing resources can be represented and managed. It enables a framework for agentless, standards-based monitoring of hardware resources for ESXi. This framework consists of a CIM object manager, often called a CIM broker, and a set of CIM providers.

CIM providers are used as the mechanism to provide management access to device drivers and underlying hardware. Hardware vendors, which include both the server manufacturers and specific hardware device vendors, can write providers to provide monitoring and management of their particular devices. VMware also writes providers that implement monitoring of server hardware, ESX/ESXi storage infrastructure, and virtualization-specific resources. These providers run inside the ESXi system and hence are designed to be extremely lightweight and focused on specific management tasks. The CIM object manager in ESXi implements a standard CMPI interface developers can use to plug in new providers. However, the providers must be packaged with the system image, and cannot be installed at run time.

The CIM broker takes information from all CIM providers and presents it to the outside world via standard APIs, including WS-MAN. Figure 3 shows a diagram of the CIM management model.

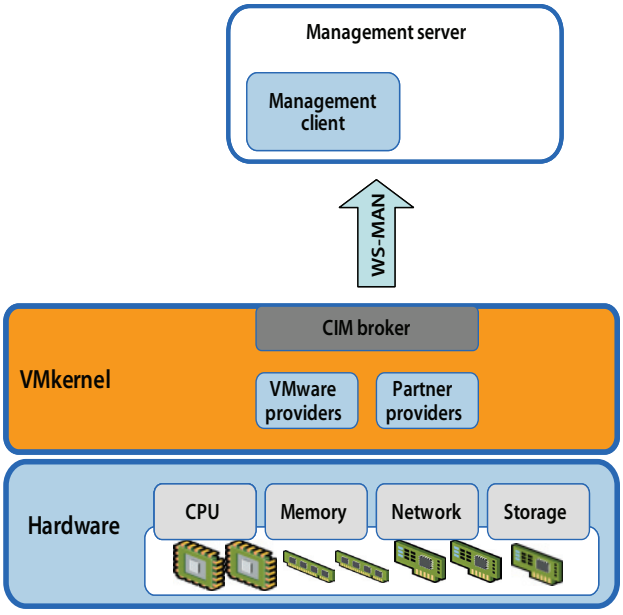

Figure 3: CIM Management model

VI API

The VMware Virtual Infrastructure API provides a powerful interface for developing applications to integrate with the VMware Infrastructure. The VI API enables your program or framework to invoke VirtualCenter Web Service interface functions on VirtualCenter to manage and control ESX/ESXi. The VI SDK provides developers with a full environment for creating applications that interact with ESXi in a variety of programming languages.

The VI API is actually what is used by the management clients provided by VMware, such as the VI Client and remote command line interfaces. Furthermore, this API works for VirtualCenter as well as ESX/ESXi. The only difference is that certain functions that affect multiple hosts, such as VMotion, are implemented only in VirtualCenter. Figure 4 depicts how the VI API is used with VMware Infrastructure.

Together, the VI API and the CIM standard provide a comprehensive way to manage an ESXi system from a remote or central location. The advantage of this model is that, instead of relying upon locally installed agents, which must be adjusted whenever the underlying platform changes and reinstalled and managed as they are updated, all software related to monitoring and management of a system can exist on an external and centralized system. It becomes much easier to maintain this software, as opposed to managing multiple distributed agents. This approach to management also further enables the ESXi host to become a stateless entity, because there is nothing to install locally on the host. Eliminating agents from running locally also means that all the compute resources are available for running virtual machines.

Summary

The ESXi architecture offers a variety of advantages over other virtualization platforms, including::

- Little state information — An ESXi system can be treated for practical purposes as a stateless compute node, with all the state information easily uploaded from a saved configuration file.
- Better security — With a small footprint and minimal interfaces, an ESXi system has a lower overall attack surface.
- Hardware-like reliability — When it is integrated into firmware, software is much less likely to become corrupted than when it is stored on disk. The option of eliminating the local disk drive can provide even greater system reliability.
- Table 1 summarizes the architectural differences between ESX 3 and ESXi

	VMware ESXi	VMware ESX 3
On-disk footprint	32MB	2GB
Bootstrap	Direct from boot loader	Service console driven
Direct management interaction	DCUI	Service console shell session
Hardware monitoring agents	CIM plug-in modules	Full applications in service console
Other agents	Implemented via VI SDK only	Full applications in service console
Scripts, automation and troubleshooting	DCUI, remote command line interfaces, and VI SDK	Service console shell and VI SDK
Other software	Moved to outside environment	Resident in service console

Table 1: Differences between ESXi and ESX 3

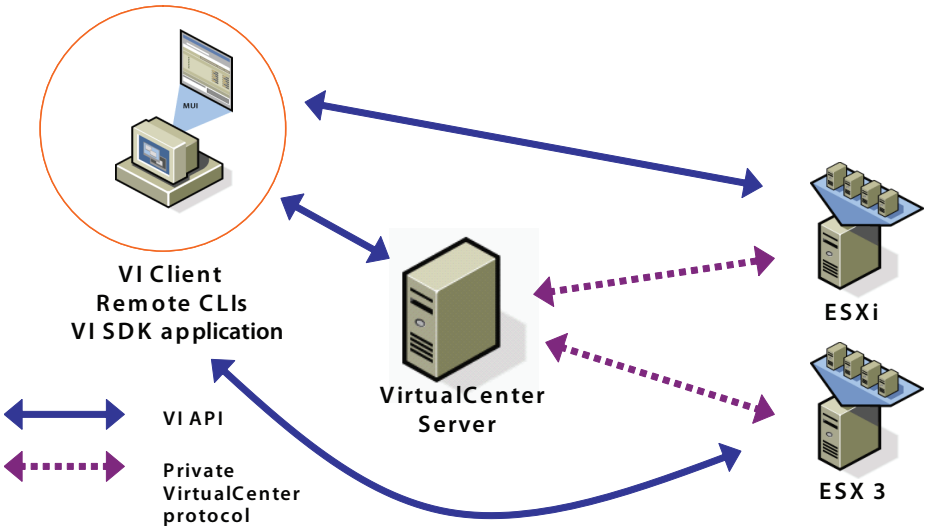

Figure 4: Using the VMware Virtual Infrastructure API in a VMware Infrastructure environment

About the Author

Charu Chaubal is Technical Marketing Manager at VMware, where he specializes in enterprise datacenter management with a focus on security. Previously, he worked at Sun Microsystems, where he had over 7 years experience with designing and developing distributed resource management and grid infrastructure software solutions. Charu received a Bachelor of Science in Engineering from the University of Pennsylvania, and a Ph.D. from the University of California at Santa Barbara, where he studied the numerical modeling of complex fluids. He is the author of numerous publications and several patents in the fields of datacenter automation and numerical price optimization.

Acknowledgements

The author would like to thank Olivier Cremel and John Gilmartin for their invaluable help in producing this document.

Revision: 20081024 WP-030-PRD-02-02

VMware, Inc. 3401 Hillview Ave. Palo Alto CA 94304 USA Tel 650-475-5000 Fax 650-475-5001 www.vmware.com

vmware®