

Cloud Computing

COURSE LECTURER : SAMUEL WINFUL

CLOUD ENABLING TECHNOLOGIES

Service Oriented Architecture

- REST and Systems of Systems
- Web Services
- Publish-Subscribe Model
- Basics of Virtualization
- Types of Virtualization
- Implementation Levels of Virtualization
- Virtualization Structures
- Tools and Mechanisms
- Virtualization of CPU
- Memory
- I/O Devices
- Virtualization Support and Disaster Recovery.

Service Oriented Architecture(SOA)

A service encapsulates a software component that gives a set of coherent and related functionalities that can be reused and integrated into larger and more complex applications.

The term service is a general abstraction that encompasses several different implementations using different technologies and protocols.

Don Box identifies four major characteristics with the intention of identify a service.

- *Boundaries are explicit*
- *Services are autonomous*
- *Services share schema and contracts*
- *Services compatibility is determined based on policy*

Boundaries are explicit

- A service oriented applications are generally composed of services that are spread across different domains, trust authorities and execution environments.


Services are autonomous

- Services are components that exist to offer functionality.
- Services are aggregated and coordinated to build more complex system.
- Services are not designed to be part of a specific system but they can be integrated in several software systems.
- The notion of autonomy also affects the way services handle failures.

Services share schema and contracts

- Services never share class and interface definitions.
- In object oriented systems, services are not expressed in terms of classes or interfaces but they define in terms of schemas and contracts.
- Technologies such as XML and SOAP provide the appropriate tools to support such features rather than class definition and an interface declaration.

Services compatibility is determined based on policy

- Service orientation separates structural compatibility from semantic compatibility.
 - Structural compatibility is based on contracts and schema and can be validated by machine based techniques.
 - Semantic compatibility is expressed in the form of policies that define the capabilities and requirements for a service.
- 

-
- Service Oriented architecture is an architectural style supporting service orientation.
 - This architectural style organizes a software system into a collection of interacting services.
 - SOA encompasses a set of design principles that structure system development and provide means for integrating components into a coherent and decentralized system.
 - SOA based computing packages functionalities into a set of interoperable services, which can be integrated into different software systems belonging to separate business domains.
 - There are two major roles exist in SOA
 - Service provider
 - Service consumer

-
- First, the service provider is the maintainer of the service and the organization that makes available one or more services for others to use.
 - To advertise services, the provider can publish them in a registry along with a service contract that specifies the nature of the service, how to use the service, the requirements for the service
 - Second, the service consumer can locate the service metadata in the registry and develop the required client components to bind and use the service.
 - Service providers and consumers can belong to different organization bodies.
 - It is very common in SOA based computing systems that components play the roles of both service provider and service consumer.


-
- Services might aggregate information and data retrieved from other services or create workflows of services to satisfy the request of a given service consumer. This practice is called as service orchestration, which more generally describes the automated arrangement, coordination and management of more complex computer systems, middleware and services.
 - Another important interaction pattern is service composition is the coordinated interaction of services without a single point of control.
 - SOA provides a reference model for architecting several software systems primarily for enterprise business applications and systems.

-
- Interoperability, standards and service contracts plays a fundamental role.
 - In particular, the following list of guiding principles characterize SOA platforms:
 - *Standardized service contract*
 - Services adhere to a given communication agreement, which is specified through one or more service description documents.
 - *Loose coupling*
 - ✓ Services are designed as self-contained components, maintain relationships that minimize dependencies and only require being aware of each other.
 - ✓ Service contracts will enforce the required interaction among services.
 - ✓ This simplifies the flexible aggregation of services and enables a more agile design strategy that supports the evolution of the enterprise business.

■ *Abstraction*

- ✓ A service is completely defined by service contracts and description documents.
- ✓ Abstraction hiding the logic, which is encapsulated within their implementation.
- ✓ The use of service description documents and contracts removes the need to consider the technical implementation details.
- ✓ It provides a more intuitive framework to define software systems within a business context.

■ *Reusability*

- ✓ Designed as components, services can be reused more efficiently, thus reducing development time and the associated costs.
 - ✓ Reusability allows for a more agile design and cost effective system implementation and deployment.
- 


- ***Autonomy***

- ✓ Services have control over the logic they encapsulate and do not know about their implementation.

- ***Lack of state***

- ✓ By providing a stateless interaction pattern, services increase the chance of being reused and aggregated, particularly in a scenario in which a single service is used by multiple consumers that belong to different administrative and business domains.

- ***Discoverability***

- ✓ Services are defined by description documents that constitute supplemental metadata through which they can be effectively discovered.
 - ✓ Service discovery provides an effective means for utilizing third party resources.
- 

- ***Composability***

- ✓ Using services as building blocks, difficult operations can be implemented.
- ✓ Service orchestration and choreography provide a solid support for composing services and achieving desired business goals.

- Together with these principles, other resources guide the use of SOA for enterprise application integration (EAI).

- The SOA manifest integrates the previously described principles with general considerations about the overall goals of a service oriented approach to enterprise application software design and what is valued in SOA.

-
- ❑ Modeling frameworks and methodologies, such as the Service Oriented Modeling Framework (SOMF) and reference architectures introduced by the Organization for Advancement of Structured Information Standards (OASIS), provide means for effectively realizing service oriented architectures.
 - ❑ SOA can be realized through several technologies.
 - ❑ The first implementations of SOA have leveraged distributed object programming technologies such as CORBA and DCOM.
 - ❑ CORBA has been a suitable platform for realizing SOA systems because it provides interoperability among different implementations and has been designed as a specification supporting the development of industrial applications.
 - ❑ Nowadays, SOA is mostly realized through Web services technology, which provides an interoperable platform for connecting systems and applications.

Web Services

- ❑ Web services are the prominent technology for implementing SOA systems and applications.
- ❑ They leverage Internet technologies and standards for building distributed systems. Several aspects make Web services the technology of choice for SOA.
 - First, they allow for interoperability across different platforms and programming languages.
 - Second, they are based on well-known and vendor independent standards such as HTTP, SOAP, XML and WSDL.
 - Third, they provide an intuitive and simple way to connect heterogeneous software systems, enabling the quick composition of services in a distributed environment.
 - Finally, they provide the features required by enterprise business applications to be used in an industrial environment.

-
- ❑ They define facilities for enabling service discovery, which allows the system architect to more efficiently compose SOA applications and service metering to assess whether a specific service complies with the contract between the service provider and the service consumer.
 - ❑ The concept behind a Web service is very simple.
 - ❑ Using as a basis the object oriented abstraction, a Web service exposes a set of operations that can be invoked by leveraging Internet based protocols.
 - ❑ The semantics for invoking Web service methods is expressed through interoperable standards such as XML and WSDL, which also provide a complete framework for expressing simple and complex types in a platform independent manner.

- ❑ Web services are made accessible by being hosted in a Web server
- ❑ HTTP is the most popular transport protocol used for interacting with Web services.

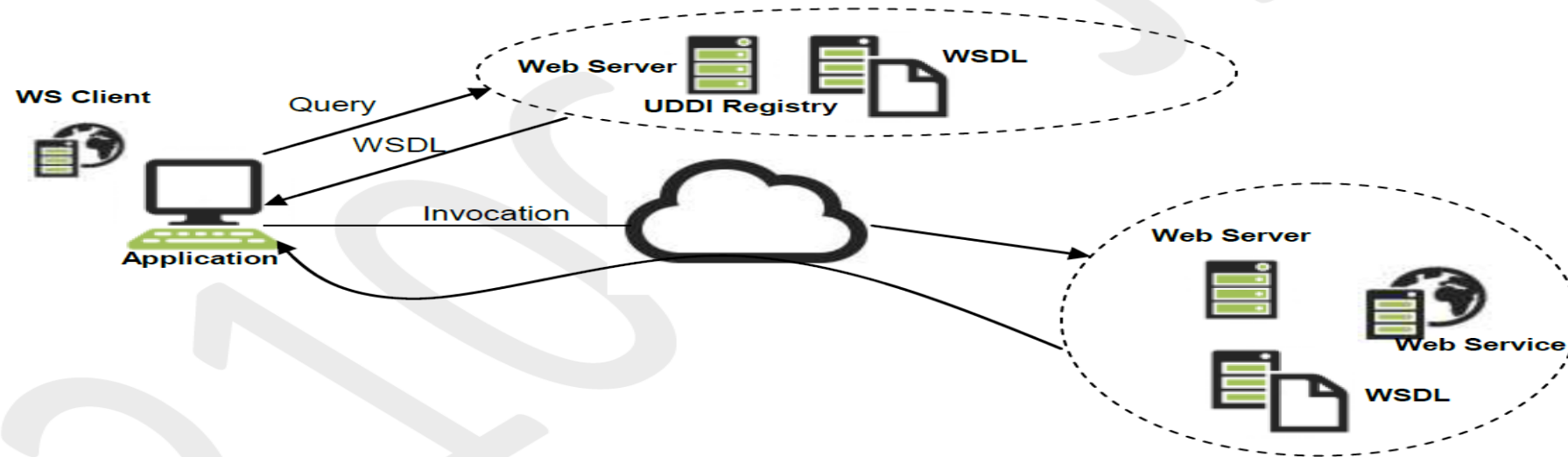


Figure 2.1 Reference scenario for Web Services

- ❑ Figure 2.1 describes the common use case scenarios for Web services.

-
- ❑ System architects develop a Web service with their technology of choice and deploy it in compatible Web or application servers.
 - ❑ The service description document is expressed by means of Web Service Definition Language (WSDL), can be either uploaded to a global registry or attached as a metadata to the service itself.
 - ❑ Service consumers can look up and discover services in global catalogs using Universal Description Discovery and Integration (UDDI).
 - ❑ The Web service description document allows service consumers to automatically generate clients for the given service and embed them in their existing application.
 - ❑ Web services are now extremely popular, so bindings exist for any mainstream programming language in the form of libraries or development support tools.

-
- ❑ This makes the use of Web services seamless and straightforward with respect to technologies such as CORBA that require much more integration effort.
 - ❑ Moreover, being interoperable, Web services constitute a better solution for SOA with respect to several distributed object frameworks, such as .NET Remoting, Java RMI, and DCOM/COM1, which limit their applicability to a single platform or environment.
 - ❑ Besides the main function of enabling remote method invocation by using Web based and interoperable standards, Web services encompass several technologies that put together and facilitate the integration of heterogeneous applications and enable service oriented computing.
 - ❑ Figure 2.2 shows the Web service technologies stack that lists all the components of the conceptual framework describing and enabling the Web services abstraction.
 - ❑ These technologies cover all the aspects that allow Web services to operate in a distributed environment, from the specific requirements for the networking to the discovery of services.

Web Service Flow (WSFL)	Security	Management	QoS
Service Discovery (UDDI)			
Service Publication (UDDI)			
Service Description (ASDL)			
XML based messaging (SOAP)			
Network (HTTP, FTP, Email, ...)			

Figure 2.2 Web services technologies stack

-
- ❑ The backbone of all these technologies is XML, which is also one of the causes of Web service's popularity and ease of use.
 - ❑ XML based languages are used to manage the low level interaction for Web service method calls (SOAP), for providing metadata about the services (WSDL), for discovery services (UDDI), and other core operations.
 - ❑ In practice, the core components that enable Web services are SOAP and WSDL.
 - ❑ Simple Object Access Protocol (SOAP) is an XML based language for exchanging structured information in a platform-independent manner, constitutes the protocol used for Web service method invocation.
 - ❑ Within a distributed context leveraging the Internet, SOAP is considered an application layer protocol that leverages the transport level, most commonly HTTP, for IPC.

-
- ❑ SOAP structures the interaction in terms of messages that are XML documents mimicking the structure of a letter, with an envelope, a header, and a body.
 - ❑ The envelope defines the boundaries of the SOAP message.
 - ❑ The header is optional and contains relevant information on how to process the message.
 - ❑ In addition to that it contains information such as routing and delivery settings, authentication, transaction contexts and authorization assertions.
 - ❑ The body contains the actual message to be processed.
 - ❑ The main uses of SOAP messages are method invocation and result retrieval.

Host : www.sample.com

Content-Type: application/soap+xml; charsetutf-8

Content-Length: <Size>

```
<?xml version= "1.0">
```

```
<soap: Envelope xmlns:soap= "http://www.w3.org/2001/12/soap-envelope"
```

```
soap:encodingStyle= "http://www.w3.org/2001/12/soap-enoding" >
```

```
<soap:Header></soap:Header>
```

```
<soap:Body xmlns=http://www.sample.com/stock>
```

```
<m:GetPrice>
```

```
<m: StockName>DELL</m:StockName>
```

```
</m:GetPrice>
```

```
</soap:Body>
```

```
</soap: Envelope>
```

Figure 2.3a SOAP Message

POST /StockPrice HTTP/1.1

Host : www.sample.com

Content-Type: application/soap+xml; charsetutf-8

Content-Length: <Size>

<?xml version= "1.0">

<soap: Envelope xmlns:soap= "<http://www.w3.org/2001/12/soap-envelope>"

soap:encodingStyle= "<http://www.w3.org/2001/12/soap-enoding>" >

<soap:Header></soap:Header>

<soap:Body xmlns=http://www.sample.com/stock>

<m:GetPriceResponse>

<m: Price>58.5</m:Price>

</m:GetPriceResponse>

</soap:Body>

</soap: Envelope>

Figure 2.3 b SOAP
Message

-
- ❑ Figure 2.3a & b shows an example of a SOAP message used to invoke a Web service method that retrieves the price of a given stock and the corresponding reply.
 - ❑ Despite the fact that XML documents are easy to produce and process in any platform or programming language, SOAP has often been considered quite inefficient because of the excessive use of markup that XML imposes for organizing the information into a well-formed document.
 - ❑ Therefore, lightweight alternatives to the SOAP/XML pair have been proposed to support Web services.

REST and Systems of Systems

- ❑ The most relevant alternative to SOAP/XML pair is Representational State Transfer (REST), which provides a model for designing network based software systems utilizing the client / server model and leverages the facilities provided by HTTP for IPC without additional burden.
- ❑ In a RESTful system, a client sends a request over HTTP using the standard HTTP methods (PUT, GET, POST, and DELETE) and the server issues a response that includes the representation of the resource.
- ❑ By relying on this minimal support, it is possible to provide whatever it needed to replace the basic and most important functionality provided by SOAP, which is method invocation.
- ❑ The GET, PUT, POST, and DELETE methods constitute a minimal set of operations for retrieving, adding, modifying and deleting the data.

-
- ❑ Together with an appropriate URI organization to identify resources, all the atomic operations required by a Web service are implemented.
 - ❑ The content of data is still transmitted using XML as part of the HTTP content, but the additional markup required by SOAP is removed.
 - ❑ For this reason, REST represents a lightweight alternative to SOAP, which works effectively in contexts where additional aspects beyond those manageable through HTTP are absent.
 - ❑ RESTful Web services operate in an environment where no additional security beyond the one supported by HTTP is required.
 - ❑ This is not a great limitation, and RESTful Web services are quite popular and used to deliver functionalities at enterprise scale:
 - ◦ Twitter
 - ◦ Yahoo! (search APIs, maps, photos, etc)
 - ◦ Flickr
 - ◦ Amazon.com

-
- ❑ Web Service Description Language (WSDL) is an XML based language for the description of Web services.
 - ❑ It is used to define the interface of a Web service in terms of methods to be called and types and structures of the required parameters and return values.
 - ❑ In Figure 2.3 we notice that the SOAP messages for invoking the Get Price method and receiving the result do not have any information about the type and structure of the parameters and the return values.
 - ❑ This information is stored within the WSDL document attached to the Web service.
 - ❑ Therefore, Web service consumer applications already know which types of parameters are required and how to interpret results.

-
- ❑ As an XML based language, WSDL allows for the automatic generation of Web service clients that can be easily embedded into existing applications.
 - ❑ Moreover, XML is a platform and language independent specification, so clients for web services can be generated for any language that is capable of interpreting XML data.
 - ❑ This is a fundamental feature that enables Web service interoperability and one of the reasons that make such technology a solution of choice for SOA.
 - ❑ Besides those directly supporting Web services, other technologies that characterize Web 2.0 and contribute to enrich and empower Web applications and then SOA based systems.
 - ❑ These fall under the names of Asynchronous JavaScript and XML (AJAX), JavaScript Standard Object Notation (JSON) and others.

-
- ❑ AJAX is a conceptual framework based on JavaScript and XML that enables asynchronous behavior in Web applications by leveraging the computing capabilities of modern Web browsers.
 - ❑ This transforms simple Web pages in complete applications and used to enrich the user experience.
 - ❑ AJAX uses XML to exchange data with Web services and applications • An alternative to XML is JSON, which allows representing objects and collections of objects in a platform independent manner.
 - ❑ Often it is preferred to transmit data in an AJAX context because compared to XML, it is a lighter notation and therefore allows transmitting the same amount of information in a more concise form.

Publish-Subscribe Model

- ❑ Publish-and-subscribe message model introduces a different message passing strategy, one that is based on notification among components.
- ❑ There are two major roles:
 - ✓ The publisher and the subscriber
- ❑ The publisher provides facilities for the subscriber to register its interest in a specific topic or event.
- ❑ Specific conditions holding true on the publisher side can trigger the creation of messages that are attached to a specific event.
- ❑ A message will be available to all the subscribers that registered for the corresponding event.

❑ There are two major strategies for dispatching the event to the subscribers:

✓ Push strategy

❑ In this case it is the responsibility of the publisher to notify all the subscribers using method invocation.

✓ Pull strategy

❑ In this case the publisher simply makes available the message for a specific event and it is responsibility of the subscribers to check whether there are messages on the events that are registered.

❑ Publish and subscribe model is very suitable for implementing systems based on the one to many communication model and simplifies the implementation of indirect communication patterns.

❑ It is, in fact, not necessary for the publisher to know the identity of the subscribers to make the communication happen

Basics of Virtualization

- ❑ Virtualization technology is one of the fundamental components of cloud computing, especially in regard to infrastructure based services.
- ❑ Virtualization allows the creation of a secure, customizable and isolated execution environment for running application.
- ❑ Virtualization is a large umbrella of technologies and concepts that are meant to provide an abstract environment whether virtual hardware or an operating system to run applications.
- ❑ The term virtualization is often synonymous with hardware virtualization, which plays a fundamental role in efficiently delivering Infrastructure as a Service (IaaS) solutions for cloud computing.

❑ Virtualization technologies have gained renewed interested recently due to the confluence of several phenomena:

- ✓ Increased performance and computing capacity.
- ✓ Underutilized hardware and software resources
- ✓ Lack of space
- ✓ Greening initiatives
- ✓ Rise of administrative costs

❑ Virtualization is a broad concept that refers to the creation of a virtual version of something, whether hardware, a software environment, storage and a network.

❑ In a virtualized environment, there are three major components:

- ✓ Guest
- ✓ Host
- ✓ Virtualization layer

❑ The guest represents the system component that interacts with the virtualization layer rather than with the host, as would normally happen.

❑ The host represents the original environment where the guest is supposed to be managed.

❑ The virtualization layer is responsible for recreating the same or a different environment where the guest will operate.

Characteristics of virtualized environments

❑ Increased security

- ✓ The ability to control the execution of a guest in a completely transparent manner opens new possibilities for delivering a secure, controlled execution environment.
- ✓ The virtual machine represents an emulated environment in which the guest is executed.
- ✓ This level of indirection allows the virtual machine manager to control and filter the activity of the guest, thus preventing some harmful operations from being performed.

❑ Managed execution Virtualization of the execution environment not only allows increased security, but a wider range of features also can be implemented.

❑ In particular, sharing, aggregation, emulation, and isolation are the most relevant features

❑ Sharing

- ✓ Virtualization allows the creation of a separate computing environment within the same host.
- ✓ In this way it is possible to fully exploit the capabilities of a powerful guest, which would otherwise be underutilized.

❑ Aggregation

- ✓ Not only is it possible to share physical resource among several guests but virtualization also allows aggregation, which is the opposite process.
- ✓ A group of separate hosts can be tied together and represented to guests as a single virtual host.

❑ Emulation

- ✓ Guest programs are executed within an environment that is controlled by the virtualization layer, which ultimately is a program.
- ✓ This allows for controlling and tuning the environment that is exposed to guests.

❑ Isolation

- ✓ Virtualization allows providing guests whether they are operating systems, applications, or other entities with a completely separate environment, in which they are executed.
- ✓ The guest program performs its activity by interacting with an abstraction layer, which provides access to the underlying resources.
- ✓ Benefits of Isolation
 - First it allows multiple guests to run on the same host without interfering with each other.
 - Second, it provides a separation between the host and the guest.

❑ Another important capability enabled by virtualization is performance tuning.

❑ This feature is a reality at present, given the considerable advances in hardware and software supporting virtualization

-
- ❑ It becomes easier to control the performance of the guest by finely tuning the properties of the resources exposed through the virtual environment.
 - ❑ This capability provides a means to effectively implement a quality of service (QoS) infrastructure that more easily fulfills the service level agreement (SLA) established for the guest.
 - ❑ Portability
 - ✓ The concept of portability applies in different ways according to the specific type of virtualization considered.
 - ✓ In the case of a hardware virtualization solution, the guest is packaged into a virtual image that, in most cases, can be safely moved and executed on top of different virtual machines

Types of Virtualization

- ❑ Virtualization is mainly used to emulate execution environments, storage and networks.
- ❑ Execution virtualization techniques into two major categories by considering the type of host they require.
- ❑ Process level techniques are implemented on top of an existing operating system, which has full control of the hardware.
- ❑ System level techniques are implemented directly on hardware and do not require or require a minimum of support from existing operating system.
- ❑ Within these two categories we can list various techniques that offer the guest a different type of virtual computation environment:
 - ✓ Bare hardware
 - ✓ Operating system resources
 - ✓ Low level programming language
 - ✓ Application libraries

-
- ❑ Execution virtualization includes all techniques that aim to emulate an execution environment that is separate from the one hosting the virtualization layer.
 - ❑ All these techniques concentrate their interest on providing support for the execution of programs, whether these are the operating system, a binary specification of a program compiled against an abstract machine model or an application.
 - ❑ Therefore, execution virtualization can be implemented directly on top of the hardware by the operating system, an application and libraries (dynamically or statically) linked to an application image.
 - ❑ Modern computing systems can be expressed in terms of the reference model described in Figure 2.4.

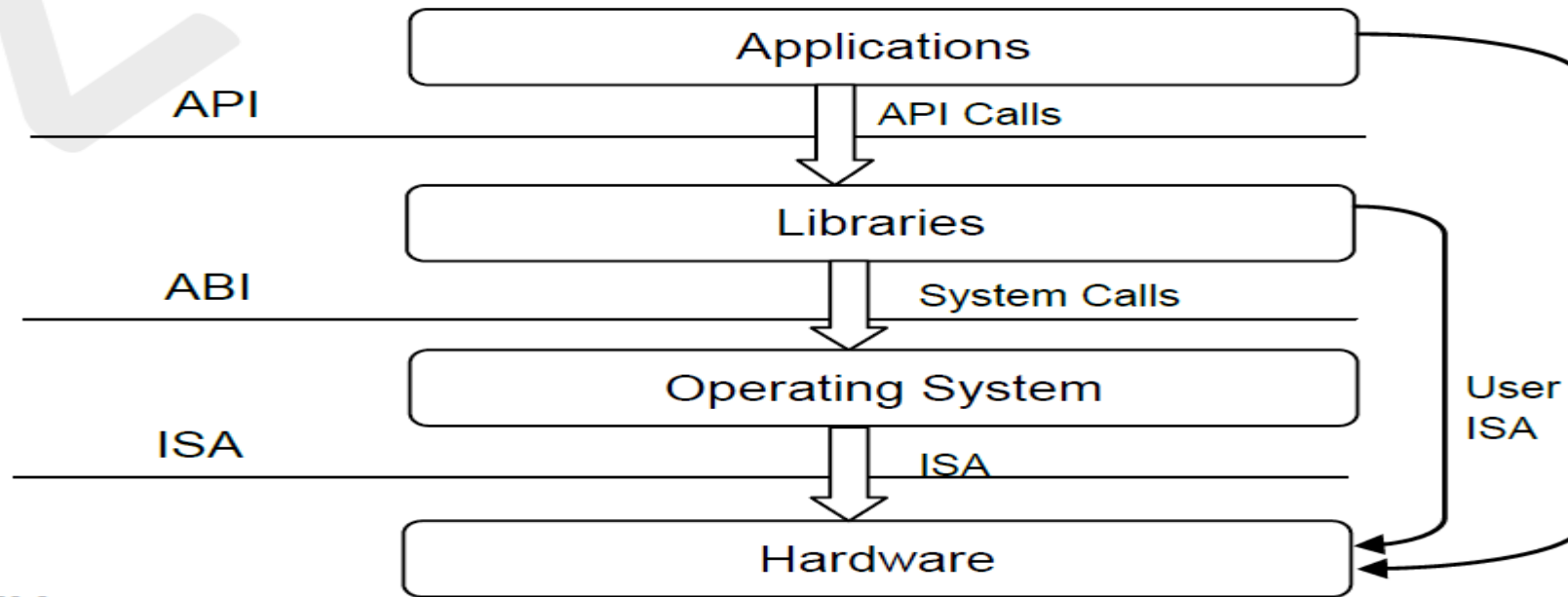


Figure 2.4 Machine reference model

-
- ❑ At the bottom layer, the model for the hardware is expressed in terms of the Instruction Set Architecture (ISA), which defines the instruction set for the processor, registers, memory and an interrupt management.
 - ❑ ISA is the interface between hardware and software.
 - ❑ ISA is important to the operating system (OS) developer (System ISA) and developers of applications that directly manage the underlying hardware (User ISA).
 - ❑ The application binary interface (ABI) separates the operating system layer from the applications and libraries, which are managed by the OS.
 - ❑ ABI covers details such as low level data types, alignment, call conventions and defines a format for executable programs.
 - ❑ System calls are defined at this level

-
- ❑ This interface allows portability of applications and libraries across operating systems that implement the same ABI.
 - ❑ The highest level of abstraction is represented by the application programming interface (API), which interfaces applications to libraries and the underlying operating system.
 - ❑ For this purpose, the instruction set exposed by the hardware has been divided into different security classes that define who can operate with them.
 - ❑ The first distinction can be made between privileged and non privileged instructions.
 - ✓ Non privileged instructions are those instructions that can be used without interfering with other tasks because they do not access shared resources.
 - ✓ This category contains all the floating, fixed-point, and arithmetic instructions.

-
- ❑ Privileged instructions are those that are executed under specific restrictions and are mostly used for sensitive operations, which expose (behavior-sensitive) or modify (control-sensitive) the privileged state.
 - ❑ Some types of architecture feature more than one class of privileged instructions and implement a finer control of how these instructions can be accessed.
 - ❑ For instance, a possible implementation features a hierarchy of privileges illustrate in the figure 2.5 in the form of ring-based security: Ring 0, Ring 1, Ring 2, and Ring 3;
 - ✓ Ring 0 is in the most privileged level and Ring 3 in the least privileged level.
 - ✓ Ring 0 is used by the kernel of the OS, rings 1 and 2 are used by the OS level services, and Ring 3 is used by the user.
 - ✓ Recent systems support only two levels, with Ring 0 for supervisor mode and Ring 3 for user mode.

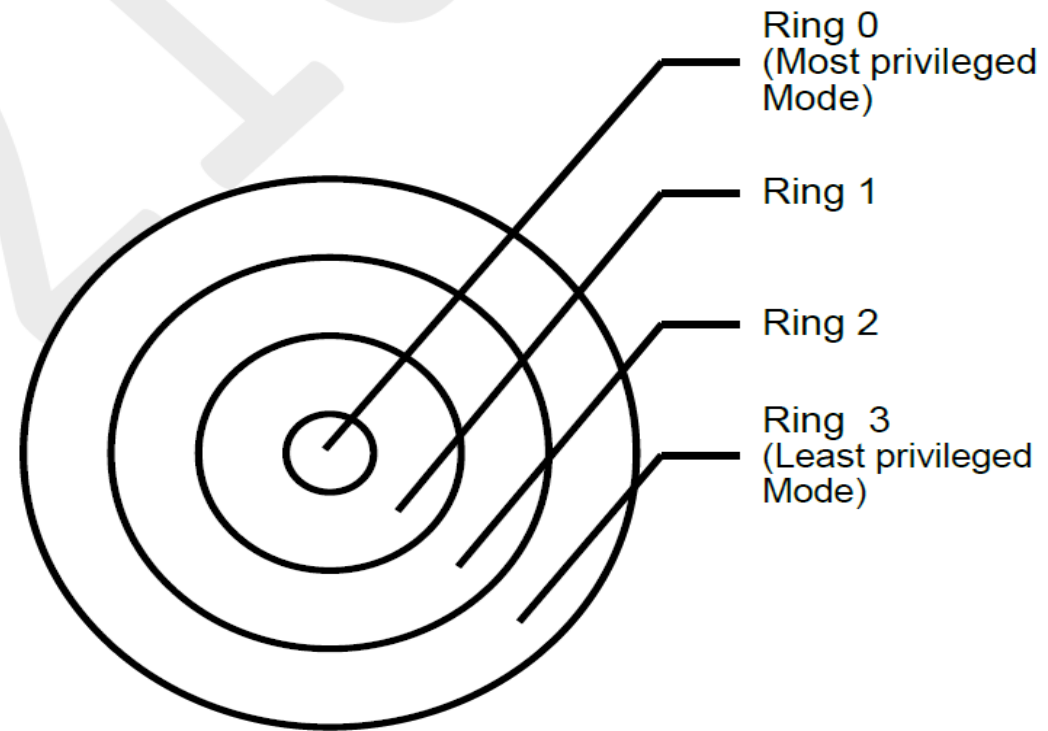


Figure 2.5 Security rings

-
- ❑ All the current systems support at least two different execution modes: supervisor mode and user mode.
 - ✓ The supervisor mode denotes an execution mode in which all the instructions (privileged and non privileged) can be executed without any restriction.
 - ✓ This mode, also called master mode or kernel mode, is generally used by the operating system (or the hypervisor) to perform sensitive operations on hardware level resources.
 - ✓ In user mode, there are restrictions to control the machine level resources.
 - ❑ The distinction between user and supervisor mode allows us to understand the role of the hypervisor and why it is called that.
 - ❑ Conceptually, the hypervisor runs above the supervisor mode and from here the prefix “hyper” is used.
 - ❑ In reality, hypervisors are run in supervisor mode and the division between privileged and non privileged instructions has posed challenges in designing virtual machine managers.

Hardware level virtualization

- ❑ Hardware level virtualization is a virtualization technique that provides an abstract execution environment in terms of computer hardware on top of which a guest operating system can be run.
- ❑ In this model, the guest is represented by the operating system, the host by the physical computer hardware, the virtual machine by its emulation and the virtual machine manager by the hypervisor.
- ❑ The hypervisor is generally a program or a combination of software and hardware that allows the abstraction of the underlying physical hardware.
- ❑ Hardware level virtualization is also called system virtualization, since it provides ISA to virtual machines, which is the representation of the hardware interface of a system.
- ❑ This is to differentiate it from process virtual machines, which expose ABI to virtual machines.
- ❑ Hypervisors is a fundamental element of hardware virtualization is the hypervisor, or virtual machine manager (VMM).

-
- ❑ It recreates a hardware environment in which guest operating systems are installed.
 - ❑ There are two major types of hypervisor: Type I and Type II. Figure 2.6 shows different type of hypervisors.
 - ✓ Type I hypervisors run directly on top of the hardware.
 - Type I hypervisor take the place of the operating systems and interact directly with the ISA interface exposed by the underlying hardware and they emulate this interface in order to allow the management of guest operating systems.
 - This type of hypervisor is also called a native virtual machine since it runs natively on hardware.
 - ✓ Type II hypervisors require the support of an operating system to provide virtualization services.
 - This means that they are programs managed by the operating system, which interact with it through the ABI and emulate the ISA of virtual hardware for guest operating systems.
 - This type of hypervisor is also called a hosted virtual machine since it is hosted within an operating system.

Hardware assisted virtualization

- ❑ Hardware assisted virtualization refers to a scenario in which the hardware provides architectural support for building a virtual machine manager able to run a guest operating system in complete isolation.
- ❑ This technique was originally introduced in the IBM System/370.
- ❑ At present, examples of hardware assisted virtualization are the extensions to the x86 architecture introduced with Intel-VT (formerly known as Vanderpool) and AMD-V (formerly known as Pacifica).
- ❑ These extensions, which differ between the two vendors, are meant to reduce the performance penalties experienced by emulating x86 hardware with hypervisors.
- ❑ Before the introduction of hardware assisted virtualization, software emulation of x86 hardware was significantly costly from the performance point of view.
- ❑ The reason for this is that by design the x86 architecture did not meet the formal requirements introduced by Popek and Goldberg and early products were using binary translation to trap some sensitive instructions and provide an emulated version.

-
- ❑ Products such as VMware Virtual Platform, introduced in 1999 by VMware, which pioneered the field of x86 virtualization, were based on this technique.
 - ❑ After 2006, Intel and AMD introduced processor extensions and a wide range of virtualization solutions took advantage of them: Kernel-based Virtual Machine (KVM), VirtualBox, Xen, VMware, Hyper-V, Sun xVM, Parallels, and others.

Full virtualization

- ❑ Full virtualization refers to the ability to run a program, most likely an operating system, directly on top of a virtual machine and without any modification, as though it were run on the raw hardware.
- ❑ To make this possible, virtual machine managers are required to provide a complete emulation of the entire underlying hardware.
- ❑ The principal advantage of full virtualization is complete isolation, which leads to enhanced security, ease of emulation of different architectures and coexistence of different systems on the same platform.
- ❑ Whereas it is a desired goal for many virtualization solutions, full virtualization poses important concerns related to performance and technical implementation.

-
- A key challenge is the interception of privileged instructions such as I/O instructions:

Since they change the state of the resources exposed by the host, they have to be contained within the virtual machine manager.

- A simple solution to achieve full virtualization is to provide a virtual environment for all the instructions, thus posing some limits on performance.

- A successful and efficient implementation of full virtualization is obtained with a combination of hardware and software, not allowing potentially harmful instructions to be executed directly on the host.

Paravirtualization

- ❑ Paravirtualization is a not transparent virtualization solution that allows implementing thin virtual machine managers.
- ❑ Paravirtualization techniques expose a software interface to the virtual machine that is slightly modified from the host and, as a consequence, guests need to be modified.
- ❑ The aim of paravirtualization is to provide the capability to demand the execution of performance critical operations directly on the host, thus preventing performance losses that would otherwise be experienced in managed execution.
- ❑ This allows a simpler implementation of virtual machine managers that have to simply transfer the execution of these operations, which were hard to virtualize, directly to the host.
- ❑ To take advantage of such an opportunity, guest operating systems need to be modified and explicitly ported by remapping the performance critical operations through the virtual machine software interface.

-
- ❑ This is possible when the source code of the operating system is available, and this is the reason that paravirtualization was mostly explored in the opensource and academic environment.
 - ❑ This technique has been successfully used by Xen for providing virtualization solutions for Linux-based operating systems specifically ported to run on Xen hypervisors.
 - ❑ Operating systems that cannot be ported can still take advantage of para virtualization by using ad hoc device drivers that remap the execution of critical instructions to the paravirtualization APIs exposed by the hypervisor.
 - ❑ Xen provides this solution for running Windows based operating systems on x86 architectures.
 - ❑ Other solutions using paravirtualization include VMWare, Parallels, and some solutions for embedded and real-time environments such as TRANGO, Wind River, and XtratuM.

Partial virtualization

- ❑ Partial virtualization provides a partial emulation of the underlying hardware, thus not allowing the complete execution of the guest operating system in complete isolation.
- ❑ Partial virtualization allows many applications to run transparently, but not all the features of the operating system can be supported as happens with full virtualization.
- ❑ An example of partial virtualization is address space virtualization used in time sharing systems; this allows multiple applications and users to run concurrently in a separate memory space, but they still share the same hardware resources (disk, processor, and network).
- ❑ Historically, partial virtualization has been an important milestone for achieving full virtualization, and it was implemented on the experimental IBM M44/44X.
- ❑ Address space virtualization is a common feature of contemporary operating systems.

Operating system level virtualization

- ❑ Operating system level virtualization offers the opportunity to create different and separated execution environments for applications that are managed concurrently.
- ❑ Differently from hardware virtualization, there is no virtual machine manager or hypervisor and the virtualization is done within a single operating system where the OS kernel allows for multiple isolated user space instances.
- ❑ The kernel is also responsible for sharing the system resources among instances and for limiting the impact of instances on each other.
- ❑ A user space instance in general contains a proper view of the file system which is completely isolated and separate IP addresses, software configurations and access to devices.
- ❑ Operating systems supporting this type of virtualization are general purpose, timeshared operating systems with the capability to provide stronger namespace and resource isolation.

-
- ❑ This virtualization technique can be considered an evolution of the chroot mechanism in Unix systems.
 - ❑ The chroot operation changes the file system root directory for a process and its children to a specific directory.
 - ❑ As a result, the process and its children cannot have access to other portions of the file system than those accessible under the new root directory.
 - ❑ Because Unix systems also expose devices as parts of the file system, by using this method it is possible to completely isolate a set of processes.
 - ❑ Following the same principle, operating system level virtualization aims to provide separated and multiple execution containers for running applications.
 - ❑ This technique is an efficient solution for server consolidation scenarios in which multiple application servers share the same technology: operating system, application server framework, and other components.
 - ❑ Examples of operating system-level virtualizations are FreeBSD Jails, IBM Logical Partition (LPAR), Solaris Zones and Containers, Parallels Virtuozzo Containers, OpenVZ, iCore Virtual Accounts, Free Virtual Private Server (FreeVPS), and others.

Programming language-level virtualization

- ❑ Programming language level virtualization is mostly used to achieve ease of deployment of applications, managed execution, portability across different platforms and operating systems.
- ❑ It consists of a virtual machine executing the byte code of a program which is the result of the compilation process.
- ❑ Compilers implemented and used this technology to produce a binary format representing the machine code for an abstract architecture.
- ❑ The characteristics of this architecture vary from implementation to implementation.
- ❑ Generally these virtual machines constitute a simplification of the underlying hardware instruction set and provide some high level instructions that map some of the features of the languages compiled for them.
- ❑ At runtime, the byte code can be either interpreted or compiled on the fly against the underlying hardware instruction set.

-
- ❑ Programming language level virtualization has a long trail in computer science history and originally was used in 1966 for the implementation of Basic Combined Programming Language (BCPL), a language for writing compilers and one of the ancestors of the C programming language.
 - ❑ Other important examples of the use of this technology have been the UCSD Pascal and Smalltalk.
 - ❑ Virtual machine programming languages become popular again with Sun's introduction of the Java platform in 1996.
 - ❑ The Java virtual machine was originally designed for the execution of programs written in the Java language, but other languages such as Python, Pascal, Groovy and Ruby were made available.
 - ❑ The ability to support multiple programming languages has been one of the key elements of the Common Language Infrastructure (CLI) which is the specification behind .NET Framework.

Application level virtualization

- ❑ Application level virtualization is a technique allowing applications to be run in runtime environments that do not natively support all the features required by such application
- ❑ In this scenario, applications are not installed in the expected runtime environment but are run as though they were.
- ❑ In general, these techniques are mostly concerned with partial file systems, libraries, and operating system component emulation.
- ❑ Such emulation is performed by a thin layer called a program or an operating system component that is in charge of executing the application.
- ❑ Emulation can also be used to execute program binaries compiled for different hardware architectures.

❑ In this case, one of the following strategies can be implemented:

❑ Interpretation: In this technique every source instruction is interpreted by an emulator for executing native ISA instructions, leading to poor performance. Interpretation has a minimal startup cost but a huge overhead, since each instruction is emulated.

❑ Binary translation: In this technique every source instruction is converted to native instructions with equivalent functions. After a block of instructions is translated, it is cached and reused.

❑ Application virtualization is a good solution in the case of missing libraries in the host operating system.

❑ In this case a replacement library can be linked with the application or library calls can be remapped to existing functions available in the host system.

❑ Another advantage is that in this case the virtual machine manager is much lighter since it provides a partial emulation of the runtime environment compared to hardware virtualization.

-
- ❑ Compared to programming level virtualization, which works across all the applications developed for that virtual machine, application level virtualization works for a specific environment.
 - ❑ It supports all the applications that run on top of a specific environment.
 - ❑ One of the most popular solutions implementing application virtualization is Wine, which is a software application allowing Unix-like operating systems to execute programs written for the Microsoft Windows platform.
 - ❑ Wine features a software application acting as a container for the guest application and a set of libraries, called Winelib, that developers can use to compile applications to be ported on Unix systems.
 - ❑ Wine takes its inspiration from a similar product from Sun, Windows Application Binary Interface (WABI) which implements the Win 16 API specifications on Solaris.
 - ❑ A similar solution for the Mac OS X environment is CrossOver, which allows running Windows applications directly on the Mac OS X operating system.
 - ❑ VMware ThinApp is another product in this area, allows capturing the setup of an installed application and packaging it into an executable image isolated from the hosting operating system.

Other types of virtualization

- ❑ Other than execution virtualization, other types of virtualization provide an abstract environment to interact with.
- ❑ These mainly cover storage, networking, and client/server interaction.

Storage virtualization

- ❑ Storage virtualization is a system administration practice that allows decoupling the physical organization of the hardware from its logical representation.
- ❑ Using this technique, users do not have to be worried about the specific location of their data, which can be identified using a logical path.
- ❑ Storage virtualization allows us to harness a wide range of storage facilities and represent them under a single logical file system.
- ❑ There are different techniques for storage virtualization, one of the most popular being network based virtualization by means of storage area networks (SANs).
- ❑ SANs use a network accessible device through a large bandwidth connection to provide storage facilities.

Network virtualization

- ❑ Network virtualization combines hardware appliances and specific software for the creation and management of a virtual network.
- ❑ Network virtualization can aggregate different physical networks into a single logical network (external network virtualization) or provide network like functionality to an operating system partition (internal network virtualization).
- ❑ The result of external network virtualization is generally a virtual LAN (VLAN).
- ❑ A VLAN is an aggregation of hosts that communicate with each other as though they were located under the same broadcasting domain.

-
- ❑ Internal network virtualization is generally applied together with hardware and operating system-level virtualization, in which the guests obtain a virtual network interface to communicate with.
 - ❑ There are several options for implementing internal network virtualization:
 - ✓ The guest can share the same network interface of the host and use Network Address Translation (NAT) to access the network;
 - ✓ The virtual machine manager can emulate, and install on the host, an additional network device, together with the driver.
 - ✓ The guest can have a private network only with the guest.

Desktop virtualization

- ❑ Desktop virtualization abstracts the desktop environment available on a personal computer in order to provide access to it using a client/server approach.
- ❑ Desktop virtualization provides the same outcome of hardware virtualization but serves a different purpose.
- ❑ Similarly to hardware virtualization, desktop virtualization makes accessible a different system as though it were natively installed on the host but this system is remotely stored on a different host and accessed through a network connection.
- ❑ Moreover, desktop virtualization addresses the problem of making the same desktop environment accessible from everywhere.
- ❑ Although the term desktop virtualization strictly refers to the ability to remotely access a desktop environment, generally the desktop environment is stored in a remote server or a data center that provides a high availability infrastructure and ensures the accessibility and persistence of the data.

-
- ❑ In this scenario, an infrastructure supporting hardware virtualization is fundamental to provide access to multiple desktop environments hosted on the same server.
 - ❑ A specific desktop environment is stored in a virtual machine image that is loaded and started on demand when a client connects to the desktop environment.
 - ❑ This is a typical cloud computing scenario in which the user leverages the virtual infrastructure for performing the daily tasks on his computer.
 - ❑ The advantages of desktop virtualization are high availability, persistence, accessibility, and ease of management.
 - ❑ The basic services for remotely accessing a desktop environment are implemented in software components such as Windows Remote Services, VNC, and X Server.
 - ❑ Infrastructures for desktop virtualization based on cloud computing solutions include Sun Virtual Desktop Infrastructure (VDI), Parallels Virtual Desktop Infrastructure (VDI), Citrix XenDesktop, and others.

Application server virtualization

- ❑ Application server virtualization abstracts a collection of application servers that provide the same services as a single virtual application server by using load balancing strategies and providing a high availability infrastructure for the services hosted in the application server.
- ❑ This is a particular form of virtualization and serves the same purpose of storage virtualization by providing a better quality of service rather than emulating a different environment.

Implementation Levels of Virtualization

- ❑ Virtualization is a computer architecture technology by which multiple virtual machines (VMs) are multiplexed in the same hardware machine.
- ❑ The purpose of a VM is to enhance resource sharing by many users and improve computer performance in terms of resource utilization and application flexibility.
- ❑ Hardware resources (CPU, memory, I/O devices) or software resources (operating system and software libraries) can be virtualized in various functional layers.
- ❑ The idea is to separate the hardware from the software to yield better system efficiency. For example, computer users gained access to much enlarged memory space when the concept of virtual memory was introduced.
- ❑ Similarly, virtualization techniques can be applied to enhance the use of compute engines, networks and storage.
- ❑ With sufficient storage, any computer platform can be installed in another host computer, even if they use processors with different instruction sets and run with distinct operating systems on the same hardware.

Levels of virtualization implementation

- ❑ A traditional computer runs with a host operating system specially tailored for its hardware architecture, as shown in Figure 2.7(a).
- ❑ After virtualization, different user applications managed by their own operating systems (guest OS) can run on the same hardware, independent of the host OS. This is often done by adding additional software, called a virtualization layer as shown in Figure 2.7(b).
- ❑ This virtualization layer is known as hypervisor or virtual machine monitor (VMM). The VMs are shown in the upper boxes, where applications run with their own guest OS over the virtualized CPU, memory, and I/O resources.
- ❑ The main function of the software layer for virtualization is to virtualize the physical hardware of a host machine into virtual resources to be used by the VMs, exclusively. This can be implemented at various operational levels, as we will discuss shortly.
- ❑ The virtualization software creates the abstraction of VMs by interposing a virtualization layer at various levels of a computer system.
- ❑ Common virtualization layers include the instruction set architecture (ISA) level, hardware level, operating system level, library support level, and application level

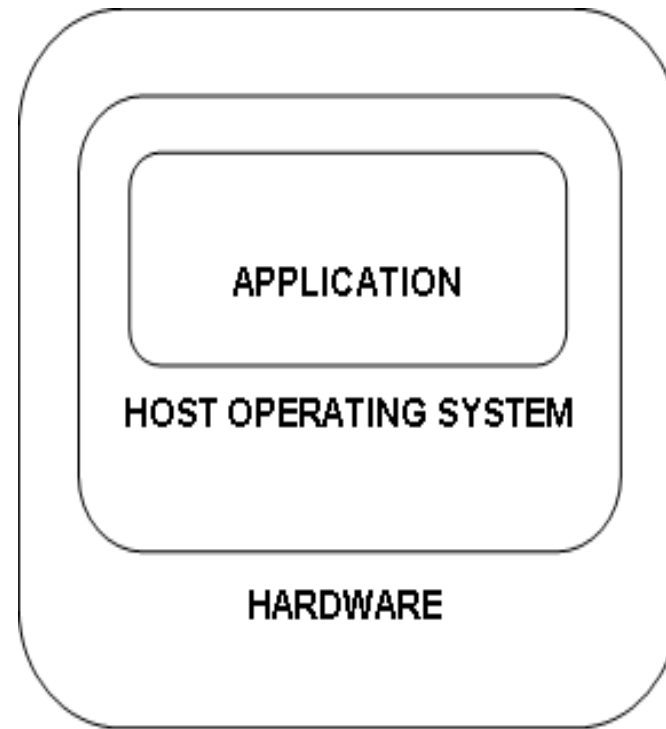


Figure 2.7 (a) Traditional Computer

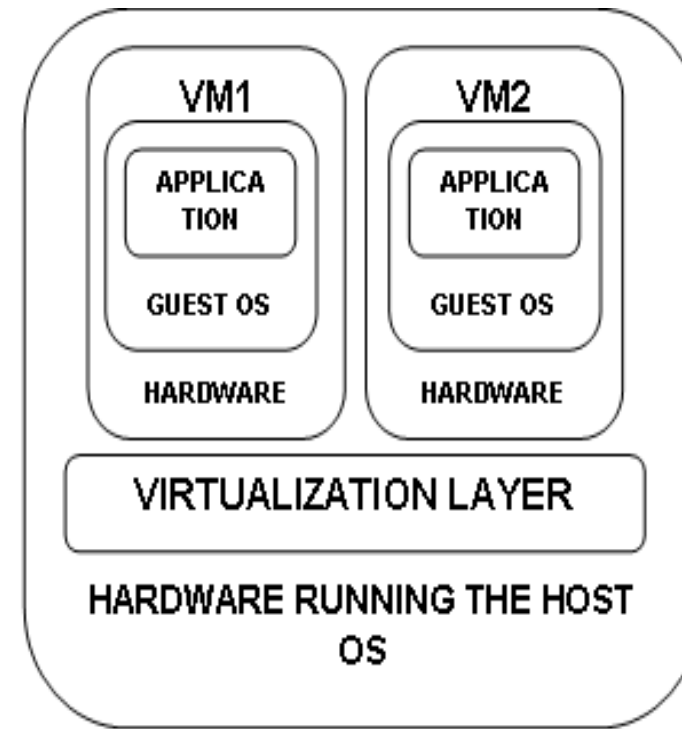


Figure (b) After Virtualization

Instruction set architecture level

- ❑ At the ISA level, virtualization is performed by emulating a given ISA by the ISA of the host machine. For example, MIPS binary code can run on an x86-based host machine with the help of ISA emulation.
- ❑ With this approach, it is possible to run a large amount of legacy binary code written for various processors on any given new hardware host machine. Instruction set emulation leads to virtual ISAs created on any hardware machine.
- ❑ The basic emulation method is through code interpretation. An interpreter program interprets the source instructions to target instructions one by one.
- ❑ One source instruction may require tens or hundreds of native target instructions to perform its function. This process is relatively slow.
- ❑ For better performance, dynamic binary translation is desired. This approach translates basic blocks of dynamic source instructions to target instructions.

-
- The basic blocks can also be extended to program traces or super blocks to increase translation efficiency.
 - Instruction set emulation requires binary translation and optimization

APPLICATION LEVEL

JVM / .NET CLR / Panot

LIBRARY (USER-LEVEL API) LEVEL

WINE/ WABI/ LXRUN / VISUAL MAINWIN / VCUDA

OPERATING SYSTEM LEVEL

JAIL / VIRTUAL ENV / ENSIM'S VPS / FVM

HARDWARE ABSTRACTION LAYER (HAL) LEVEL

VMWARE / VIRTUAL PC / XEN / L4

INSTRUCTION SET ARCHITECTURE (ISA) LEVEL

BOCHS / CRUSOE / QEMU / BIRD / DYNAMO

Figure 2.8 Virtualization ranging from hardware to applications in five abstraction levels

Hardware abstraction level

- ❑ Hardware-level virtualization is performed right on top of the bare hardware. On the one hand, this approach generates a virtual hardware environment for a VM. On the other hand, the process manages the underlying hardware through virtualization.
- ❑ The idea is to virtualize a computer's resources, such as its processors, memory, and I/O devices.
- ❑ The intention is to upgrade the hardware utilization rate by multiple users concurrently.
- ❑ The idea was implemented in the IBM VM/370 in the 1960s. Xen hypervisor has been applied to virtualize x86-based machines to run Linux or other guest OS applications.

Operating system level

- ❑ This refers to an abstraction layer between traditional OS and user applications. OS-level virtualization creates isolated containers on a single physical server and the OS instances to utilize the hardware and software in data centers.
- ❑ The containers behave like real servers. OS-level virtualization is commonly used in creating virtual hosting environments to allocate hardware resources among a large number of mutually distrusting users.

-
- ❑ It is also used, to a lesser extent, in consolidating server hardware by moving services on separate hosts into containers or VMs on one server.
 - ❑ Operating system virtualization inserts a virtualization layer inside an operating system to partition a machine's physical resources. It enables multiple isolated VMs within a single operating system kernel.

-
- ❑ This kind of VM is often called a virtual execution environment (VE), Virtual Private System (VPS), or simply container.
 - ❑ Compared to hardware-level virtualization, the benefits of OS extensions are twofold: VMs at the operating system level have minimal startup/shutdown costs, low resource requirements, and high scalability
 - ❑ For an OS-level VM, it is possible for a VM and its host environment to synchronize state changes when necessary.
 - ❑ These benefits can be achieved via two mechanisms of OS-level virtualization:
 - ❑ All OS-level VMs on the same physical machine share a single operating system kernel
 - ✓ The virtualization layer can be designed in a way that allows processes in VMs to access as many resources of the host machine as possible, but never to modify them.

❑ Virtualization Support for the Linux Platform

- ✓ OpenVZ is an OS-level tool designed to support Linux platforms to create virtual environments for running VMs under different guest Operating Systems.
- ✓ OpenVZ is an open source container-based virtualization solution built on Linux. To support virtualization and isolation of various subsystems, limited resource management, and check pointing, OpenVZ modifies the Linux kernel.

Library support level

- ❑ Most applications use APIs exported by user level libraries rather than using lengthy system calls by the OS. Since most systems provide well-documented APIs, such an interface becomes another candidate for virtualization
- ❑ Virtualization with library interfaces is possible by controlling the communication link between applications and the rest of a system through API hooks.
- ❑ The software tool WINE has implemented this approach to support Windows applications on top of UNIX hosts. Another example is the vCUDA which allows applications executing within VMs to leverage GPU hardware acceleration.
- ❑ Library level virtualization is also known as user-level Application Binary Interface (ABI) or API emulation.
- ❑ This type of virtualization can create execution environments for running alien programs on a platform rather than creating a VM to run the entire operating system.
- ❑ API call interception and remapping are the key functions performed.

-
- ❑ Lxrun is really a system call emulator that enables Linux applications written for x86 hosts to run on UNIX systems.
 - ❑ Similarly, Wine offers library support for virtualizing x86 processors to run Windows applications on UNIX hosts.
 - ❑ Visual MainWin offers a compiler support system to develop Windows applications using Visual Studio to run on some UNIX hosts.
 - ❑ The vCUDA for Virtualization of General-Purpose GPUs

-
- ❑ CUDA is a programming model and library for general-purpose GPUs. It leverages the high performance of GPUs to run compute-intensive applications on host operating systems. However, it is difficult to run CUDA applications on hardware-level VMs directly.
 - ❑ vCUDA virtualizes the CUDA library and can be installed on guest OSes. When CUDA applications run on a guest OS and issue a call to the CUDA API, vCUDA intercepts the call and redirects it to the CUDA API running on the host OS.

User application level

- Virtualization at the application level virtualizes an application as a VM.
- On a traditional OS, an application often runs as a process. Therefore, application-level virtualization is also known as process level virtualization. The most popular approach is to deploy high level language (HLL) VMs.
- In this scenario, the virtualization layer sits as an application program on top of the operating system, and the layer exports an abstraction of a VM that can run programs written and compiled to a particular abstract machine definition.
- Any program written in the HLL and compiled for this VM will be able to run on it. The Microsoft .NET CLR and Java Virtual Machine (JVM) are two good examples of this class of VM.
- Other forms of application-level virtualization are known as application isolation, application sandboxing, or application streaming.

-
- The process involves wrapping the application in a layer that is isolated from the host OS and other applications.
 - The result is an application that is much easier to distribute and remove from user workstations.
 - An example is the LANDesk application virtualization platform which deploys software applications as self contained, executable files in an isolated environment without requiring installation, system modifications or elevated security privileges.

Virtualization Structures, Tools and Mechanisms

- In general, there are three typical classes of VM architecture.
- The virtualization layer is responsible for converting portions of the real hardware into virtual hardware.
- Therefore, different operating systems such as Linux and Windows can run on the same physical machine, simultaneously.
- Depending on the position of the virtualization layer, there are several classes of VM architectures, namely the hypervisor architecture, paravirtualization and host based virtualization.
- The hypervisor is also known as the VMM (Virtual Machine Monitor). They both perform the same virtualization operations.

Hypervisor and Xen architecture

- The hypervisor supports hardware level virtualization on bare metal devices like CPU, memory, disk and network interfaces.
- The hypervisor software sits directly between the physical hardware and its OS. This virtualization layer is referred to as either the VMM or the hypervisor.
- The hypervisor provides hyper calls for the guest OSes and applications.
- Depending on the functionality, a hypervisor can assume micro kernel architecture like the Microsoft Hyper-V.
- It can assume monolithic hypervisor architecture like the VMware ESX for server virtualization.
- A micro kernel hypervisor includes only the basic and unchanging functions (such as physical memory management and processor scheduling).
- The device drivers and other changeable components are outside the hypervisor.
- A monolithic hypervisor implements all the aforementioned functions, including those of the device drivers. Therefore, the size of the hypervisor code of a micro-kernel hypervisor is smaller than that of a monolithic hypervisor.
- Essentially, a hypervisor must be able to convert physical devices into virtual resources dedicated for the deployed VM to use.

Xen architecture

- Xen is an open source hypervisor program developed by Cambridge University.
- Xen is a microkernel hypervisor, which separates the policy from the mechanism.
- The Xen hypervisor implements all the mechanisms, leaving the policy to be handled by Domain 0. Figure 2.9 shows architecture of Xen hypervisor.
- Xen does not include any device drivers natively. It just provides a mechanism by which a guest OS can have direct access to the physical devices.
- As a result, the size of the Xen hypervisor is kept rather small.
- Xen provides a virtual environment located between the hardware and the OS.

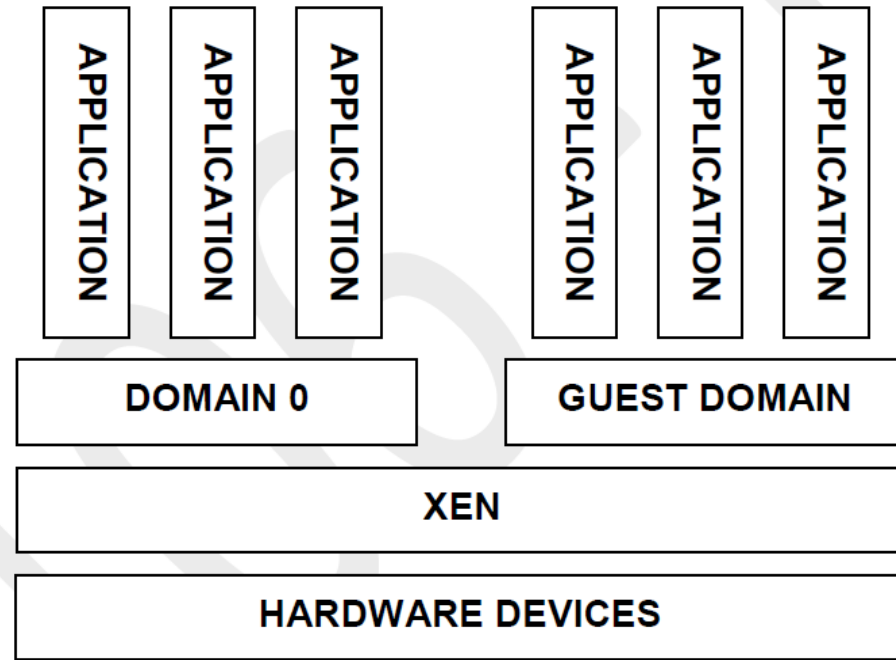


Figure 2.9 Xen domain 0 for control and I/O & guest domain for user applications.

-
- The core components of a Xen system are the hypervisor, kernel, and applications.
 - The organization of the three components is important.
 - Like other virtualization systems, many guest OSes can run on top of the hypervisor.
 - However, not all guest OSes are created equal, and one in particular controls the others.
 - The core components of a Xen system are the hypervisor, kernel, and applications.
 - The organization of the three components is important.
 - Like other virtualization systems, many guest OSes can run on top of the hypervisor.
 - However, not all guest OSes are created equal, and one in particular controls the others.

-
- The guest OS, which has control ability, is called Domain 0, and the others are called Domain U.
 - Domain 0 is a privileged guest OS of Xen. It is first loaded when Xen boots without any file system drivers being available.
 - Domain 0 is designed to access hardware directly and manage devices. Therefore, one of the responsibilities of Domain 0 is to allocate and map hardware resources for the guest domains (the Domain U domains).
 - For example, Xen is based on Linux and its security level is C2. Its management VM is named Domain 0 which has the privilege to manage other VMs implemented on the same host.
 - If Domain 0 is compromised, the hacker can control the entire system. So, in the VM system, security policies are needed to improve the security of Domain 0.
 - Domain 0, behaving as a VMM, allows users to create, copy, save, read, modify, share, migrate and roll back VMs as easily as manipulating a file, which flexibly provides tremendous benefits for users.

Binary translation with full virtualization

- Depending on implementation technologies, hardware virtualization can be classified into two categories: full virtualization and host based virtualization.
- Full virtualization does not need to modify the host OS. It relies on binary translation to trap and to virtualize the execution of certain sensitive, non virtualizable instructions.
- The guest OSes and their applications consist of noncritical and critical instructions.
- In a host-based system, both a host OS and a guest OS are used.
- A virtualization software layer is built between the host OS and guest OS.
- With full virtualization, noncritical instructions run on the hardware directly while critical instructions are discovered and replaced with traps into the VMM to be emulated by software.
- Both the hypervisor and VMM approaches are considered full virtualization.
- The VMM scans the instruction stream and identifies the privileged, control and behavior sensitive instructions. When these instructions are identified, they are trapped into the VMM, which emulates the behavior of these instructions.

-
- The method used in this emulation is called binary translation.
 - Full virtualization combines binary translation and direct execution.
 - An alternative VM architecture is to install a virtualization layer on top of the host OS.
 - This host OS is still responsible for managing the hardware.
 - The guest OSes are installed and run on top of the virtualization layer. Dedicated applications may run on the VMs. Certainly, some other applications can also run with the host OS directly.
 - Host based architecture has some distinct advantages, as enumerated next.
 - First, the user can install this VM architecture without modifying the host OS.
 - Second, the host-based approach appeals to many host machine configurations.

Paravirtualization with compiler support

- When x86 processor is virtualized, a virtualization layer is inserted between the hardware and the OS.
- According to the x86 ring definitions, the virtualization layer should also be installed at Ring 0. Different instructions at Ring 0 may cause some problems.
- Although paravirtualization reduces the overhead, it has incurred other problems.
 - First, its compatibility and portability may be in doubt, because it must support the unmodified OS as well.
 - Second, the cost of maintaining paravirtualized OSes is high, because they may require deep OS kernel modifications.
 - Finally, the performance advantage of paravirtualization varies greatly due to workload variations.
- Compared with full virtualization, paravirtualization is relatively easy and more practical. The main problem in full virtualization is its low performance in binary translation.

-
- KVM is a Linux paravirtualization system. It is a part of the Linux version 2.6.20 kernel.
 - In KVM, Memory management and scheduling activities are carried out by the existing Linux kernel.
 - The KVM does the rest, which makes it simpler than the hypervisor that controls the entire machine.
 - KVM is a hardware assisted and paravirtualization tool, which improves performance and supports unmodified guest OSes such as Windows, Linux, Solaris, and other UNIX variants.
 - Unlike the full virtualization architecture which intercepts and emulates privileged and sensitive instructions at runtime, paravirtualization handles these instructions at compile time.

-
- The guest OS kernel is modified to replace the privileged and sensitive instructions with hypercalls to the hypervisor or VMM. Xen assumes such paravirtualization architecture.
 - The guest OS running in a guest domain may run at Ring 1 instead of at Ring 0. This implies that the guest OS may not be able to execute some privileged and sensitive instructions. The privileged instructions are implemented by hypercalls to the hypervisor.

Virtualization of CPU, Memory and I/O Devices

- To support virtualization, processors such as the x86 employ a special running mode and instructions known as hardware assisted virtualization.
- For the x86 architecture, Intel and AMD have proprietary technologies for hardware assisted virtualization.

Hardware support for virtualization

- Modern operating systems and processors permit multiple processes to run simultaneously. If there is no protection mechanism in a processor, all instructions from different processes will access the hardware directly and cause a system crash.
- All processors have at least two modes, user mode and supervisor mode, to ensure controlled access of critical hardware.
- Instructions running in supervisor mode are called privileged instructions. Other instructions are unprivileged instructions.
- In a virtualized environment, it is more difficult to make OSes and applications run correctly because there are more layers in the machine stack.
- At the time of this writing, many hardware virtualization products were available.
- The VMware Workstation is a VM software suite for x86 and x86-64 computers.

-
- This software suite allows users to set up multiple x86 and x86-64 virtual computers and to use one or more of these VMs simultaneously with the host operating system.
 - The VMware Workstation assumes the host-based virtualization.
 - Xen is a hypervisor for use in IA-32, x86-64, Itanium and PowerPC 970 hosts.
 - One or more guest OS can run on top of the hypervisor.
 - KVM is a Linux kernel virtualization infrastructure.
 - KVM can support hardware assisted virtualization and paravirtualization by using the Intel VT-x or AMD-v and VirtIO framework, respectively.
 - The VirtIO framework includes a paravirtual Ethernet card, a disk I/O controller and a balloon device for adjusting guest memory usage and a VGA graphics interface using VMware drivers.

CPU virtualization

- A VM is a duplicate of an existing computer system in which a majority of the VM instructions are executed on the host processor in native mode.
- The unprivileged instructions of VMs run directly on the host machine for higher efficiency.
- The critical instructions are divided into three categories: privileged instructions, control sensitive instructions, and behavior sensitive instructions.
- Privileged instructions execute in a privileged mode and will be trapped if executed outside this mode.

-
- Control sensitive instructions attempt to change the configuration of resources used.
 - Behavior sensitive instructions have different behaviors depending on the configuration of resources, including the load and store operations over the virtual memory.
 - CPU architecture is virtualizable if it supports the ability to run the VM's privileged and unprivileged instructions in the CPU's user mode while the VMM runs in supervisor mode.
 - The privileged instructions including control and behavior sensitive instructions of a VM are executed; they are trapped in the VMM.
 - RISC CPU architectures can be naturally virtualized because all control and behavior sensitive instructions are privileged instructions.
 - The x86 CPU architectures are not primarily designed to support virtualization.

Hardware-assisted CPU virtualization

- This technique attempts to simplify virtualization because full or paravirtualization is complicated.
- Intel and AMD add an additional mode called privilege mode level (some people call it Ring-1) to x86 processors.
- Therefore, operating systems can still run at Ring 0 and hypervisor can run at Ring 1.
- All the privileged and sensitive instructions are trapped in the hypervisor automatically.
- This technique removes the difficulty of implementing binary translation of full virtualization.
- It also lets the operating system run in VMs without modification.

Memory virtualization

- Virtual memory virtualization is similar to the virtual memory support provided by modern operating systems.
- In a traditional execution environment, the operating system maintains mappings of virtual memory to machine memory using page tables, which is a one stage mapping from virtual memory to machine memory.
- All modern x86 CPUs include a memory management unit (MMU) and a translation lookaside buffer (TLB) to optimize virtual memory performance.
- However, in a virtual execution environment, virtual memory virtualization involves sharing the physical system memory in RAM and dynamically allocating it to the physical memory of the VMs.
- That means a two stage mapping process should be maintained by the guest OS and the VMM, respectively: virtual memory to physical memory and physical memory to machine memory.
- MMU virtualization should be supported, which is transparent to the guest OS.
- The guest OS continues to control the mapping of virtual addresses to the physical memory addresses of VMs.
- But the guest OS cannot directly access the actual machine memory.
- The VMM is responsible for mapping the guest physical memory to the actual machine memory.

-
- Since each page table of the guest OSes has a separate page table in the VMM corresponding to it, the VMM page table is called the shadow page table.
 - Nested page tables add another layer of indirection to virtual memory.
 - The MMU already handles virtual-to-physical translations as defined by the OS. Then the physical memory addresses are translated to machine addresses using another set of page tables defined by the hypervisor.
 - VMware uses shadow page tables to perform virtual-memory-to-machine-memory address translation.
 - Processors use TLB hardware to map the virtual memory directly to the machine memory to avoid the two levels of translation on every access.
 - When the guest OS changes the virtual memory to a physical memory mapping, the VMM updates the shadow page tables to enable a direct lookup.
 - The AMD Barcelona processor has featured hardware assisted memory virtualization since 2007.
 - It provides hardware assistance to the two stage address translation in a virtual execution environment by using a technology called nested paging.

I/O virtualization

- I/O virtualization involves managing the routing of I/O requests between virtual devices and the shared physical hardware.
- There are three ways to implement I/O virtualization: full device emulation, paravirtualization, and direct I/O.
- Full device emulation is the first approach for I/O virtualization. Generally, this approach emulates well known and real world devices.
- All the functions of a device or bus infrastructure, such as device enumeration, identification, interrupts, and DMA are replicated in software.
- This software is located in the VMM and acts as a virtual device.
- The I/O access requests of the guest OS are trapped in the VMM which interacts with the I/O devices.
- A single hardware device can be shared by multiple VMs that run concurrently. However, software emulation runs much slower than the hardware it emulates.

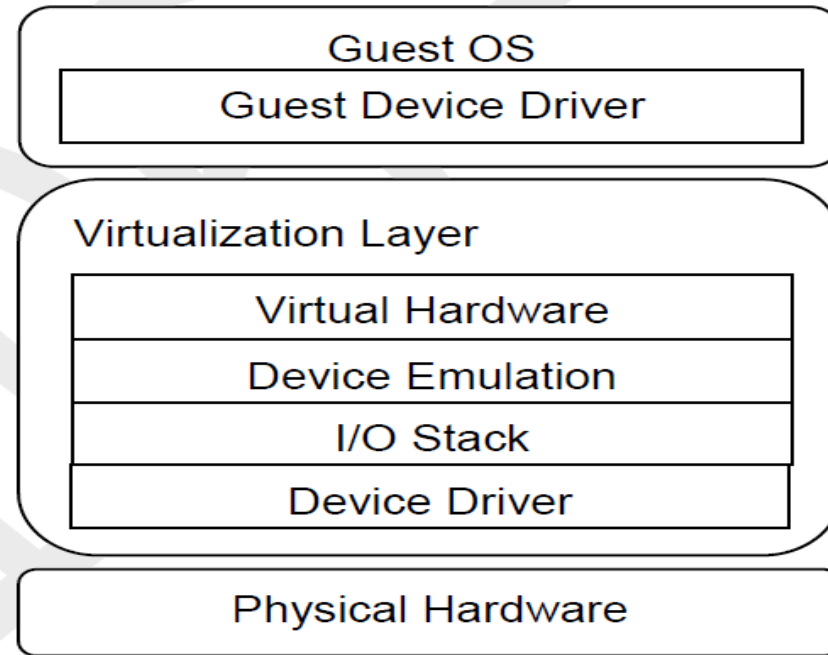


Figure 2.10 Device emulation for I/O Virtualization

-
- The paravirtualization method of I/O virtualization is typically used in Xen. It is also known as the split driver model consisting of a frontend driver and a backend driver.
 - The frontend driver is running in Domain U and the backend driver is running in Domain 0. They interact with each other via a block of shared memory.
 - The frontend driver manages the I/O requests of the guest OSes and the backend driver is responsible for managing the real I/O devices and multiplexing the I/O data of different VMs.
 - Para I/O-virtualization achieves better device performance than full device emulation, it comes with a higher CPU overhead.
 - Direct I/O virtualization lets the VM access devices directly. It can achieve close to native performance without high CPU costs.
 - However, current direct I/O virtualization implementations focus on networking for mainframes. There are a lot of challenges for commodity hardware devices.

-
- For example, when a physical device is reclaimed (required by workload migration) for later reassignment, it may have been set to an arbitrary state (e.g., DMA to some arbitrary memory locations) that can function incorrectly or even crash the whole system.
 - Since software based I/O virtualization requires a very high overhead of device emulation, hardware-assisted I/O virtualization is critical.
 - Intel VT-d supports the remapping of I/O DMA transfers and device generated interrupts. The architecture of VT-d provides the flexibility to support multiple usage models that may run unmodified, special-purpose, or “virtualization-aware” guest OSes.
 - Another way to help I/O virtualization is via self virtualized I/O (SV-IO).
 - The key idea of SV-IO is to harness the rich resources of a multicore processor. All tasks associated with virtualizing an I/O device are encapsulated in SV-IO.
 - It provides virtual devices and an associated access API to VMs and a management API to the VMM.
 - SV-IO defines one virtual interface (VIF) for every kind of virtualized I/O device, such as virtual network interfaces, virtual block devices (disk), virtual camera devices,

Virtualization in multi-core processors

- Virtualizing a multi-core processor is relatively more complicated than virtualizing a uniprocessor.
- Multi-core virtualization has raised some new challenges to computer architects, compiler constructors, system designers, and application programmers.
- There are mainly two difficulties: Application programs must be parallelized to use all cores fully, and software must explicitly assign tasks to the cores, which is a very complex problem.
 - The first challenge, new programming models, languages, and libraries are needed to make parallel programming easier.
 - The second challenge has spawned research involving scheduling algorithms and resource management policies.
- Dynamic heterogeneity is emerging to mix the fat CPU core and thin GPU cores on the same chip, which further complicates the multi core or many core resource management.
- The dynamic heterogeneity of hardware infrastructure mainly comes from less reliable transistors and increased complexity in using the transistors.