Distributed Computing?

Distributed computing refers to a system where processing and data storage is distributed across multiple devices or systems, rather than being handled by a single central device. In a distributed system, each device or system has its own processing capabilities and may also store and manage its own data. These devices or systems work together to perform tasks and share resources, with no single device serving as the central hub.

One example of a distributed computing system is a cloud computing system, where resources such as computing power, storage, and networking are delivered over the Internet and accessed on demand. In this type of system, users can access and use shared resources through a web browser or other client software.

Components

There are several key components of a Distributed Computing System

- **Devices or Systems:** The devices or systems in a distributed system have their own processing capabilities and may also store and manage their own data.
- **Network:** The network connects the devices or systems in the distributed system, allowing them to communicate and exchange data.
- **Resource Management:** Distributed systems often have some type of resource management system in place to allocate and manage shared resources such as computing power, storage, and networking.

The architecture of a Distributed Computing System is typically a Peer-to-Peer Architecture, where devices or systems can act as both clients and servers and communicate directly with each other.

Characteristics

There are several characteristics that define a Distributed Computing System

- **Multiple Devices or Systems:** Processing and data storage is distributed across multiple devices or systems.
- **Peer-to-Peer Architecture:** Devices or systems in a distributed system can act as both clients and servers, as they can both request and provide services to other devices or systems in the network.
- **Shared Resources:** Resources such as computing power, storage, and networking are shared among the devices or systems in the network.
- **Horizontal Scaling:** Scaling a distributed computing system typically involves adding more devices or systems to the network to increase processing and storage capacity. This can be done through hardware upgrades or by adding additional devices or systems to the network..

Advantages and Disadvantages

Advantages of the Distributed Computing System are:

- **Scalability:** Distributed systems are generally more scalable than centralized systems, as they can easily add new devices or systems to the network to increase processing and storage capacity.
- **Reliability:** Distributed systems are often more reliable than centralized systems, as they can continue to operate even if one device or system fails.
- **Flexibility:** Distributed systems are generally more flexible than centralized systems, as they can be configured and reconfigured more easily to meet changing computing needs.

There are a few limitations to Distributed Computing System

- **Complexity:** Distributed systems can be more complex than centralized systems, as they involve multiple devices or systems that need to be coordinated and managed.
- **Security:** It can be more challenging to secure a distributed system, as security measures must be implemented on each device or system to ensure the security of the entire system.
- **Performance:** Distributed systems may not offer the same level of performance as centralized systems, as processing and data storage is distributed across multiple devices or systems.

Applications

Distributed Computing Systems have a number of applications, including:

Introduction to Parallel Computing

Before taking a toll on Parallel Computing, first, let's take a look at the background of computations of computer software and why it failed for the modern era.

Computer software was written conventionally for serial computing. This meant that to solve a problem, an algorithm divides the problem into smaller instructions. These discrete instructions are then executed on the Central Processing Unit of a computer one by one. Only after one instruction is finished, next one starts.

A real-life example of this would be people standing in a queue waiting for a movie ticket and there is only a cashier. The cashier is giving tickets one by one to the persons. The complexity of this situation increases when there are 2 queues and only one cashier.

So, in short, Serial Computing is following:

1. In this, a problem statement is broken into discrete instructions.
2. Then the instructions are executed one by one.
3. Only one instruction is executed at any moment of time.

Look at point 3. This was causing a huge problem in the computing industry as only one instruction was getting executed at any moment of time. This was a huge waste of hardware resources as only one part of the hardware will be running for particular instruction and of time. As problem statements were getting heavier and bulkier, so does the amount of time in execution of those statements. Examples of processors are Pentium 3 and Pentium 4.

Now let's come back to our real-life problem. We could definitely say that complexity will decrease when there are 2 queues and 2 cashiers giving tickets to 2 persons simultaneously. This is an example of Parallel Computing.

Parallel Computing :
It is the use of multiple processing elements simultaneously for solving any problem. Problems are broken down into instructions and are solved concurrently as each resource that has been applied to work is working at the same time.

Advantages of Parallel Computing over Serial Computing are as follows:

1. It saves time and money as many resources working together will reduce the time and cut potential costs.
2. It can be impractical to solve larger problems on Serial Computing.
3. It can take advantage of non-local resources when the local resources are finite.
4. Serial Computing 'wastes' the potential computing power, thus Parallel Computing makes better work of the hardware.

Types of Parallelism:

1. **Bit-level parallelism –**
   It is the form of parallel computing which is based on the increasing processor's size. It reduces the number of instructions that the system must execute in order to perform a task on large-sized data.
   Example: Consider a scenario where an 8-bit processor must compute the sum of two 16-bit integers. It must first sum up the 8 lower-order bits, then add the 8 higher-order bits, thus requiring two instructions to perform the operation. A 16-bit processor can perform the operation with just one instruction.
2. **Instruction-level parallelism –**
   A processor can only address less than one instruction for each clock cycle phase. These instructions can be re-ordered and grouped which are later on executed concurrently without affecting the result of the program. This is called instruction-level parallelism.
3. **Task Parallelism –**
   Task parallelism employs the decomposition of a task into subtasks and then allocating each of the subtasks for execution. The processors perform the execution of sub-tasks concurrently.
4. **Data-level parallelism (DLP) –**

Instructions from a single stream operate concurrently on several data – Limited by non-regular data manipulation patterns and by memory bandwidth

Why parallel computing?

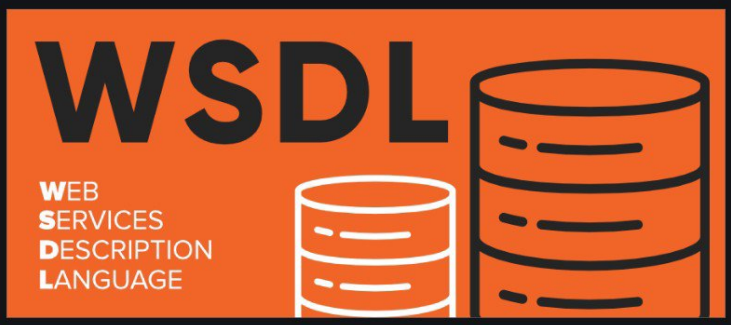
- The whole real-world runs in dynamic nature i.e. many things happen at a certain time but at different places concurrently. This data is extensively huge to manage.
- Real-world data needs more dynamic simulation and modeling, and for achieving the same, parallel computing is the key.
- Parallel computing provides concurrency and saves time and money.
- Complex, large datasets, and their management can be organized only and only using parallel computing's approach.
- Ensures the effective utilization of the resources. The hardware is guaranteed to be used effectively whereas in serial computation only some part of the hardware was used and the rest rendered idle.
- Also, it is impractical to implement real-time systems using serial computing.

Applications of Parallel Computing:

- Databases and Data mining

WSDL Full Form

WSDL stands for **Web Services Description Language**. It was developed jointly by IBM and Microsoft and recommended on June 26' 2007 by the W3C. Written in XML, it is used in describing web services. These descriptions include service location and methods. It works in coordination with SOAP and UDDI in order to provide web services, i.e SOAP is used to call web services that are listed in WSDL. Generally, a typical WSDL contains information about definition, datatypes, messages, service, bindings, targetNamespace, and port type. Prerequisites for learning about WSDL is basic XML schema and namespaces.



Characteristics

- It specifies the operations that will be performed by the web services and how these services must be accessed.
- It is based on the XML protocol used for exchanging information in distributed systems.
- It describes the specifications to be met for interfacing with XML oriented services.
- WSDL works combinedly with SOAP and UDDI.

Advantages

- It provides a systematic approach to defining web services.
- Used to reduced the total LOC which is must to access the web services.
- It can be updated dynamically which allows the users to seamlessly upgrade to new patterns.

Disadvantages

- Single mode (one-way) messaging is prohibited.
- It cannot include more than one file i.e cannot have more than one <wsdl:include> element.
- It does not support output mapping.

Last Updated : 20 Sep, 2021

Previous
RAID Full Form

Next
CDMA Full Form

Basics of SOAP – Simple Object Access Protocol

Introduction:
Simple Object Access Protocol(SOAP) is a network protocol for exchanging structured data between nodes. It uses XML format to transfer messages. It works on top of application layer protocols like HTML and SMTP for notations and transmission. SOAP allows processes to communicate throughout platforms, languages and operating systems, since protocols like HTTP are already installed on all platforms. SOAP was designed by Bob Atkinson, Don Box, Dave Winer, and Mohsen Al-Ghosein at Microsoft in 1998. SOAP was maintained by the XML Protocol Working Group of the World Wide Web Consortium until 2009.

Message Format:

SOAP message transmits some basic information as given below

- Information about message structure and instructions on processing it.
- Encoding instructions for application defined data types.
- Information about Remote Procedure Calls and their responses.

The message in XML format contains three parts

1. **Envelope:**
   It specifies that the XML message is a SOAP message. A SOAP message can be defined as an XML document containing header and body encapsulated in the envelope. The fault is within the body of the message.
2. **Header:**
   This part is not mandatory. But when it is present it can provide crucial information about the applications.
3. **Body:**
   It contains the actual message that is being transmitted. Fault is contained within the body tags.
4. **Fault:**
   This section contains the status of the application and also contains errors in the application. This section is also optional. It should not appear more than once in a SOAP message.

Sample Message:

```
Content-Type: application/soap+xml
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
    <env:Header>
        <m:GetLastTradePrice xmlns:m="Some-URI" />
    </env:Header>
    <env:Body>
        <symbol xmlns:p="Some-URI" >DIS</symbol>
    </env:Body>
</env:Envelope>
```

Source: https://tools.ietf.org/html/rfc4227

Advantages of SOAP

1. SOAP is a light weight data interchange protocol because it is based on XML.
2. SOAP was designed to be OS and Platform independent.
3. It is built on top of HTTP which is installed in most systems.
4. It is suggested by W3 consortium which is like a governing body for the Web.
5. SOAP is mainly used for Web Services and Application Programming Interfaces (APIs).

Overview of SOAP Message Handlers

Web Services and their clients may need to access the SOAP message for additional processing of the message request or response. You can create SOAP message handlers to enable Web Services and clients to perform this additional processing on the SOAP message. A SOAP message handler provides a mechanism for intercepting the SOAP message in both the request and response of the Web Service.

A simple example of using handlers is to access information in the header part of the SOAP message. You can use the SOAP header to store Web Service specific information and then use handlers to manipulate it.

You can also use SOAP message handlers to improve the performance of your Web Service. After your Web Service has been deployed for a while, you might discover that many consumers invoke it with the same parameters. You could improve the performance of your Web Service by caching the results of popular invokes of the Web Service (assuming the results are static) and immediately returning these results when appropriate, without ever invoking the back-end components that implement the Web Service. You implement this performance improvement by using handlers to check the request SOAP message to see if it contains the popular parameters.

JAX-WS supports two types of SOAP message handlers: SOAP handlers and logical handlers. SOAP handlers can access the entire SOAP message, including the message headers and body. Logical handlers can access the payload of the message only, and cannot change any protocol-specific information (like headers) in a message.

Adding Server-side SOAP Message Handlers: Main Steps

The following procedure describes the high-level steps to add SOAP message handlers to your Web Service.

It is assumed that you have created a basic JWS file that implements a Web Service and that you want to update the Web Service by adding SOAP message handlers and handler chains. It is also assumed that you have set up an Ant-based development environment and that you have a working `build.xml`

Adding Server-side SOAP Message Handlers: Main Steps

The following procedure describes the high-level steps to add SOAP message handlers to your Web Service.

It is assumed that you have created a basic JWS file that implements a Web Service and that you want to update the Web Service by adding SOAP message handlers and handler chains. It is also assumed that you have set up an Ant-based development environment and that you have a working `build.xml` file that includes a target for running the `jwsc` Ant task. For more information, see in *Getting Started With WebLogic Web Services Using JAX-WS*:

- Use Cases and Examples
- Developing WebLogic Web Services
- Programming the JWS File
- Invoking Web Services

Table 8-1 Steps to Add SOAP Message Handlers to a Web Service

#	Step	Description
1	Design the handlers and handler chains.	Design SOAP message handlers and group them together in a *handler chain*. See Designing the SOAP Message Handlers and Handler Chains.
2	For each handler in the handler chain, create a Java class that implements the SOAP message handler interface.	See Creating the SOAP Message Handler.
3	Update your JWS file, adding annotations to configure the SOAP message handlers.	See Configuring Handler Chains in the JWS File.
4	Create the handler chain configuration file.	See Creating the Handler Chain Configuration File.
5	Compile all handler classes in the handler chain and rebuild your Web Service.	See Compiling and Rebuilding the Web Service.

Adding Client-side SOAP Message Handlers: Main Steps

You can configure client-side SOAP message handlers for both stand-alone clients and clients that run inside of WebLogic Server. You create the actual Java client-side handler in the same way you create a server-side handler—by creating a Java class that implements the SOAP message handler interface. In many cases you can use the exact same handler class on both the Web Service running on WebLogic Server *and* the client applications that invoke the Web Service. For example, you can write a generic logging handler class that logs all sent and received SOAP messages, both for the server and for the client.

The following procedure describes the high-level steps to add client-side SOAP message handlers to the client application that invokes a Web Service operation.

It is assumed that you have created the client application that invokes a deployed Web Service, and

Adding Client-side SOAP Message Handlers: Main Steps

You can configure client-side SOAP message handlers for both stand-alone clients and clients that run inside of WebLogic Server. You create the actual Java client-side handler in the same way you create a server-side handler—by creating a Java class that implements the SOAP message handler interface. In many cases you can use the exact same handler class on both the Web Service running on WebLogic Server *and* the client applications that invoke the Web Service. For example, you can write a generic logging handler class that logs all sent and received SOAP messages, both for the server and for the client.

The following procedure describes the high-level steps to add client-side SOAP message handlers to the client application that invokes a Web Service operation.

It is assumed that you have created the client application that invokes a deployed Web Service, and that you want to update the client application by adding client-side SOAP message handlers and handler chains. It is also assumed that you have set up an Ant-based development environment and that you have a working `build.xml` file that includes a target for running the `clientgen` Ant task. For more information, see "Invoking a Web Service from a Stand-alone Client: Main Steps" in *Getting Started With WebLogic Web Services Using JAX-WS*.

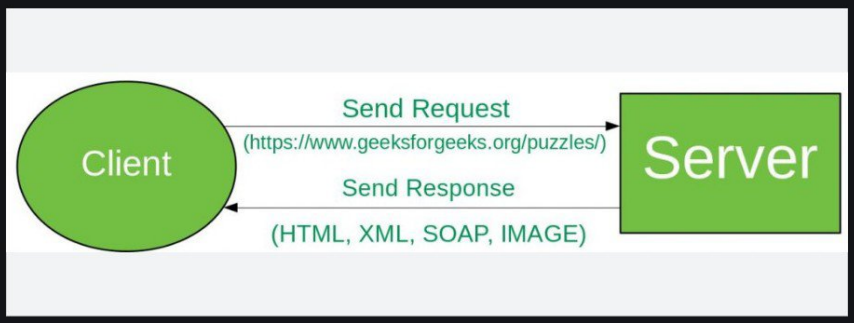
Table 8-2 Steps to Use Client-side SOAP Message Handlers

#	Step	Description
1	Design the handlers and handler chains.	This step is similar to designing the server-side SOAP message handlers, except the perspective is from the client application, rather than a Web Service. See Designing the SOAP Message Handlers and Handler Chains.
2	For each handler in the handler chain, create a Java class that implements the SOAP message handler interface.	This step is similar to designing the server-side SOAP message handlers, except the perspective is from the client application, rather than a Web Service. See Creating the SOAP Message Handler for details about programming a handler class.
3	Update your client to programmatically configure the SOAP message handlers.	See Configuring the Client-side SOAP Message Handlers.
4	Update the build.xml file that builds your application, specifying to the clientgen Ant task the customization file.	See Compiling and Rebuilding the Web Service.
5	Rebuild your client application by running the relevant task.	`prompt> ant build-client`

Representational **S**tate **T**ransfer (REST) is an architectural style that defines a set of constraints to be used for creating web services. **REST API** is a way of accessing web services in a simple and flexible way without having any processing.

REST technology is generally preferred to the more robust Simple Object Access Protocol (SOAP) technology because REST uses less bandwidth, simple and flexible making it more suitable for internet usage. It's used to fetch or give some information from a web service. All communication done via REST API uses only HTTP request.

Working: A request is sent from client to server in the form of a web URL as HTTP GET or POST or PUT or DELETE request. After that, a response comes back from the server in the form of a resource which can be anything like HTML, XML, Image, or JSON. But now JSON is the most popular format being used in Web Services.



In **HTTP** there are five methods that are commonly used in a REST-based Architecture i.e., POST, GET, PUT, PATCH, and DELETE. These correspond to create, read, update, and delete (or CRUD) operations respectively. There are other methods which are less frequently used like OPTIONS and HEAD.

- **GET:** The HTTP GET method is used to **read** (or retrieve) a representation of a resource. In the safe path, GET returns a representation in XML or JSON and an HTTP response code of 200 (OK). In an error case, it most often returns a 404 (NOT FOUND) or 400 (BAD REQUEST).
- **POST:** The POST verb is most often utilized to **create** new resources. In particular, it's used to create subordinate resources. That is, subordinate to some other (e.g. parent) resource. On successful creation, return HTTP status 201, returning a Location header with a link to the newly-created resource with the 201 HTTP status.

NOTE: POST is neither safe nor idempotent

- **PUT:** It is used for **updating** the capabilities. However, PUT can also be used to **create** a resource in the case where the resource ID is chosen by the client instead of by the server. In other words, if the PUT is to a URI that contains the value of a non-existent resource ID. On successful update, return 200 (or 204 if not returning any content in the body) from a PUT. If using PUT for create, return HTTP status 201 on successful creation. PUT is not safe operation but it's idempotent.
- **PATCH:** It is used to **modify** capabilities. The PATCH request only needs to contain the changes to the resource, not the complete resource. This resembles PUT, but the body contains a set of instructions describing how a resource currently residing on the server should be modified to produce a new version. This means that the PATCH body should not just be a modified part of the resource, but in some kind of patch language like JSON Patch or XML Patch. PATCH is neither safe nor idempotent.
- **DELETE:** It is used to **delete** a resource identified by a URI. On successful deletion, return HTTP status 200 (OK) along with a response body.

Idempotence: An idempotent HTTP method is a HTTP method that can be called many times without different outcomes. It would not matter if the method is called only once, or ten times over. The result should be the same. Again, this only applies to the result, not the resource itself.

Example:

C

```c
1. a = 4 // It is Idempotence, as final value(a = 4)
         // would not change after executing it multiple
        // times.

2. a++ // It is not Idempotence because the final value
       // will depend upon the number of times the
      // statement is executed.
```

Request and Response

Now we will see how request and response work for different **HTTP** methods. Let's assume we have an **API(**https://www.geeksforgeeks.org/api/students**)** for all students data of gfg.

- **GET:** Request for all Students.

Request
GET:/api/students

- **POST:** Request for Posting/Creating/Inserting Data

Request
POST:/api/students {"name":"Raj"}

- **PUT or PATCH:** Request for Updating Data at id=1

Request
PUT or PATCH:/api/students/1 {"name":"Raj"}

- **DELETE:** Request for Deleting Data of id=1

Request
DELETE:/api/students/1

RESTful web services are very popular because they are light weight, highly scalable and maintainable and are very commonly used to create APIs for web-based applications.

Characteristics of Virtualization

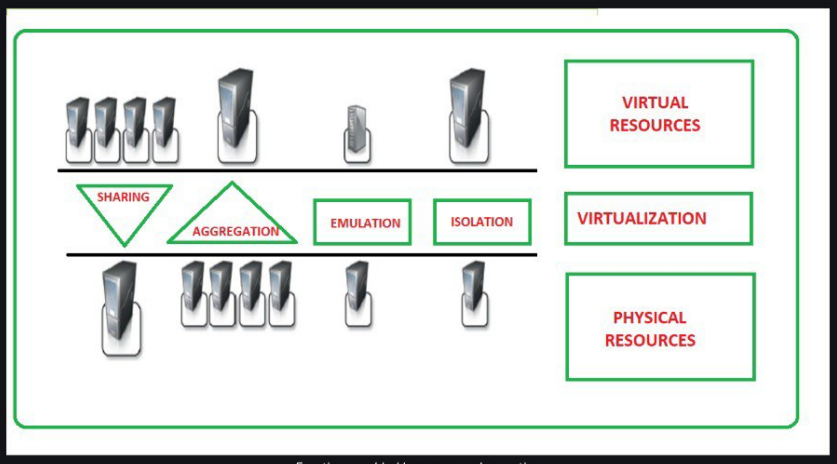
Prerequisite – Virtualization In Cloud Computing and Types

1. Increased Security –

The ability to control the execution of a guest program in a completely transparent manner opens new possibilities for delivering a secure, controlled execution environment. All the operations of the guest programs are generally performed against the virtual machine, which then translates and applies them to the host programs.
A virtual machine manager can control and filter the activity of the guest programs, thus preventing some harmful operations from being performed. Resources exposed by the host can then be hidden or simply protected from the guest. Increased security is a requirement when dealing with untrusted code.
Example-1: Untrusted code can be analyzed in Cuckoo sandboxes environment.
The term sandbox identifies an isolated execution environment where instructions can be filtered and blocked before being translated and executed in the real execution environment.
Example-2: The expression sandboxed version of the Java Virtual Machine (JVM) refers to a particular configuration of the JVM where, by means of security policy, instructions that are considered potentially harmful can be blocked.

2. Managed Execution –

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



Functions enabled by a managed execution

3. Sharing –

Virtualization allows the creation of a separate computing environment within the same host. This basic feature is used to reduce the number of active servers and limit power consumption.

4. Aggregation –

It is possible to share physical resources 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. This functionality is implemented with cluster management software, which harnesses the physical resources of a homogeneous group of machines and represents them as a single resource.

4. Aggregation –

It is possible to share physical resources 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. This functionality is implemented with cluster management software, which harnesses the physical resources of a homogeneous group of machines and represents them as a single resource.

5. Emulation –

Guest programs are executed within an environment that is controlled by the virtualization layer, which ultimately is a program. Also, a completely different environment with respect to the host can be emulated, thus allowing the execution of guest programs requiring specific characteristics that are not present in the physical host.

6. 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. The virtual machine can filter the activity of the guest and prevent harmful operations against the host.

Besides these characteristics, 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.

7. 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.
In the case of programming-level virtualization, as implemented by the JVM or the .NET runtime, the binary code representing application components (jars or assemblies) can run without any recompilation on any implementation of the corresponding virtual machine.

8.Resource sharing:

Virtualization allows multiple virtual machines to share the resources of a single physical machine, such as CPU, memory, storage, and network bandwidth. This improves hardware utilization and reduces the need for additional physical servers.

9.Flexibility:

Virtualization allows IT administrators to quickly and easily create, modify, or delete virtual machines as needed, without the need to purchase and configure additional physical hardware.

11.Hardware independence:

Virtual machines are hardware-independent, which means they can run on different types of physical hardware and can be easily moved between physical servers without needing to reconfigure the virtual machine.

12.Scalability:

Virtualization allows organizations to scale their computing resources up or down as needed, depending on changing business requirements.

13.Management:

Virtualization provides centralized management tools that allow IT administrators to monitor and manage multiple virtual machines from a single console, making it easier to troubleshoot and maintain the virtualized environment.

virtualization provides a powerful and flexible technology that can help organizations maximize their hardware utilization, improve their IT infrastructure security, and simplify their IT management.

business requirements.

13.Management:

Virtualization provides centralized management tools that allow IT administrators to monitor and manage multiple virtual machines from a single console, making it easier to troubleshoot and maintain the virtualized environment.

virtualization provides a powerful and flexible technology that can help organizations maximize their hardware utilization, improve their IT infrastructure scalability, and simplify their IT management.

14.Disaster Recovery:

Virtualization enables organizations to implement disaster recovery solutions more easily and cost-effectively. By replicating virtual machines to remote sites, organizations can quickly recover from a disaster and resume operations.

15.Testing and Development:

Virtualization allows organizations to easily create test and development environments without the need for additional physical hardware. This enables developers to test new applications and configurations in a controlled environment before deploying them to production.

16.Energy Efficiency:

Virtualization can help organizations reduce their energy consumption by consolidating multiple physical servers onto a single machine. This can result in lower electricity bills and a reduced carbon footprint.

17.Increased Uptime:

Virtualization can help improve system uptime by enabling virtual machines to be migrated to different physical hosts in the event of a hardware failure. This can minimize downtime and prevent data loss.

18.Cost Savings:

Virtualization can help organizations save money by reducing the need for additional physical hardware, lowering electricity bills, and streamlining IT operations. This can result in significant cost savings over time.

19.Improved Security Management:

Virtualization can help organizations improve their security posture by isolating applications and workloads from each other. This can prevent security breaches from spreading to other parts of the environment and reduce the attack surface.

20.Cloud Migration:

Virtualization can be a stepping stone for organizations looking to migrate to the cloud. By virtualizing their existing infrastructure, organizations can make it easier to move workloads to the cloud and take advantage of cloud-based services.

Last Updated : 13 Apr, 2023

Previous
Evolution of Cloud Computing

Next
MD5 hash in Java

Pros and cons of Virtualization in Cloud Computing

Virtualization is the creation of Virtual Version of something such as server, desktop, storage device, operating system etc.

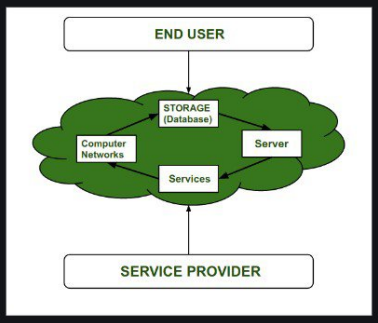
Thus, Virtualization is a technique which allows us to share a single physical instance of a resource or an application among multiple customers and an organization. Virtualization often creates many virtual resources from one physical resource.

- **Host Machine –**
  The machine on which virtual machine is going to create is known as Host Machine.

- **Guest Machine –**
  The virtual machines which are created on Host Machine is called Guest Machine.

Why Virtualization in Cloud Computing ?

Virtualization is very important concept in cloud computing. In cloud computing, a cloud vendor who will provide cloud services have all physical resources like server, storage device, network device etc. and these physical services are rented by cloud vendors so that user's will not worry about these physical services.



But it is very costly to provide physical services per customer on rent because firstly it becomes very costly and also user's will not use the fully services. So this problem can be solved by Virtualization. It is very cool approach for not only efficient use of Physical services but also reduce costs of vendors. Thus cloud vendor's can vitalize their single big server and provide smaller spec server to multiple customer's

Pros of Virtualization in Cloud Computing :

- **Utilization of Hardware Efficiently –**
  With the help of Virtualization Hardware is Efficiently used by user as well as Cloud Service Provider. In this the need of Physical Hardware System for the User is decreases and this results in less costly.In Service Provider point of View, they will vitalize the Hardware using Hardware Virtualization which decrease the Hardware requirement from Vendor side which are provided to User is decreased. Before Virtualization, Companies and organizations have to set up their own Server which require extra space for placing them, engineer's to check its performance and require extra hardware cost but with the help of Virtualization the all these limitations are removed by Cloud vendor's who provide Physical Services without setting up any Physical Hardware system.
- **Availability increases with Virtualization –**
  One of the main benefit of Virtualization is that it provides advance features which allow virtual instances to be available all the times. It also has capability to move virtual instance from one virtual Server another Server which is very tedious and risky task in Server Based System. During migration of Data from one server to another it ensures its safety. Also, we can access information from any

also user's will not use the fully services. So this problem can be solved by Virtualization. It is approach for not only efficient use of Physical services but also reduce costs of vendors. Thus cloud vendor's can vitalize their single big server and provide smaller spec server to multiple customer's

Pros of Virtualization in Cloud Computing :

- **Utilization of Hardware Efficiently –**
  With the help of Virtualization Hardware is Efficiently used by user as well as Cloud Service Provider. In this the need of Physical Hardware System for the User is decreases and this results in less costly.In Service Provider point of View, they will vitalize the Hardware using Hardware Virtualization which decrease the Hardware requirement from Vendor side which are provided to User is decreased. Before Virtualization, Companies and organizations have to set up their own Server which require extra space for placing them, engineer's to check its performance and require extra hardware cost but with the help of Virtualization the all these limitations are removed by Cloud vendor's who provide Physical Services without setting up any Physical Hardware system.
- **Availability increases with Virtualization –**
  One of the main benefit of Virtualization is that it provides advance features which allow virtual instances to be available all the times. It also has capability to move virtual instance from one virtual Server another Server which is very tedious and risky task in Server Based System. During migration of Data from one server to another it ensures its safety. Also, we can access information from any location and any time from any device.
- **Disaster Recovery is efficient and easy –**
  With the help of virtualization Data Recovery, Backup, Duplication becomes very easy. In traditional method , if somehow due to some disaster if Server system Damaged then the surety of Data Recovery is very less. But with the tools of Virtualization real time data backup recovery and mirroring become easy task and provide surety of zero percent data loss.
- **Virtualization saves Energy –**
  Virtualization will help to save Energy because while moving from physical Servers to Virtual Server's, the number of Server's decreases due to this monthly power and cooling cost decreases which will Save Money as well. As cooling cost reduces it means carbon production by devices also decreases which results in Fresh and pollution free environment.
- **Quick and Easy Set up –**
  In traditional methods Setting up physical system and servers are very time-consuming. Firstly Purchase them in bulk after that wait for shipment. When Shipment is done then wait for Setting up and after that again spend time in installing required software etc. Which will consume very time. But with the help of virtualization the entire process is done in very less time which results in productive setup.
- **Cloud Migration becomes easy –**
  Most of the companies those who already have spent a lot in the server have a doubt of Shifting to Cloud. But it is more cost-effective to shift to cloud services because all the data that is present in their server's can be easily migrated into the cloud server and save something from maintenance charge, power consumption, cooling cost, cost to Server Maintenance Engineer etc.

Cons of Virtualization :

- **Data can be at Risk –**
  Working on virtual instances on shared resources means that our data is hosted on third party resource which put's our data in vulnerable condition. Any hacker can attack on our data or try to perform unauthorized access. Without Security solution our data is in threaten situation.
- **Learning New Infrastructure –**
  As Organization shifted from Servers to Cloud. They required skilled staff who can work with cloud easily. Either they hire new IT staff with relevant skill or provide training on that skill which increase the cost of company.
- **High Initial Investment –**
  It is true that Virtualization will reduce the cost of companies but also it is truth that Cloud have high initial investment. It provides numerous services which are not required and when unskilled organization will try to set up in cloud they purchase unnecessary services which are not even required to them.

Last Updated : 31 Mar, 2021

oVirt

Article Talk

This article relies excessively on references to primary sources. *(October 2010)*
Learn more

oVirt is a free, open-source virtualization management platform. It was founded by Red Hat as a community project on which Red Hat Virtualization is based. It allows centralized management of virtual machines, compute, storage and networking resources, from an easy-to-use web-based front-end with platform independent access. KVM on x86-64, PowerPC64[2][3] and s390x[4] architecture are the only hypervisors supported, but there is an ongoing effort to support ARM architecture in a future releases.

Contents

Architecture

Features

Virtual datacenters, managed by oVirt, are categorized into storage, networking and clusters that consist of one or more oVirt nodes. Data integrity is ensured by fencing, with agents that can use various resources such as baseboard management controllers or uninterruptible power supplies.

Storage is organized within entities called storage domains and can be local or shared. Storage domains can be created using the following storage solutions or protocols:

- NFS
- iSCSI
- Fibre Channel
- POSIX compliant filesystem
- GlusterFS

Network management allows defining multiple VLANs that can be bridged to the network interfaces available on the nodes. Configuration of bonded interfaces, IP addresses, subnet masks and gateways on managed nodes are all supported within webadmin portal interface, as is SR-IOV on hardware configurations that support this feature.

Management features for compute resources include CPU pinning, defining NUMA topology, enabling kernel same-page merging, memory over-provisioning, HA VM reservation etc.

Virtual machine management enables selecting high availability priority, live migration, live snapshots, cloning virtual machines from snapshots, creating virtual machine templates, using cloud-init for automated configuration during provisioning and deployment of virtual machines. Supported guest operating systems include Linux, Microsoft Windows and FreeBSD. Access to virtual machines can be achieved from webadmin portal using SPICE, VNC and RDP protocols.

oVirt can be integrated with many open source projects, including OpenStack Glance and Neutron for disk and network provisioning, Foreman/Katello for VM/node provisioning or pulling relevant errata information into webadmin portal and can be further integrated with ManageIQ for a complete virtual infrastructure lifecycle management.[7]

Disaster recovery features include the ability to import any storage domain into different oVirt engine instances and replication can be managed from oVirt with GlusterFS geo-replication feature, or by utilizing synchronous/asynchronous block level replication provided by storage hardware vendors. oVirt engine backups can be automated and periodically transferred to a remote location.

oVirt supports hyper-converged infrastructure deployment scenarios.[8] Self-hosted engine and Gluster-based storage domains allow centralized management of all resources that can be seamlessly expanded, simply by adding an appropriate number of nodes to the cluster, without having any single points of failure. oVirt provides deep integration with Gluster, including Gluster specific performance improvements.

See also

- Red Hat Virtualization (RHV)

oVirt	
Original author(s)	Red Hat
Developer(s)	oVirt Project
Stable release	4.5.5[1] / December 1, 2023; 27 days ago
Repository	github.com/ovirt/
Written in	Java
Operating system	Linux
Platform	Java
Available in	English, Japanese, French, German, Italian, Spanish
Type	Virtual machine
Licence	Apache License 2.0
Website	www.ovirt.org