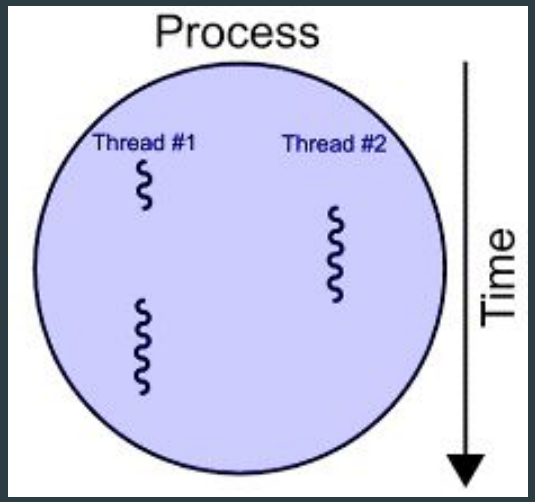
Processes

Introduction: Process

- a program that is under execution
- forms a building block in distributed systems

Introduction: Thread

- a path of execution within a process
- a process can contain multiple threads
- a thread executes its own piece of code, independently from other threads but no attempt is made to achieve a high degree of concurrency transparency



Thread Usage in Nondistributed Systems

- Problem: Consider a spreadsheet program that maintains the dependencies between different cells (whenever a cell is modified, all dependent cells are automatically updated)
- When a user changes the value in a single cell, such a modification can trigger a large series of computations
- If there is only a single thread of control, computation cannot proceed while the program is waiting for input
- Likewise, it is not easy to provide input while dependencies are being calculated

Thread Usage in Nondistributed Systems

- Problem: Consider a spreadsheet program that maintains the dependencies between different cells (whenever a cell is modified, all dependent cells are automatically updated)
- Solution: have at least two threads of control
  - one for handling interaction with the user
  - one for updating the spreadsheet

Threads in Distributed Systems

- An important property of threads is that they can provide a convenient means of allowing blocking system calls without blocking the entire process in which the thread is running
- This property makes threads particularly attractive to use in distributed systems as it makes it much easier to express communication in the form of maintaining multiple logical connections at the same time

Multithreaded Clients

- To establish a high degree of distribution transparency, distributed systems that operate in wide-area networks may need to hide long interprocess message propagation times
- The usual way to hide communication latencies is to initiate communication and immediately proceed with something else
- Example: Web browsers

Multithreaded Clients

- Consider a Web document consists of an HTML file containing plain text along with a collection of images, icons, etc
- To fetch each element of a Web document, the browser has to set up a TCP-IP connection, read the incoming data, and pass it to a display component
- Setting up a connection as well as reading incoming data are inherently blocking operations
- When dealing with long distance communication, the time for each operation to complete may be relatively long

Multithreaded Clients

- A Web browser often starts with fetching the HTML page and subsequently displays it
- To hide communication latencies as much as possible, some browsers start displaying data while it is still coming in
- While the text is made available to the user, including the facilities for scrolling and such, the browser continues with fetching other files that make up the page, such as the images. The latter are displayed as they are brought in
- The user need thus not wait until all the components of the entire page are fetched before the page is made available

Multithreaded Clients

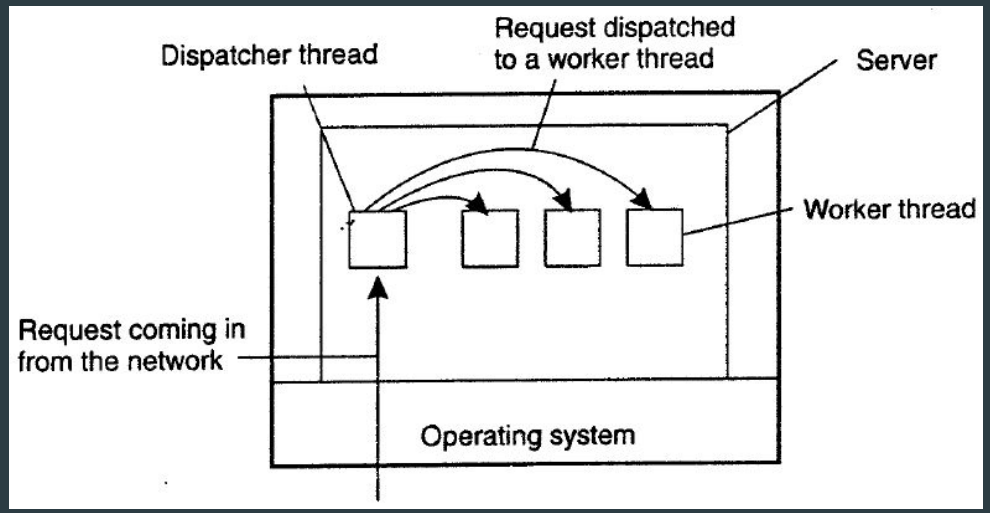
- Developing the browser as a multithreaded client simplifies matters considerably
- As soon as the main HTML file has been fetched, separate threads can be activated to take care of fetching the other parts
- Each thread sets up a separate connection to the server and pulls in the data. Setting up a connection and reading data from the server can be programmed using the standard (blocking) system calls, assuming that a blocking call does not suspend the entire process
- Meanwhile, the user notices only delays in the display of images and such, but can otherwise browse through the document

Multithreaded Clients

- If the server is heavily loaded, or just plain slow, no real performance improvements will be noticed compared to pulling in the files that make up the page strictly one after the other
- In many cases, Web servers have been replicated across multiple machines, where each server provides exactly the same set of Web documents
- The replicated servers are located at the same site, and are known under the same name
- When a request for a Web page comes in, the request is forwarded to one of the servers, often using a round-robin strategy or some other load-balancing technique
- It allows data to be transferred in parallel

Multithreaded Servers

- Consider the organization of a file server that occasionally has to block waiting for the disk
- The file server normally waits for an incoming request for a file operation, subsequently carries out the request, and then sends back the reply



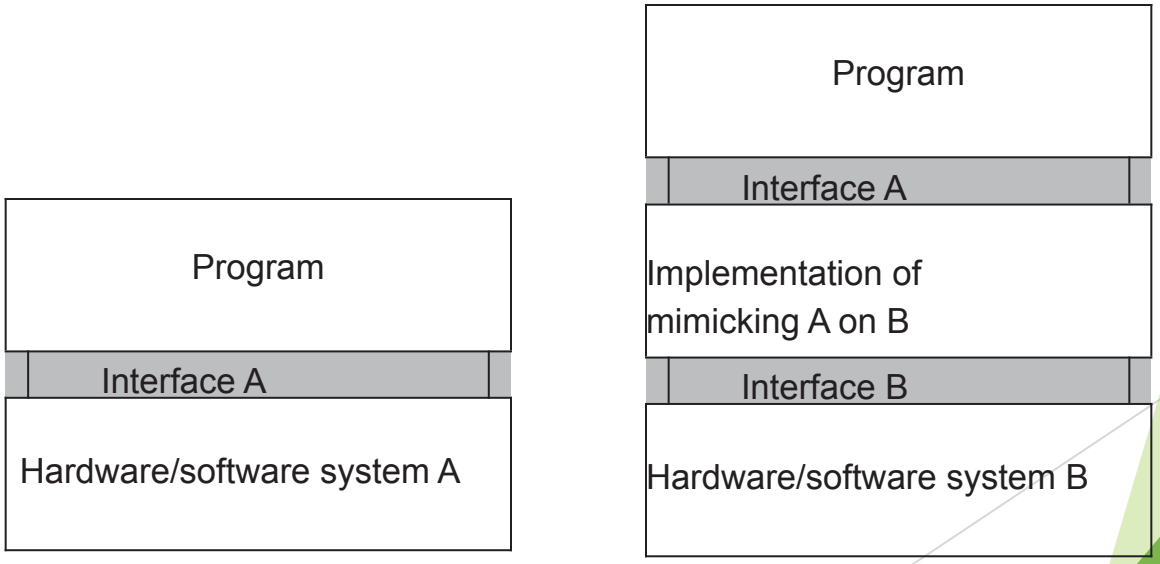
Multithreaded Servers

- Here one thread, the dispatcher, reads incoming requests for a file operation
- The requests are sent by clients to a well-known end point for this server. After examining the request, the server chooses an idle (i.e., blocked) worker thread and hands it the request
- The worker proceeds by performing a blocking read on the local file system, which may cause the thread to be suspended until the data are fetched from disk.
- If the thread is suspended, another thread is selected to be executed.

Virtualization Observation Virtualization is important:

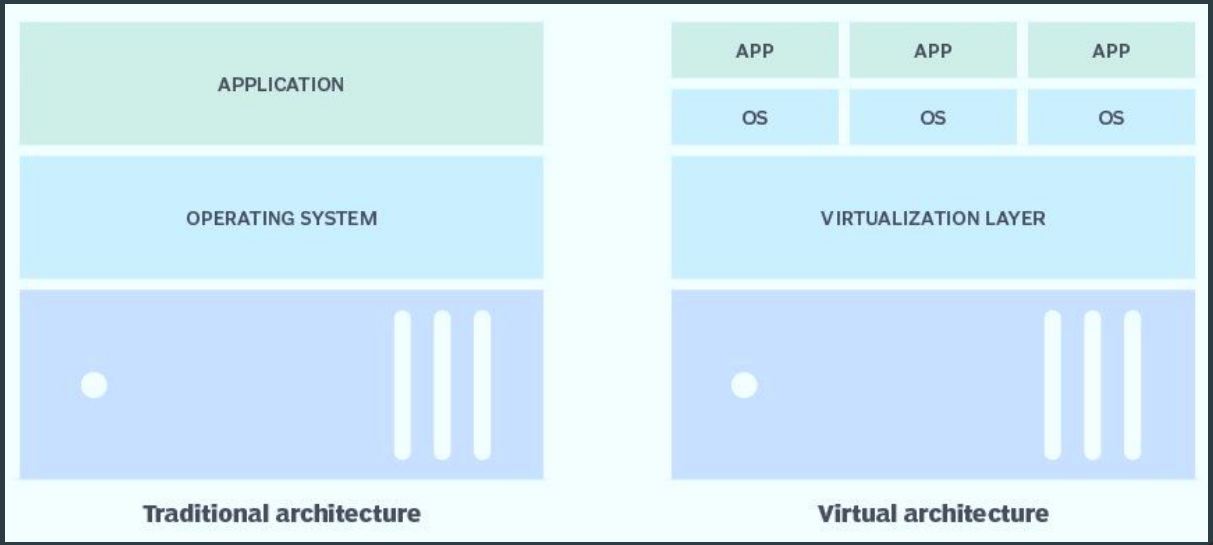
- Hardware changes faster than software
- Ease of portability and code migration
- Isolation of failing or attacked components

Principle: mimicking interfaces



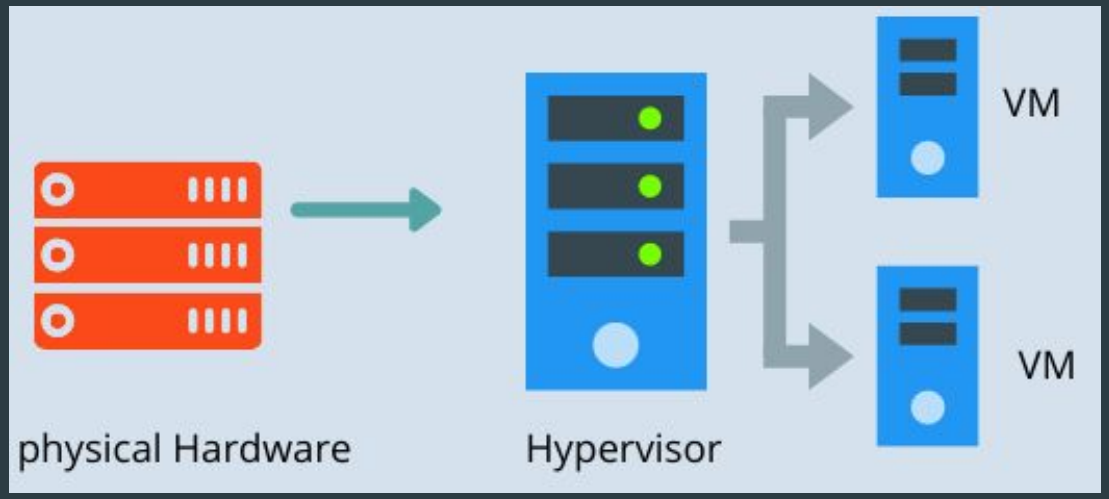
Virtualization

- A technology that allows to create multiple simulated environments or dedicated resources from a single, physical hardware system



Hypervisor

- A software that connects directly to that hardware and allows you to split one system into separate, distinct, and secure environments

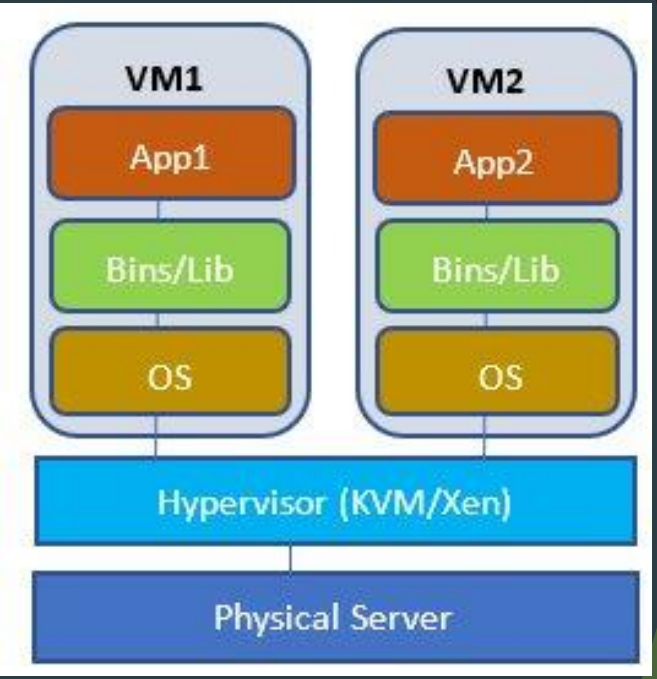


Hypervisor

- Example: Oracle VM VirtualBox

Virtual Machine (VM)

- A computer resource that uses software instead of a physical computer to run programs and deploy apps
- One or more virtual “guest” machines run on a physical “host” machine.
- Each virtual machine runs its own operating system and functions separately from the other VMs, even when they are all running on the same host.



Abhishek, Manish. (2020). High Performance Computing using Containers in Cloud. International Journal of Advanced Trends in Computer Science and Engineering. 9. 5686. 10.30534/ijatcse/2020/220942020.

Benefits of using VMs

- **Lower costs:** Virtualization reduces the amount of hardware servers necessary within a company and data center. This lowers the overall cost of buying and maintaining large amounts of hardware.
- **Agility and speed:** Spinning up a VM is relatively easy and quick and is much simpler than provisioning an entire new environment for your developers. Virtualization makes the process of running dev-test scenarios a lot quicker.

Benefits of using VMs

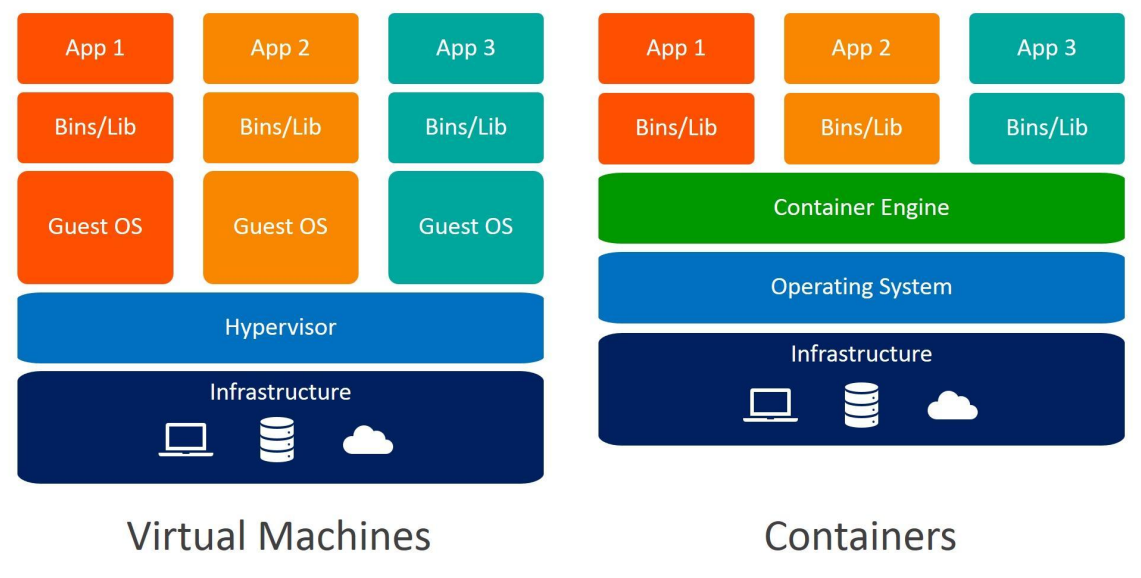
- **Easier disaster recovery:** Disaster recovery is very simple in a virtualized environment. Regular snapshots provide up-to-date data, allowing virtual machines to be feasibly backed up and recovered. Even in an emergency, a virtual machine can be migrated to a new location within minutes.
- **Security benefits:** Because virtual machines run in multiple operating systems, using a guest operating system on a VM allows you to run apps of questionable security and protects your host operating system.

Containerization

- the packaging of software code with just the operating system (OS) libraries and dependencies required to run the code to create a single lightweight executable that runs consistently on any infrastructure.

Container

- a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another.

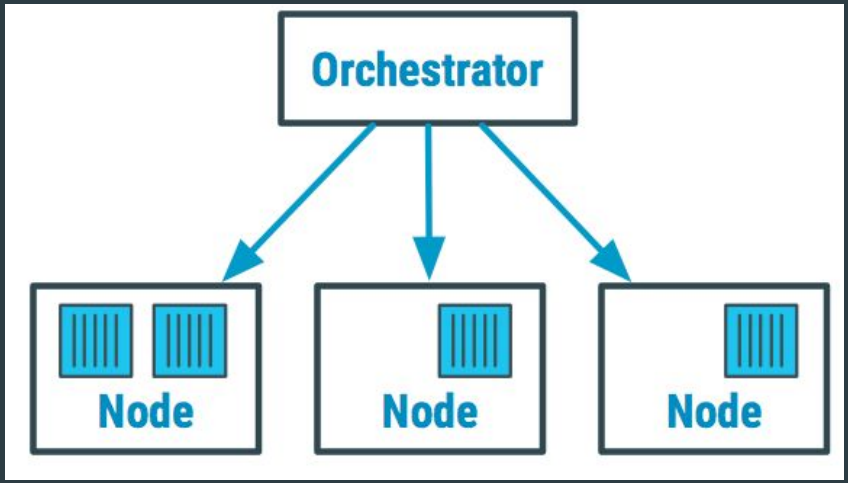


Container

- is often referred to as “lightweight - it shares the machine’s operating system kernel and does not require the overhead of associating an operating system within each application.
- is inherently smaller in capacity than a VM and requires less start-up time, allowing far more containers to run on the same compute capacity as a single VM. This drives higher server efficiencies and, in turn, reduces server and licensing costs.

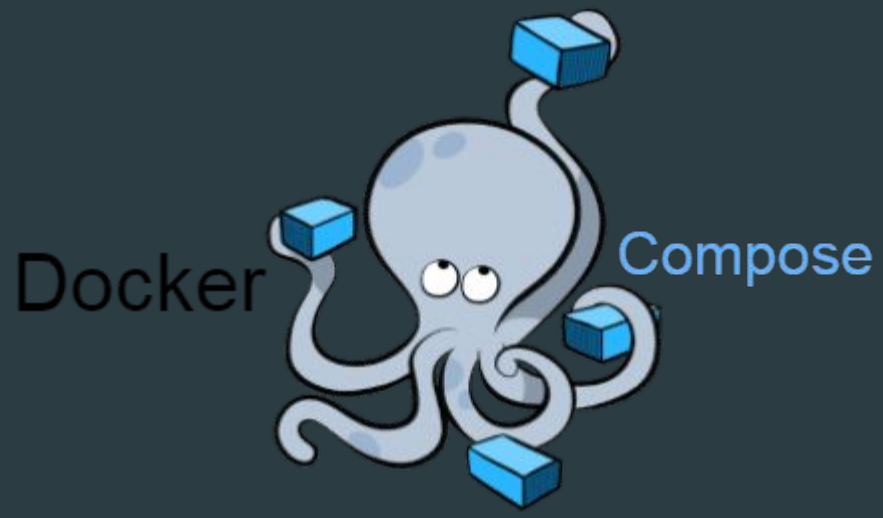
Orchestration

- A process that automates the deployment, management, scaling, networking, and availability of container-based applications.
- By abstracting the host infrastructure, container orchestration tools allow the users deploying to entire cluster as a single deployment target.



Orchestration

- Example: Docker compose



Resources

- https://www.youtube.com/watch?v=GeXwR32GCOw
- https://www.techtarget.com/searchitoperations/definition/virtualization
- https://www.docker.com/resources/what-container/
- https://www.youtube.com/watch?v=kBF6Bvth0zw