Building Software Systems

Lecture 2.3

Introduction to Cloud and Virtualisation

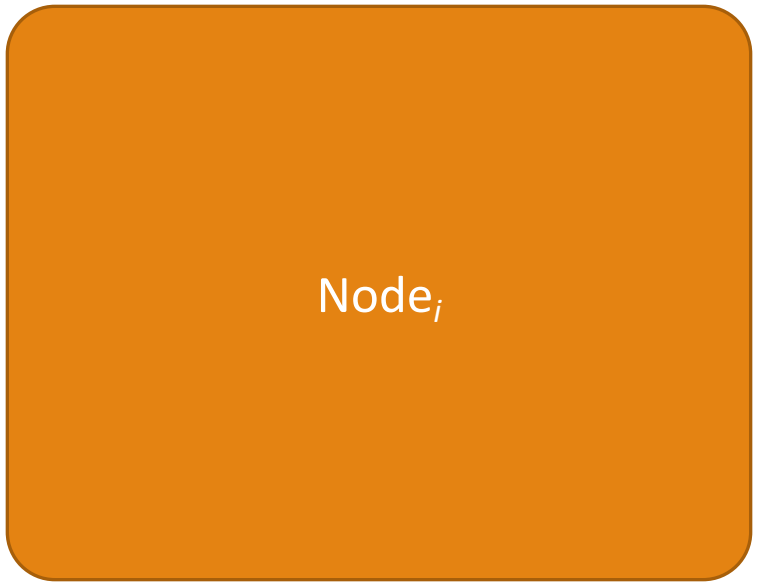
SAURABH SRIVASTAVA

ASSISTANT PROFESSOR

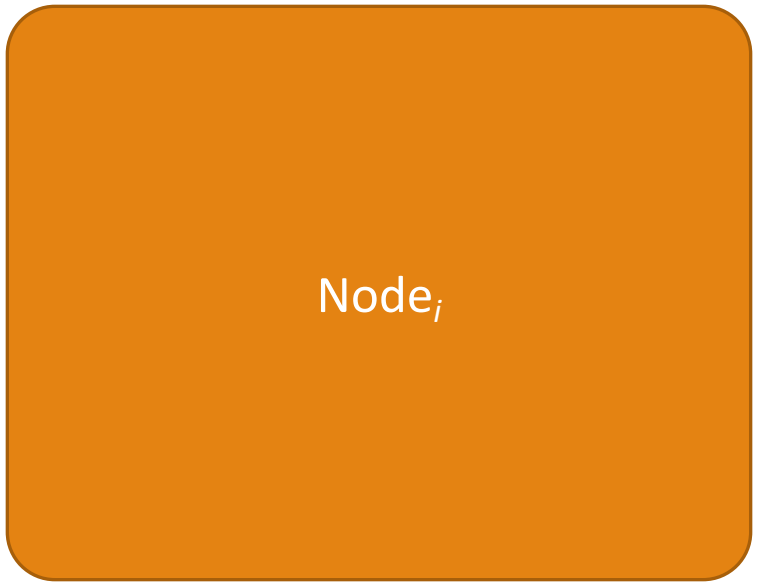
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

IIT (ISM) DHANBAD

Computation on a single node

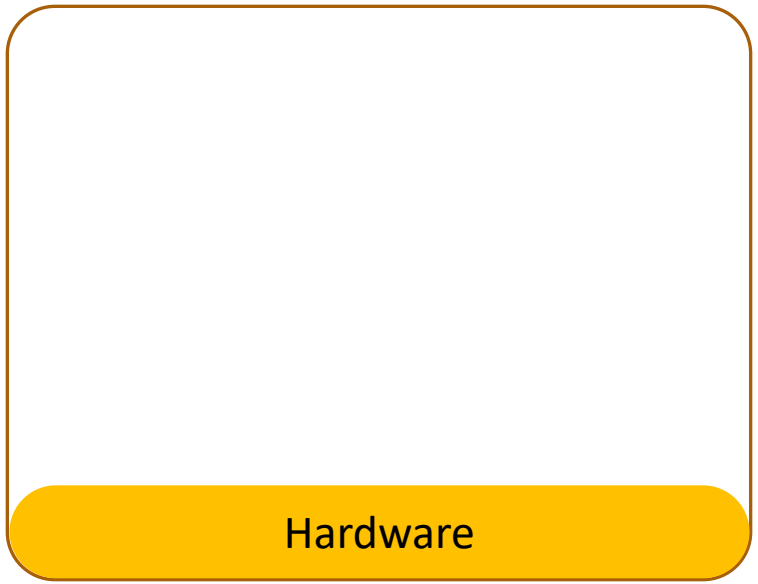


Computation on a single node



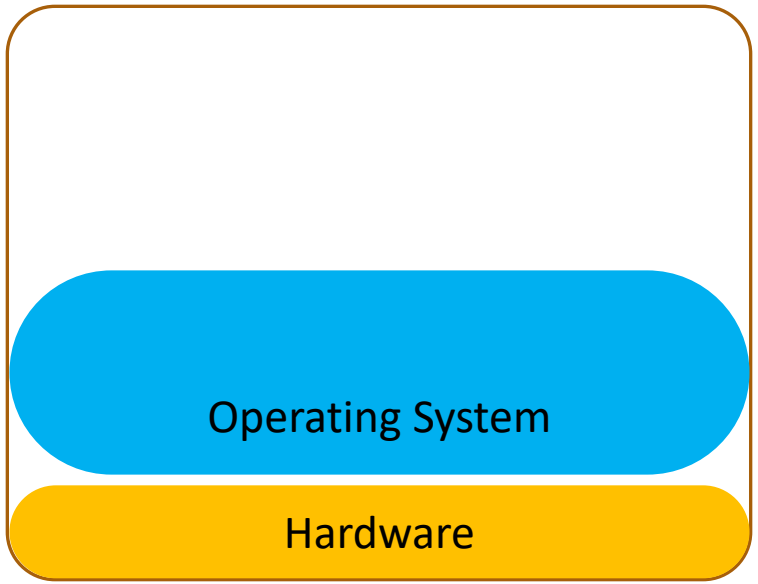
A node may be considered as an independent computational unit

Computation on a single node



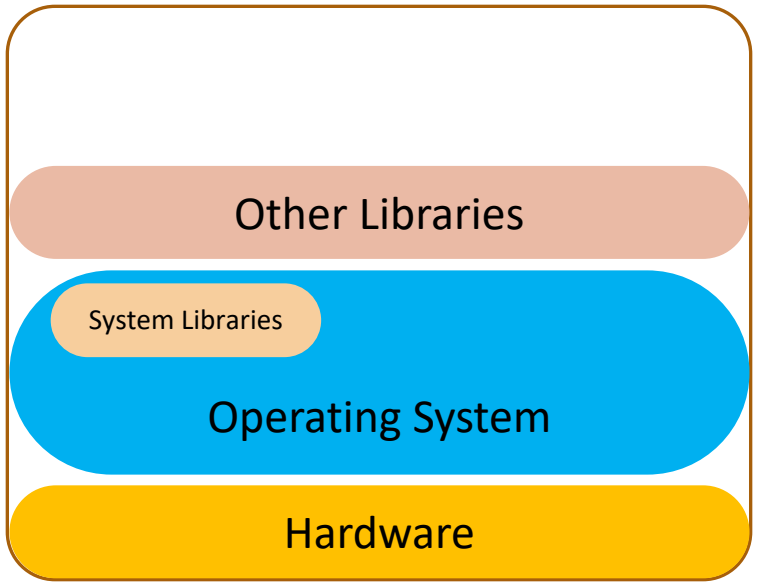
For now, assume that it means a collection of some hardware, e.g., a processor, main memory, secondary storage, I/O devices etc.

Computation on a single node



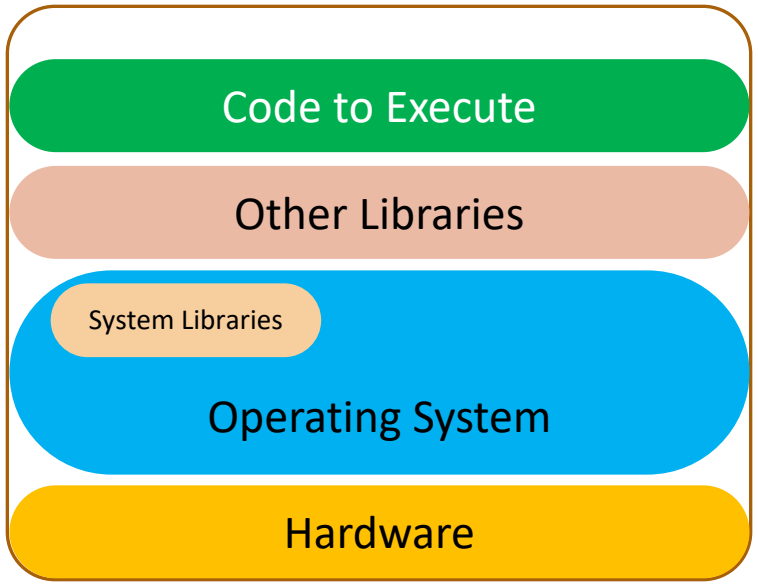
Which is managed by an Operating System (OS), such as Windows or a Linux Distribution

Computation on a single node



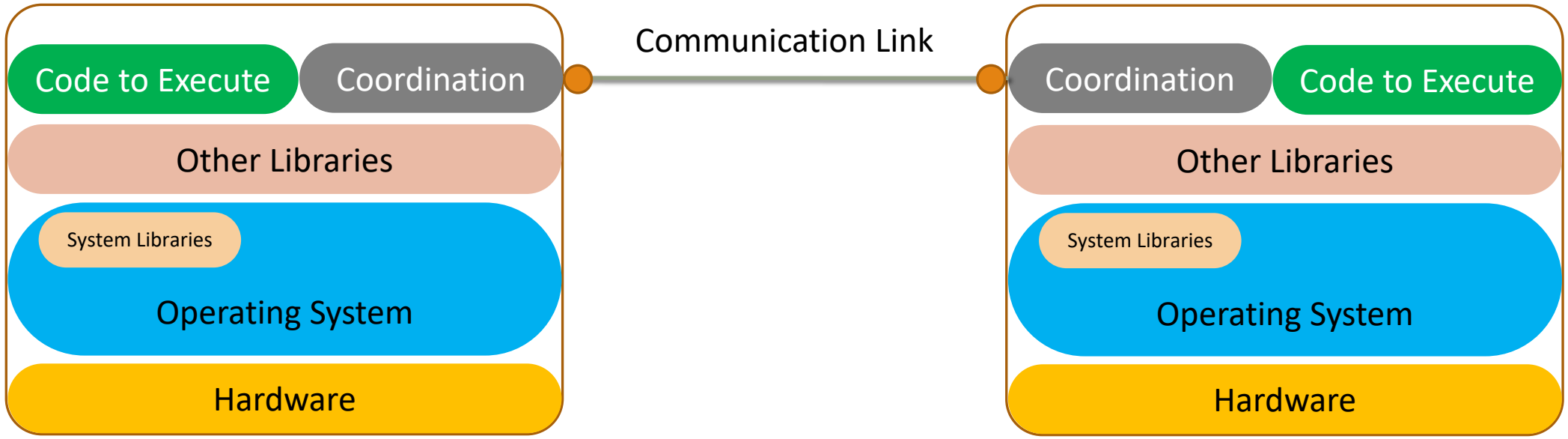
There are libraries – both system libraries (usually packaged along with the OS itself) and other user-installed libraries – which can be used by custom applications to perform computations over the hardware

Computation on a single node

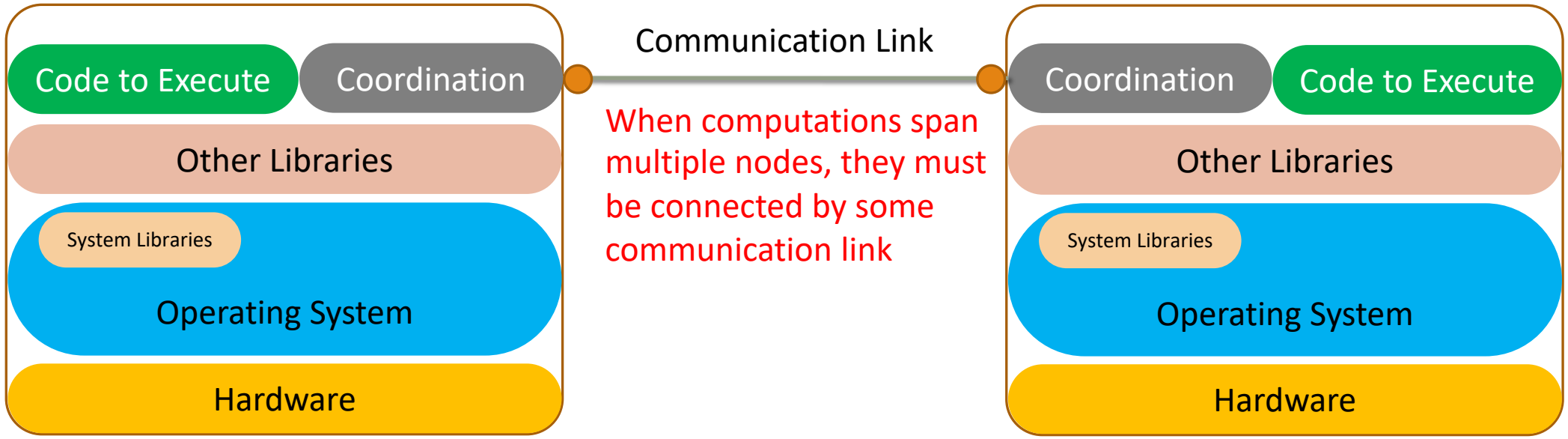


On top of this stack, custom applications or programs may be executed over the node

Computation on multiple nodes

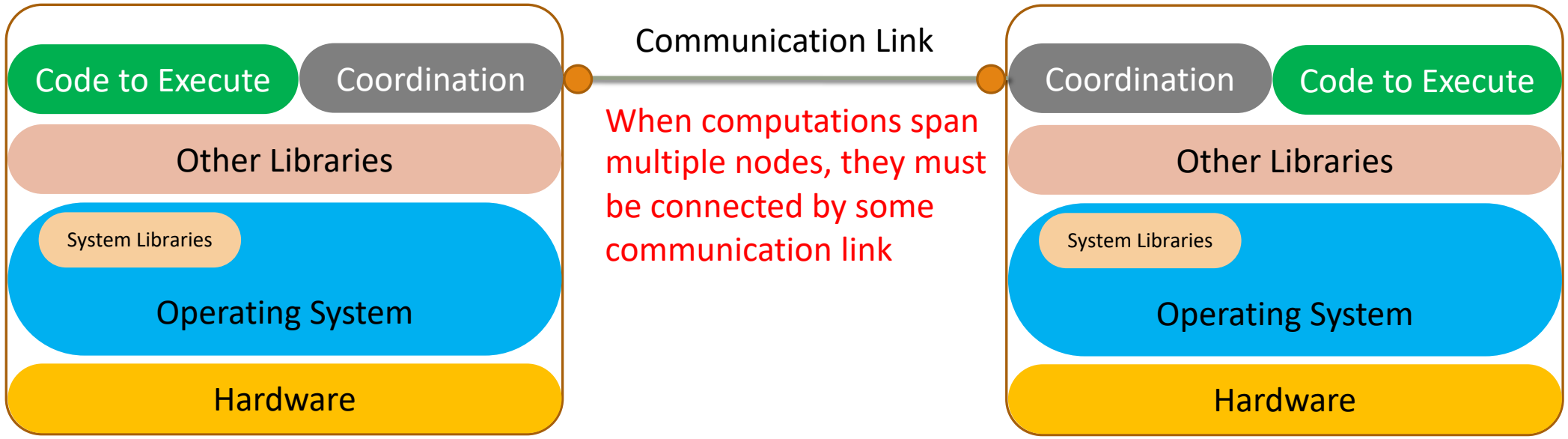


Computation on multiple nodes



Computation on multiple nodes

We also need some additional logic to perform the communication and coordination tasks



Executing code “remotely”

The nodes in a multi-node computation arrangement are usually connected through a network

- These connections may be physical – e.g., through RJ45 cables …
- … or via *Software-Defined Networking* – especially if the nodes are “virtualised” (we’ll revisit this term soon)

The coordination could be done by the application code, for instance

- Remote Procedure Call (RPC) – though which a program can fetch results of a procedure executed remotely
- Remote Method Invocation (RMI) – Java’s peer to RPC for object-oriented setup
- Common Object Resource Broker Architecture (CORBA) – generic specification for distributed computing

Otherwise, a software layer may be added on the nodes to make the bottom layers transparent

- This means that the actual hardware and/or the Operating System on the nodes are hidden from other nodes
- This layer manages the hardware – either directly, or through the operating system
- The applications communicate with this layer and perform the required tasks over one or more nodes

Executing code “remotely”

The nodes in a multi-node computation arrangement are usually connected through a network

- These connections may be physical – e.g., through RJ45 cables …
- … or via *Software-Defined Networking* – especially if the nodes are “virtualised” (we’ll revisit this term soon)

The coordination could be done by the application code, for instance

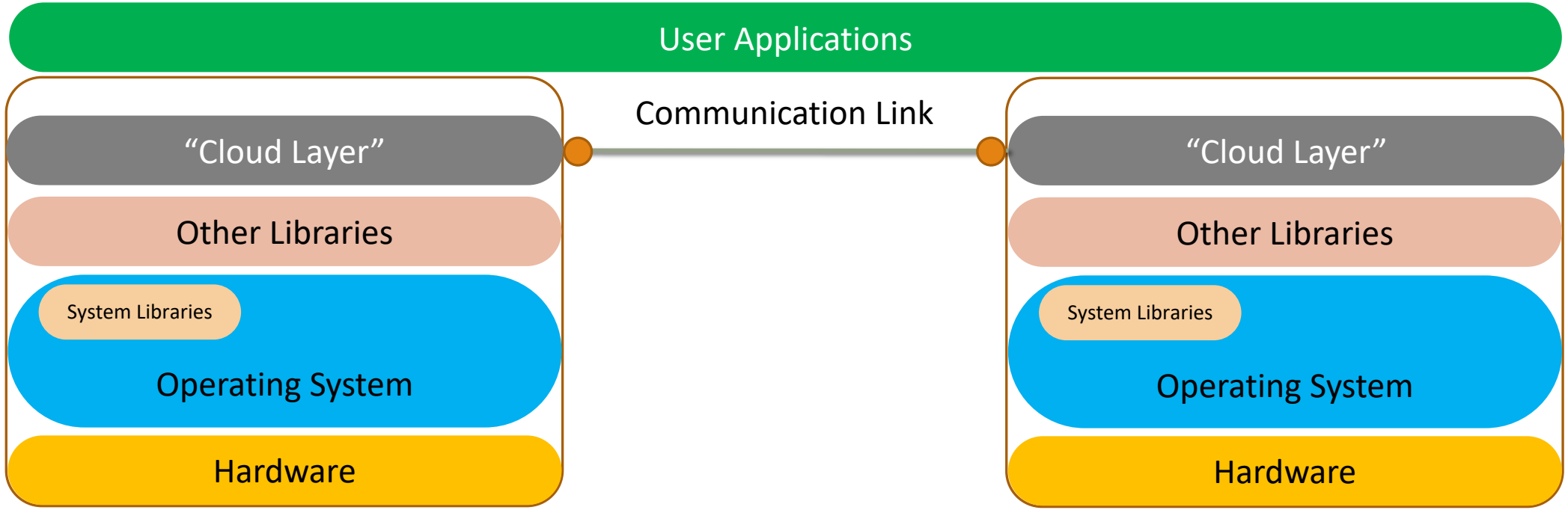
- Remote Procedure Call (RPC) – though which a program can fetch results of a procedure executed remotely
- Remote Method Invocation (RMI) – Java’s peer to RPC for object-oriented setup
- Common Object Resource Broker Architecture (CORBA) – generic specification for distributed computing

Otherwise, a software layer may be added on the nodes to make the bottom layers transparent

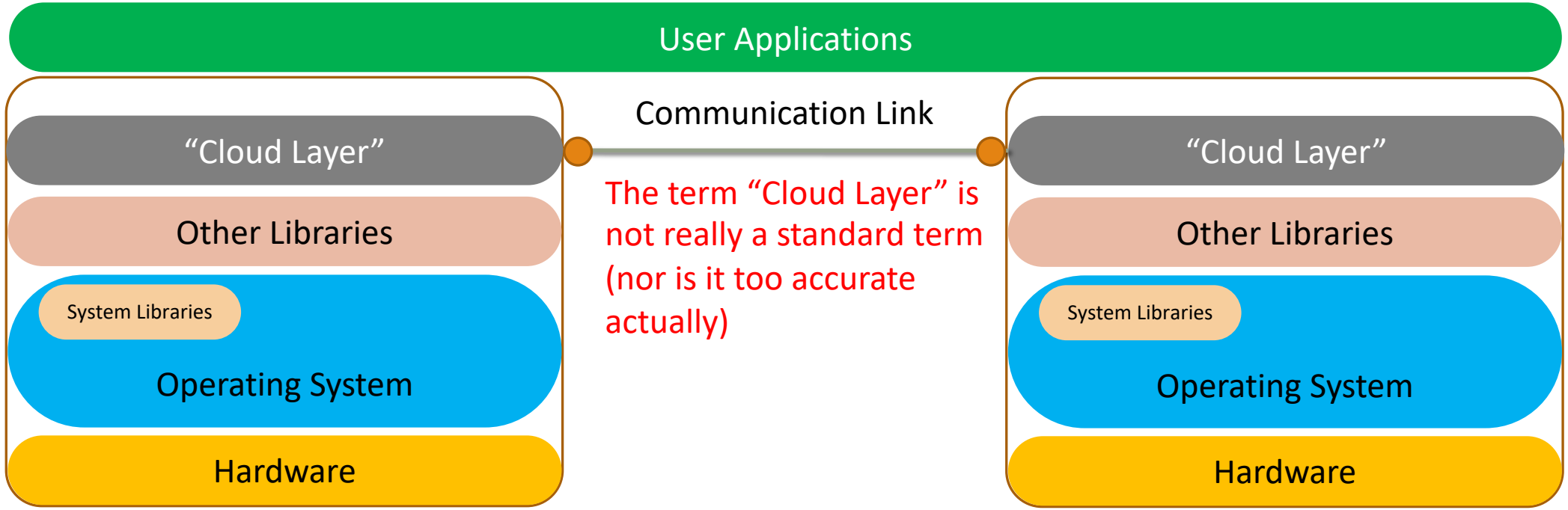
- This means that the actual hardware and/or the Operating System on the nodes are hidden from other nodes
- This layer manages the hardware – either directly, or through the operating system
- The applications communicate with this layer and perform the required tasks over one or more nodes

The bird’s eye view of a “cloud” looks very similar to what we just described !!

Computation on the “cloud”

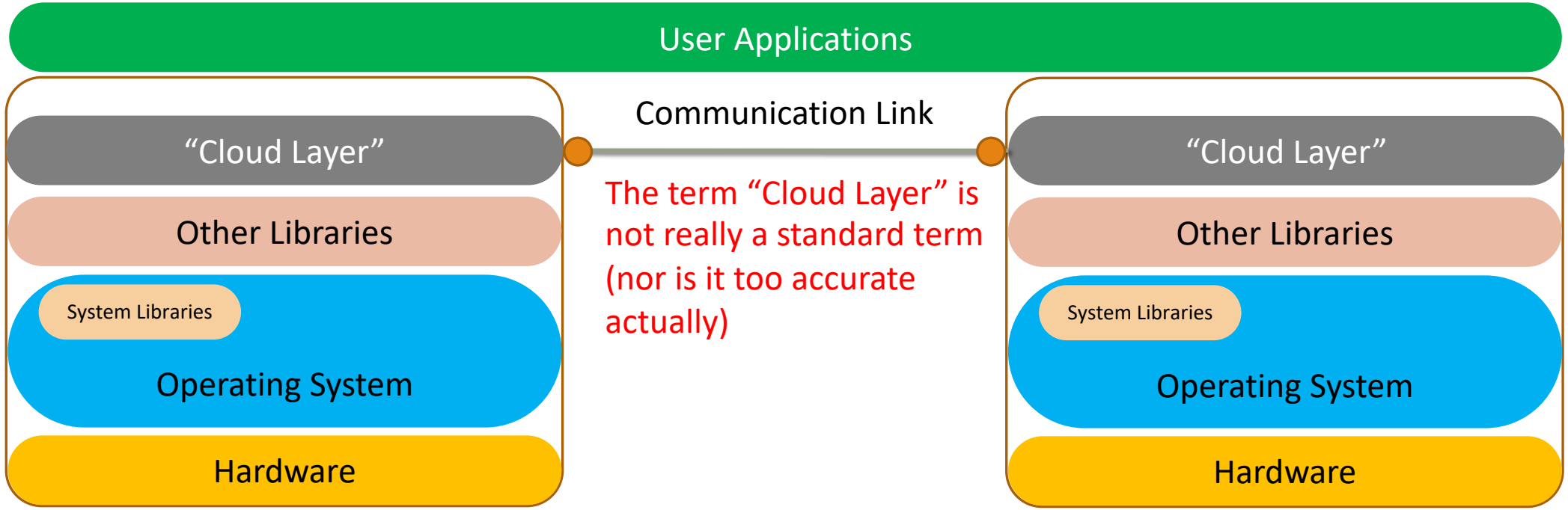


Computation on the “cloud”



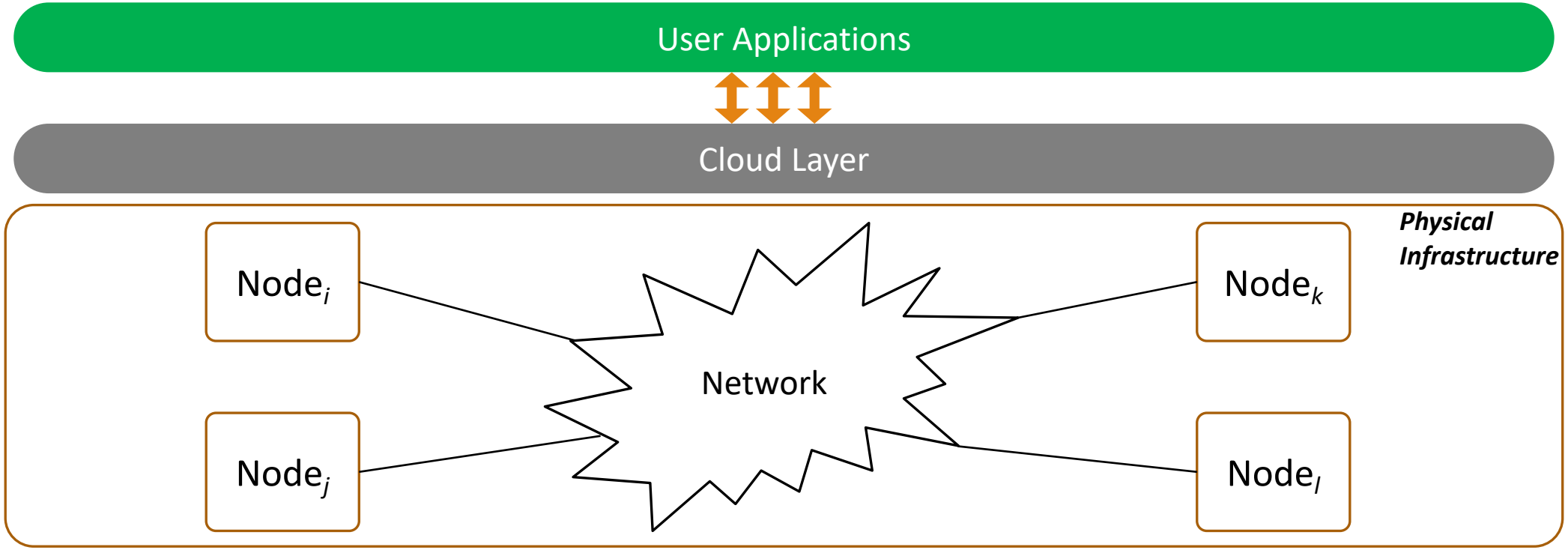
Computation on the “cloud”

We will discuss what this layering represents as we proceed



Computation on the “cloud”

Let us change our visualization slightly to make it look like below



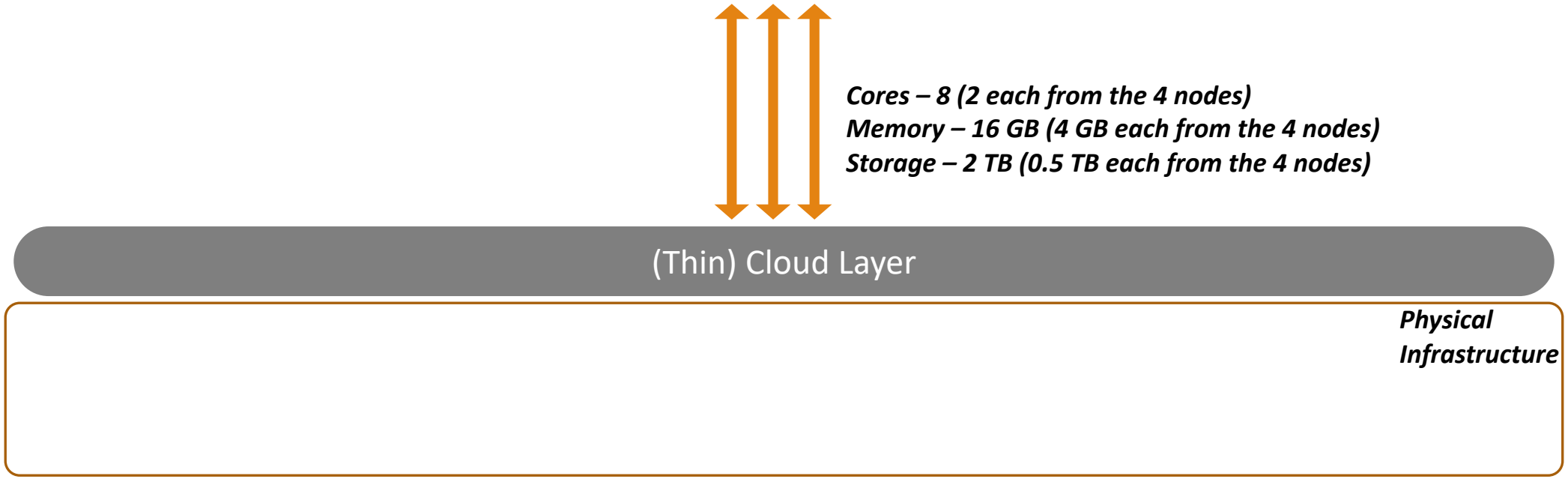
What can this “Cloud Layer” do for us?

At the bare minimum, this layer abstracts the physical hardware

- It means, you will neither know the number of nodes in the network, nor their individual configurations
- Instead, you see the underlying physical infrastructure as a *pool of resources*

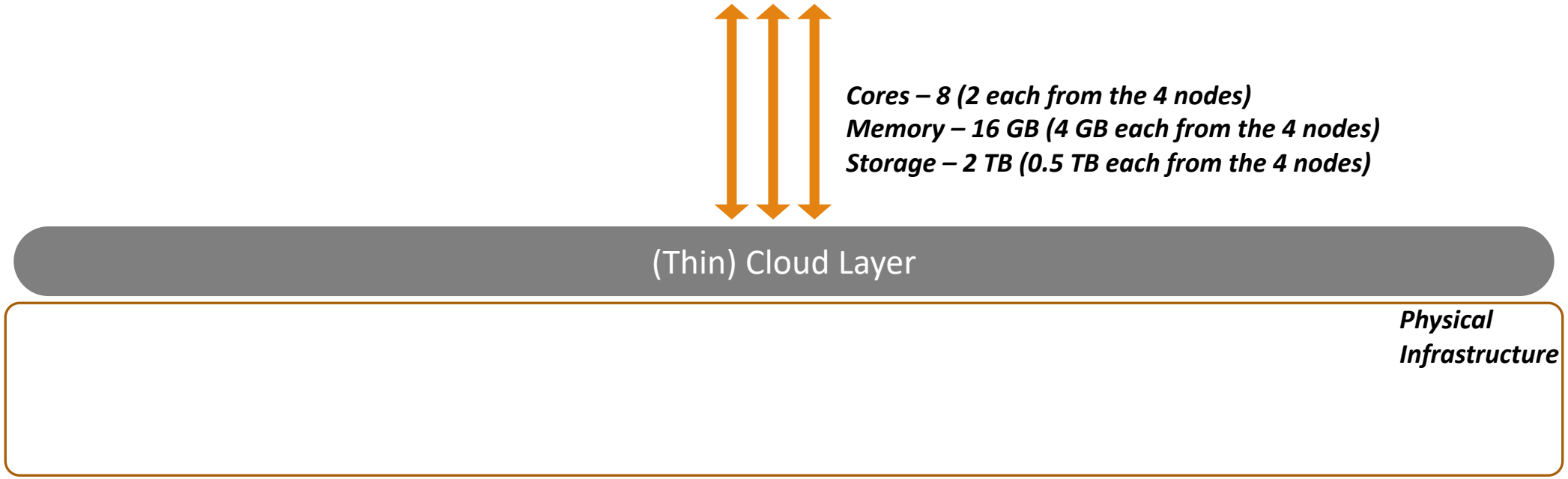
Offerings from the “cloud”

When this layer is “thin”, it provides the underlying hardware as a flexible pool of resources



Offerings from the “cloud”

This thin layer that is colloquially called as ***Infrastructure-as-a-Service*** or ***IaaS*** in short



What can this “Cloud Layer” do for us?

At the bare minimum, this layer abstracts the physical hardware

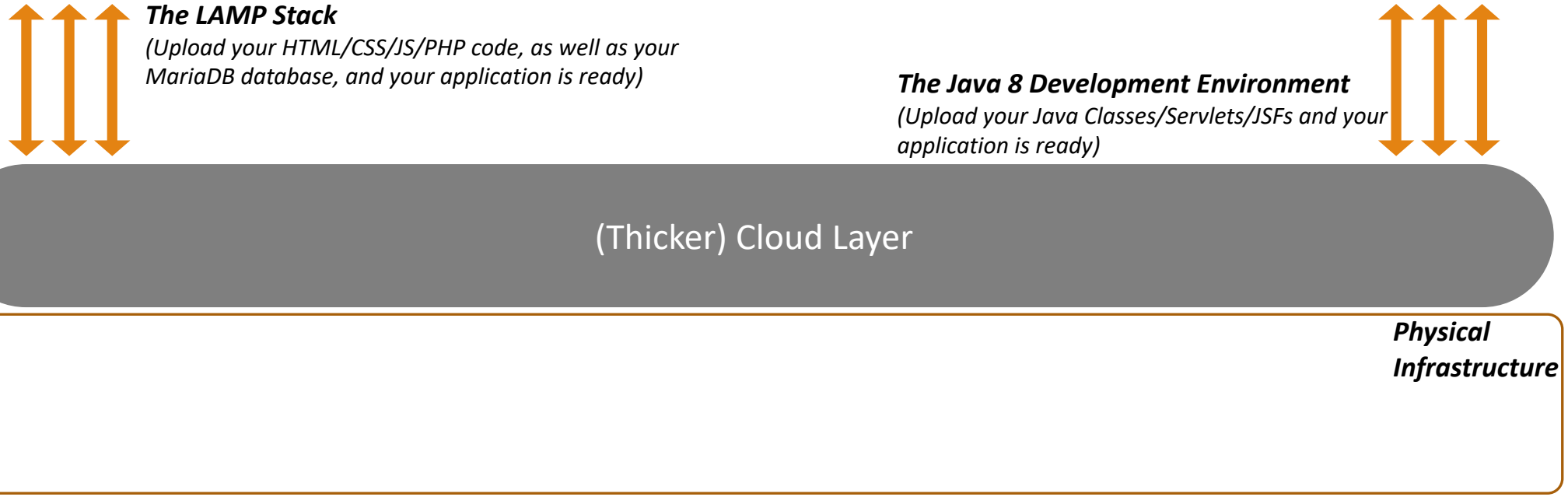
- It means, you will neither know the number of nodes in the network, nor their individual configurations
- Instead, you see the underlying physical infrastructure as a *pool of resources*

The layer can be made “thicker” – to provide even higher levels of abstraction

- For instance, it might provide you with a development or execution environment directly
- You can use this environment to create your applications without caring about the cores or memory

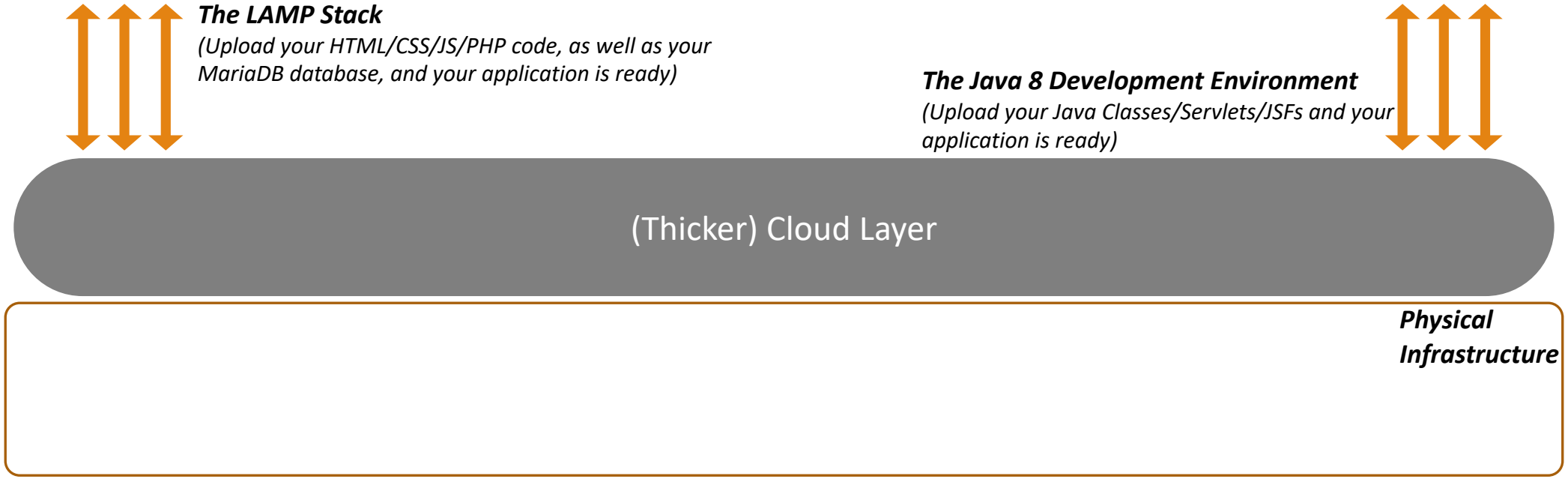
Offerings from the “cloud”

When this layer is “thicker”, it provides an abstracted development or execution framework



Offerings from the “cloud”

This version of the Cloud Layer is known as ***Platform-as-a-Service*** or ***PaaS*** in short



What can this “Cloud Layer” do for us?

At the bare minimum, this layer abstracts the physical hardware

- It means, you will neither know the number of nodes in the network, nor their individual configurations
- Instead, you see the underlying physical infrastructure as a *pool of resources*

The layer can be made “thicker” – to provide even higher levels of abstraction

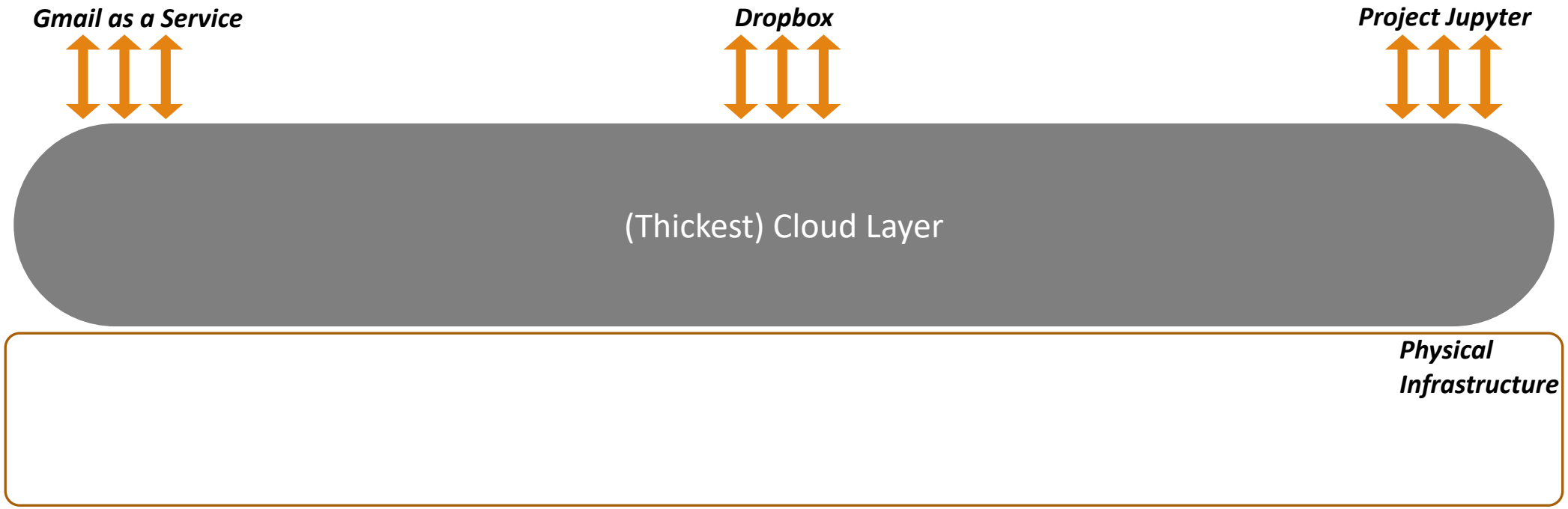
- For instance, it might provide you with a development or execution environment directly
- You can use this environment to create your applications without caring about the cores or memory

We can make this layer even more replete – to provide ready to use software solutions

- Examples of such solutions include Email Services, Cloud Storage Services, Code Notebooks etc.
- These solutions typically do not require any programming; just some configurations may be enough

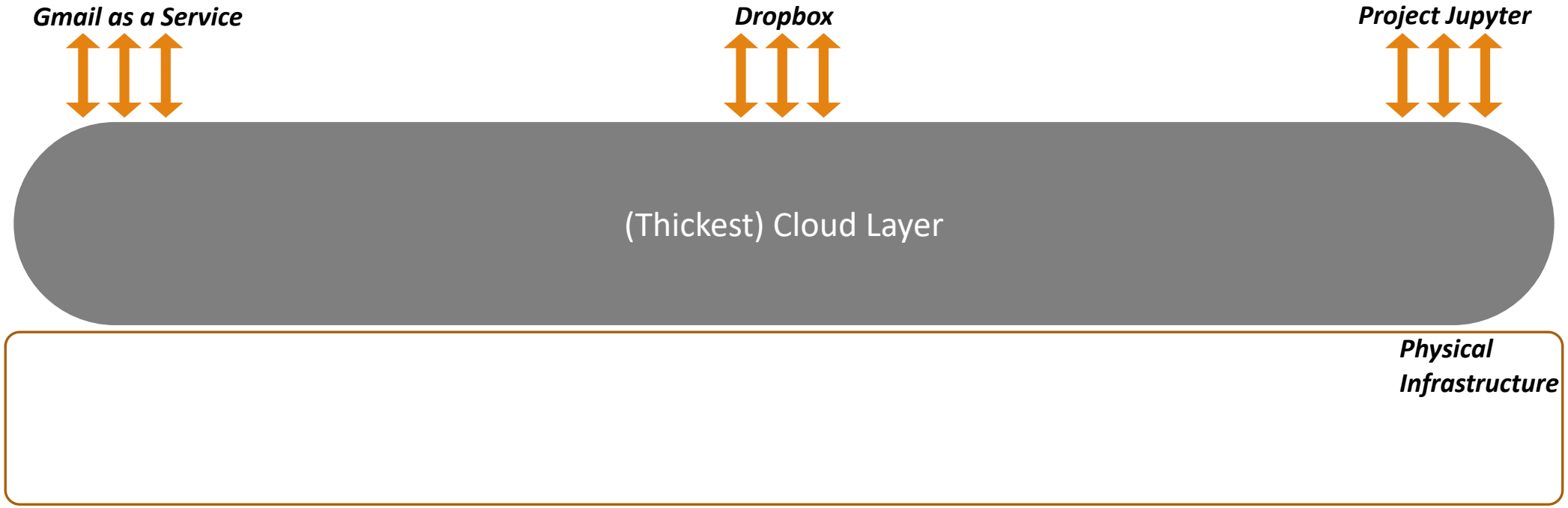
Offerings from the “cloud”

This layer may also provide ready to use solutions in its “thickest” form



Offerings from the “cloud”

This form of the Cloud layer is referred to as ***Software-as-a-Service*** or ***SaaS*** in short



What can this “Cloud Layer” do for us?

At the bare minimum, this layer abstracts the physical hardware

- It means, you will neither know the number of nodes in the network, nor their individual configurations
- Instead, you see the underlying physical infrastructure as a *pool of resources*

The layer can be made “thicker” – to provide even higher levels of abstraction

- For instance, it might provide you with a development or execution environment directly
- You can use this environment to create your applications without caring about the cores or memory

We can make this layer even more replete – to provide ready to use software solutions

- Examples of such solutions include Email Services, Cloud Storage Services, Code Notebooks etc.
- These solutions typically do not require any programming; just some configurations may be enough

In addition, this layer can be made to stretch across geographical regions

- It means that while $Node_i$ may be located in India, $Node_j$ may be stationed in the US
- As long as they may be connected via a network, the actual location of the nodes does not matter !!

A word on XaaS (Everything-as-a-Service)

The initial vision of cloud had the three service models that we discussed

- The idea was to provide flexibility to different kinds of users to pick a solution that suits them the best

However, as cloud became popular, service providers started providing more specific solutions

For instance, FaaS or Function-as-a-Service involves providing ”computation” as a service

- All the user needs to do, is provide a “function” to the cloud provider to execute
- The inputs and outputs are properly hooked to make it work (e.g., to events such as “creation of an order”)
- The user can then ”call” the function practically unlimited times, with certain SLA guarantees

DaaS or Data-as-a-Service involves providing a fully managed DBMS for use

- It involves providing access to a managed DBMS, where the quality is managed by the supplier
- The user can access/update the data via calls that look very similar to API calls
- The headache of managing (and scaling) the DBMS gets away if you choose a DaaS offering

Various Cloud Deployment Models

Private Cloud

- A cloud environment created for an organisation for its internal use
- Either managed by the organisation itself, or, by a third-party on its behalf

Community Cloud

- A joint venture of two or more organisations with some common interests
- Either managed by the organisations themselves, or, by a third-party on their behalf

Public Cloud

- A general-purpose cloud environment, created to be used by the general public
- Managed by cloud service providers (e.g., IBM or Amazon)

Hybrid Cloud

- A mixture of any of the above cloud models
- For instance, a private cloud may be the primary pool, with a public cloud as its backup for resources

How does the “Cloud Layer” work?

From our discussions till now, the “Cloud Layer” may have looked like black magic !!

- Afterall, how can so many physical nodes become a single pool by just connecting them through a network?

Infrastructure-as-a-Service *[Revisit]*

For now, let us assume that we are talking about IaaS

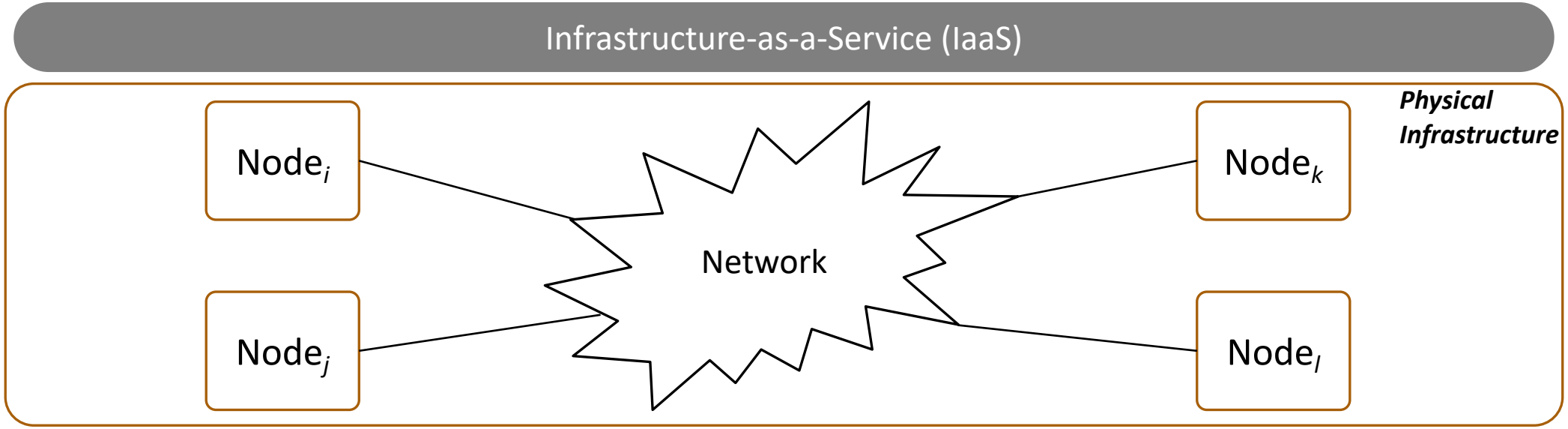
Cores – 8 (2 each from the 4 nodes)
Memory – 16 GB (4 GB each from the 4 nodes)
Storage – 2 TB (0.5 TB each from the 4 nodes)

Infrastructure-as-a-Service (IaaS)

Physical Infrastructure

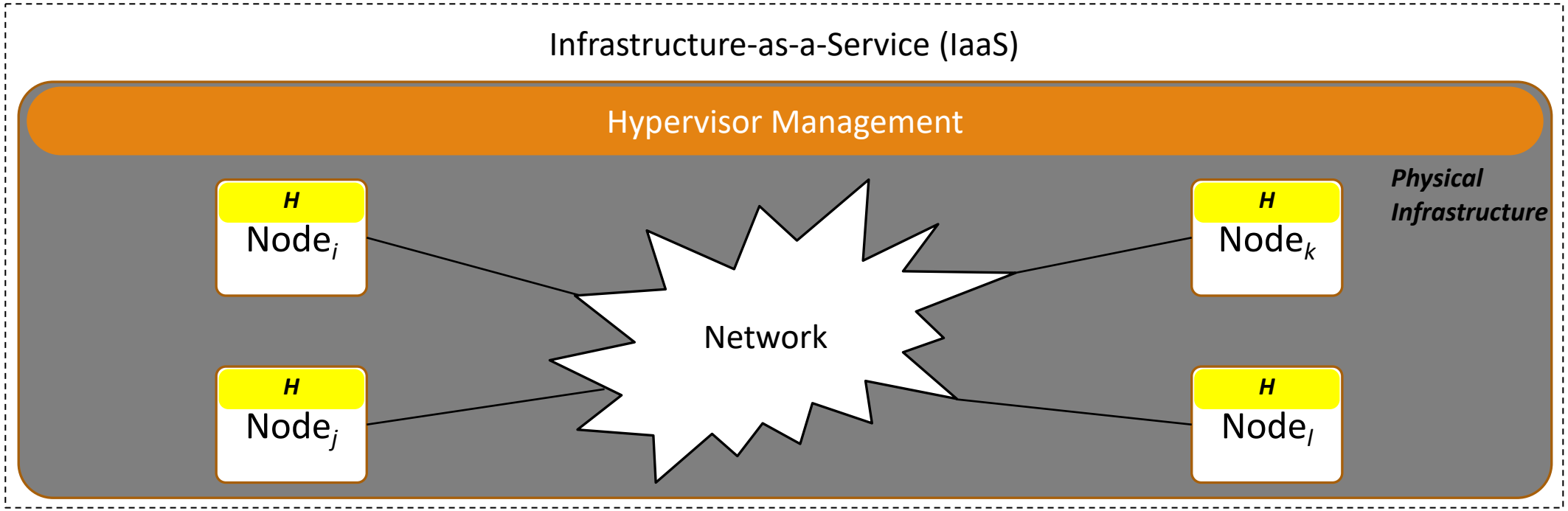
The Hypervisors

Let us blow up this image again !!



The Hypervisors

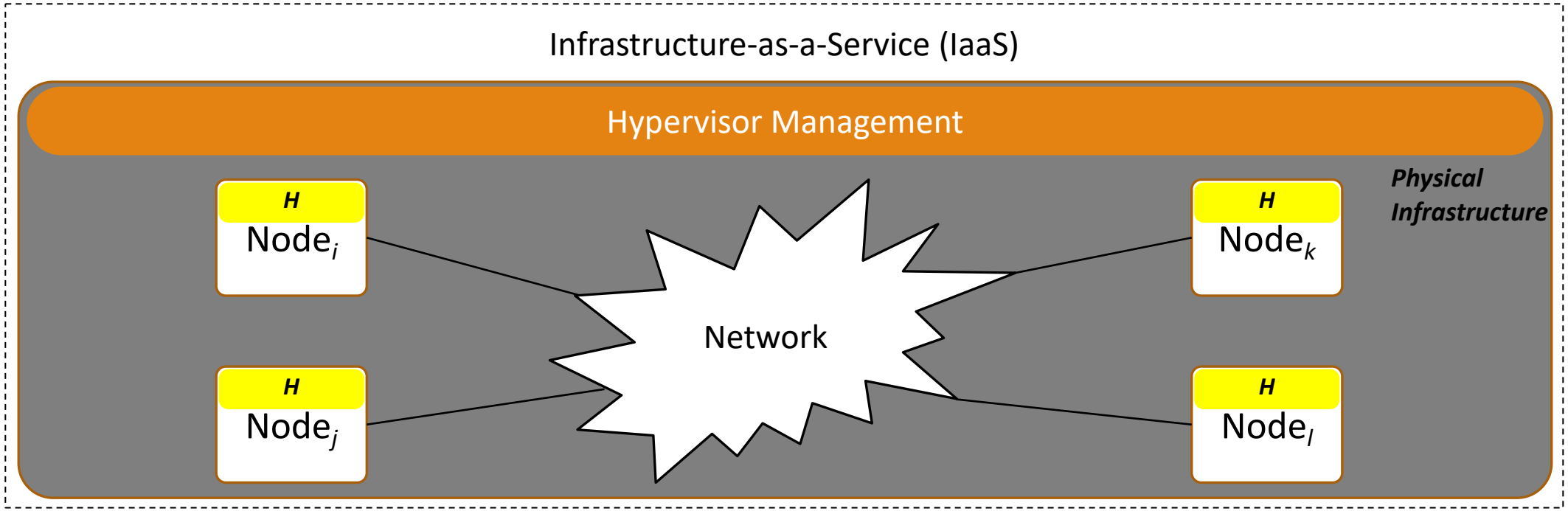
This is a more accurate depiction of an IaaS offering



The Hypervisors

This is a more accurate depiction of an IaaS offering

Here *H* depicts a Hypervisor



How does the “Cloud Layer” work?

From our discussions till now, the “Cloud Layer” may have looked like black magic !!

- Afterall, how can so many physical nodes become a single pool by just connecting them through a network?

A Hypervisor is a special software layer on every node in the physical infrastructure

- It is this layer that provides the required abstractions for creating the resource pool
- The hypervisors, along with some resource management tools, provide us a pooled view of the infrastructure

Hypervisors usually expose APIs and CLIs to communicate with them

- The IaaS management tools communicate with the hypervisors on individual machines
- These tools can instruct hypervisors to create virtualised resources in on the physical node they are operating
- Through the network access, the end user gets to access these resources without revealing the physical node

The resource pool is not really as “fluid” as it might have seemed before

- For instance, a VM provided by the IaaS layer cannot be more resourceful than what could be created …
- … on any of the physical nodes in the physical infrastructure

What is Virtualisation?

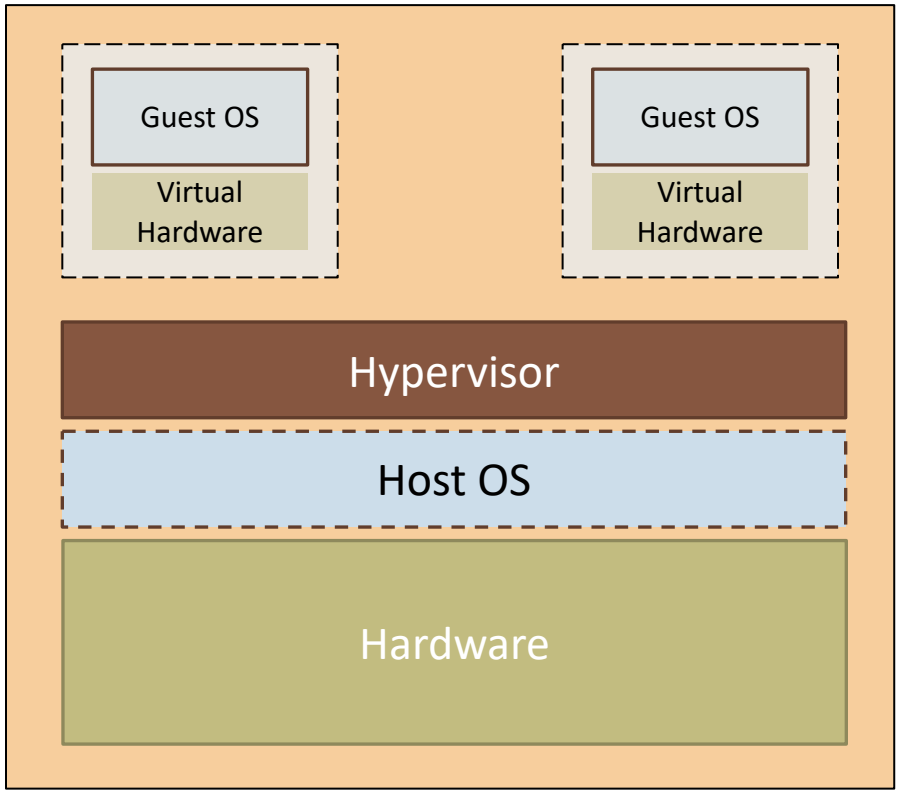
A Hypervisor or a Virtual Machine Monitor (VMM) is a software that abstracts hardware

- This abstraction allows the creation of an interface, which may be shared by multiple “virtual” machines
- The machines interact to the Hypervisor, even though, they believe they are interacting with hardware

A Hypervisor with multiple VMs

Physical Node

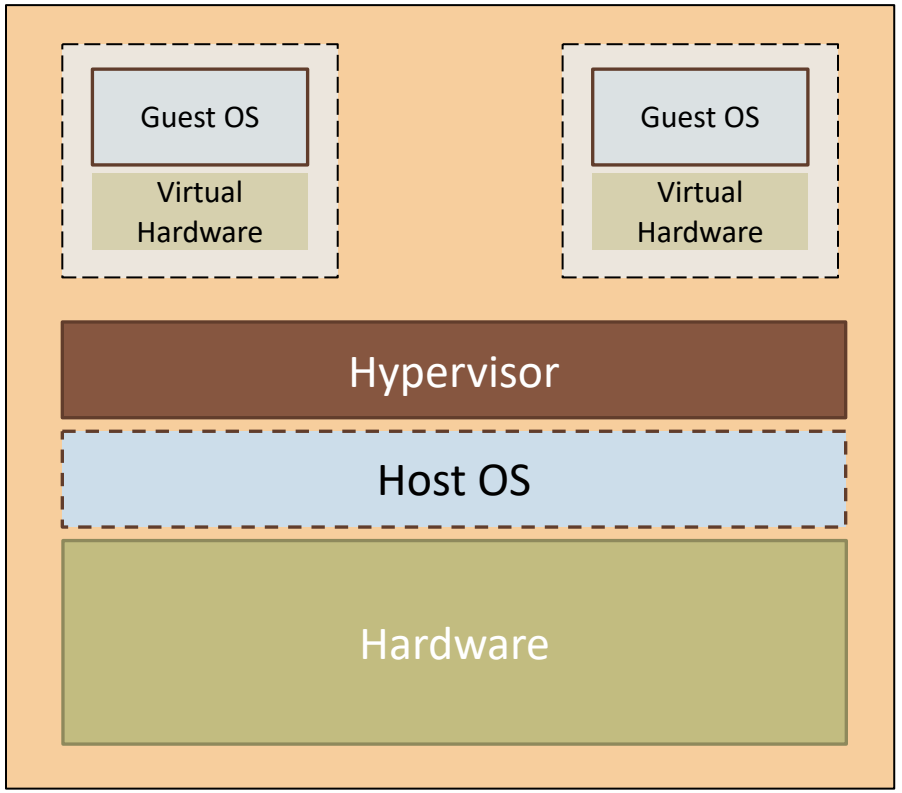
Virtual Machine



A Hypervisor with multiple VMs

Physical Node

Virtual Machine

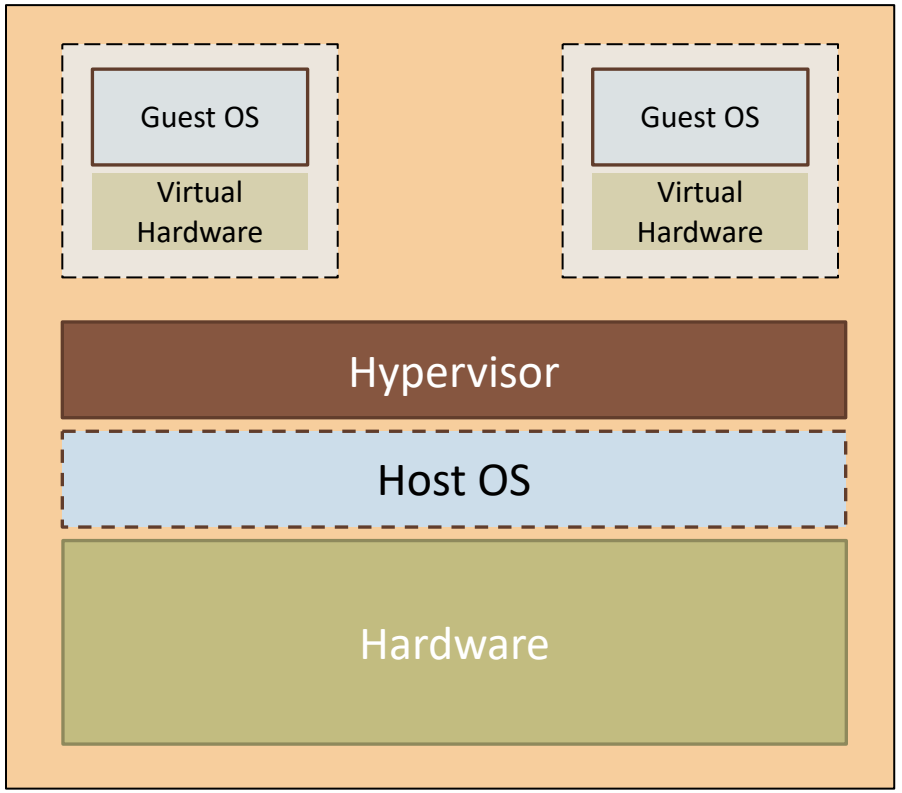


The Hypervisor makes the Virtual Machines believe that they are running over physical hardware

A Hypervisor with multiple VMs

Physical Node

Virtual Machine



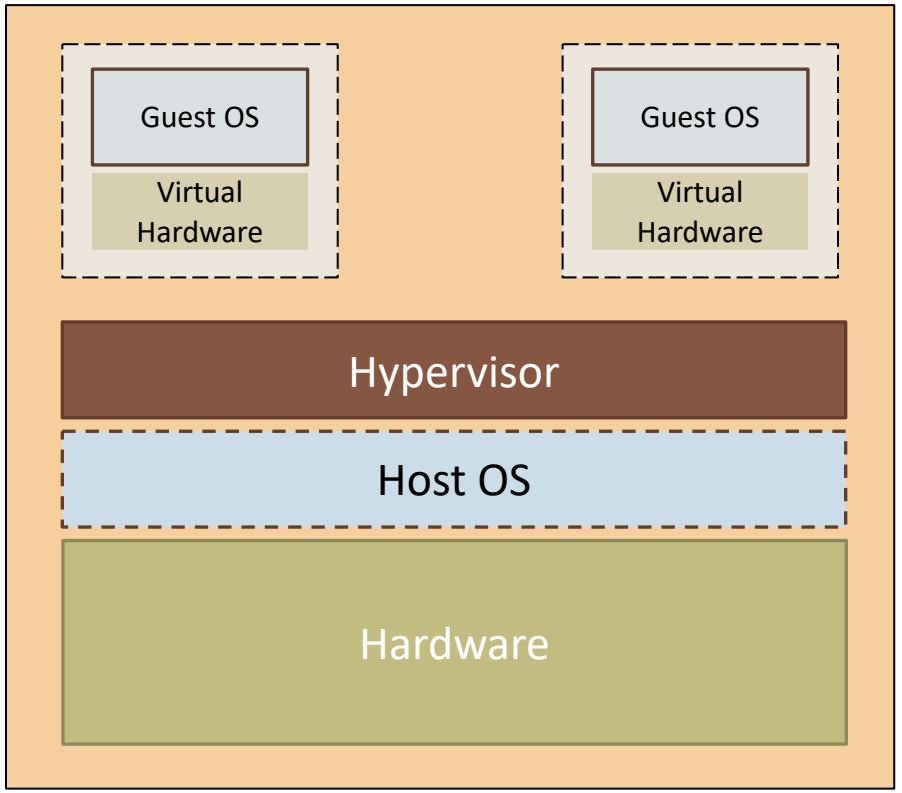
The Hypervisor makes the Virtual Machines believe that they are running over physical hardware

Even though, it may only see a small portion of the real hardware, and may share resources with other Virtual Machines on the same physical host

A Hypervisor with multiple VMs

Physical Node

Virtual Machine



The Hypervisor makes the Virtual Machines believe that they are running over physical hardware

Even though, it may only see a small portion of the real hardware, and may share resources with other Virtual Machines on the same physical host

The *Host OS* may or may not be present in the stack (we'll discuss this in a moment)

What is Virtualisation?

A hypervisor or a Virtual Machine Monitor (VMM) is a software that abstracts hardware

- This abstraction allows the creation of an interface, which may be shared by multiple “virtual” machines
- The machines interact to the Hypervisor, even though, they believe they are interacting with hardware

Ideally, a hypervisor should exhibit three important properties

- **Fidelity:** The interface exposed to the VM should be, in principle, identical to the physical hardware
- **Isolation or Safety:** The VMs must only be given controlled access, approved by the hypervisor
- **Performance:** The difference between the performance of a VM and a comparable physical machine is negligible

While virtualisation is an independent concept, it is essential for cloud environments

- For example, there are solutions which could be used to create VMs on individual machines for personal use
- However, in the context of cloud, usually, a single physical machine hosts multiple VMs
- *Condensing* multiple servers on to one (multiple VMs on Physical Host) is called *Consolidation* …
- … and the number of servers condensed is called *Consolidation Ratio* (e.g., if 8 VMs run on one machine, it is 8:1)
- Powerful servers today may have consolidation ratios of the order of a hundred !!

Types of Hypervisors

A hypervisor could either execute over an Operating System, or, directly over the hardware stack

- The image we saw showed the *Host OS* in a greyed box, essentially meaning it "may or may not be there"

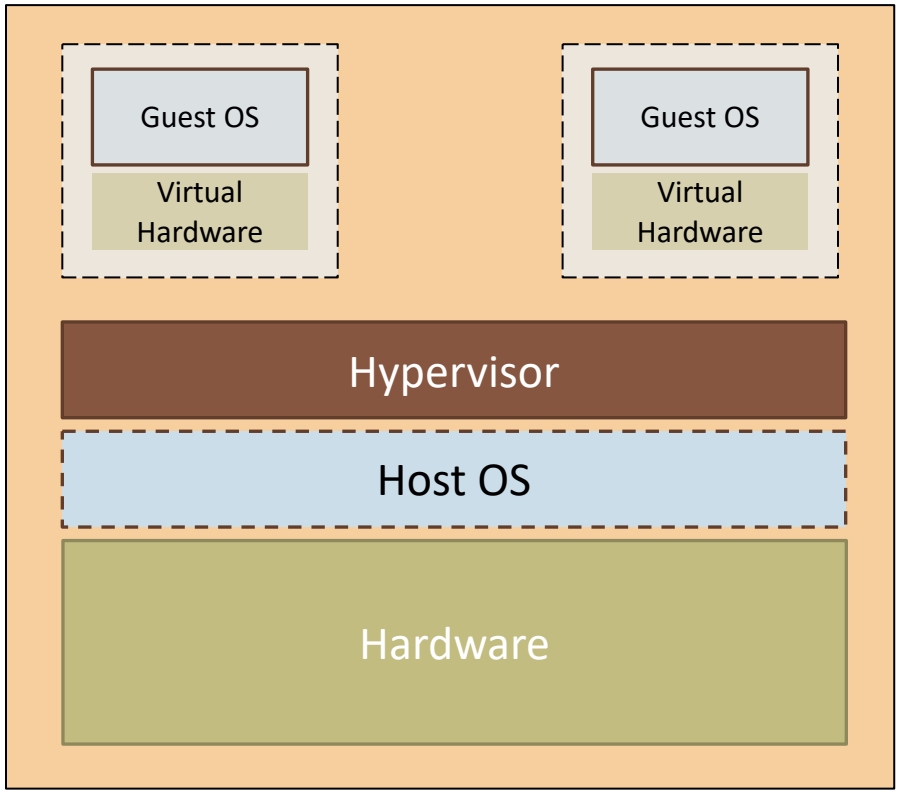
Type 1 hypervisors are those who work without an intermediary

- They execute directly over the hardware and interact directly with the physical resources
- The lack of an intermediate layer, makes Type 1 hypervisors more efficient than their counterparts
- Type 1 hypervisors are also considered more secure (since vulnerabilities in host OS may result in exploits)

A Hypervisor with multiple VMs

Physical Node

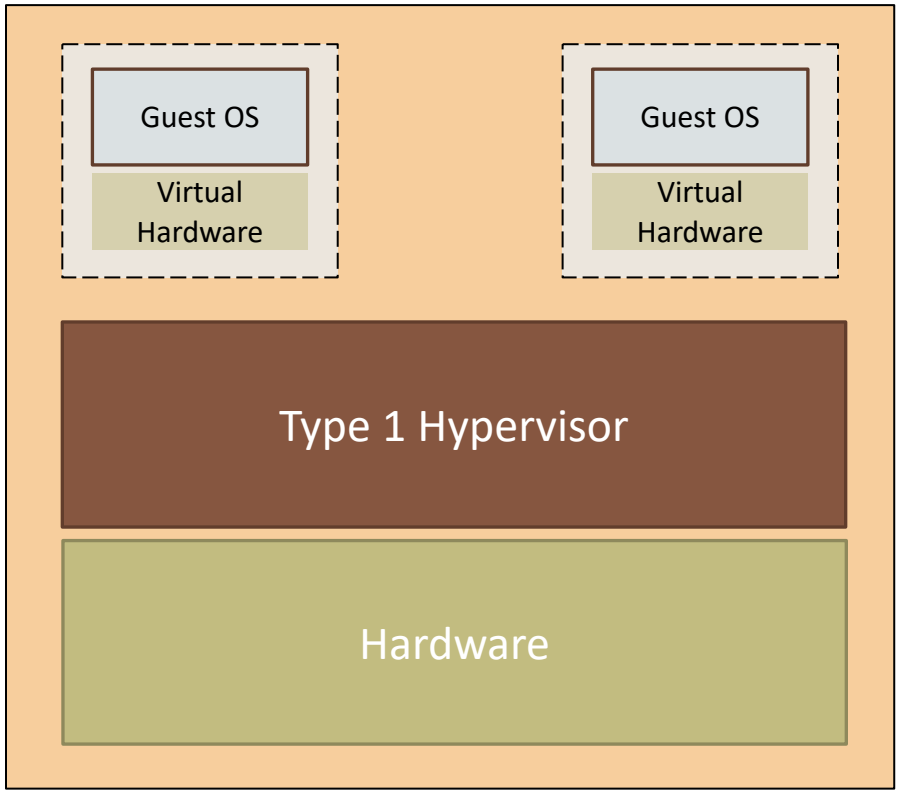
Virtual Machine



A Hypervisor with multiple VMs

Physical Node

Virtual Machine



Type 1 hypervisors operate directly over the hardware

Types of Hypervisors

A hypervisor could either execute over an Operating System, or, directly over the hardware stack

- The image we saw showed the *Host OS* in a greyed box, essentially meaning it "may or may not be there"

Type 1 hypervisors are those who work without an intermediary

- They execute directly over the hardware and interact directly with the physical resources
- The lack of an intermediate layer, makes Type 1 hypervisors more efficient than their counterparts
- Type 1 hypervisors are also considered more secure (since vulnerabilities in host OS may result in exploits)
- Examples of Type 1 hypervisors are VMWare ESX and Microsoft Hyper-V

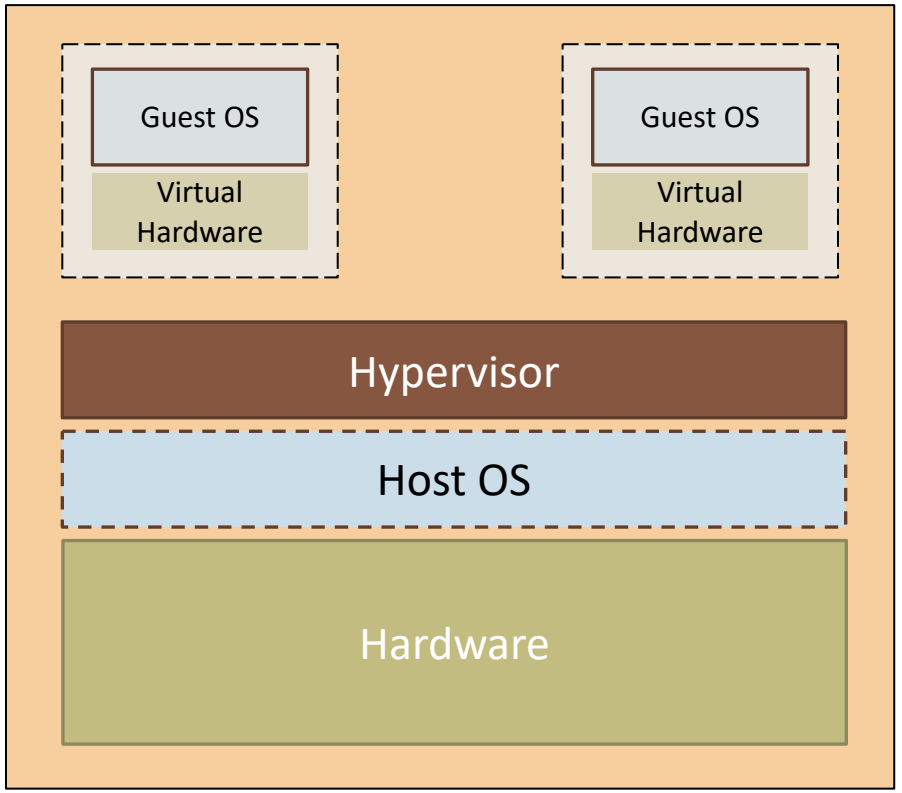
Type 2 hypervisors run over an OS installed on the physical machine (we refer to it as the host OS)

- They operate very much like any other process on the host OS
- They are usually much easier to install (it is akin to installing any other software or package)
- This ease of use comes with the trade-off of inferior performance and more security vulnerabilities
- Examples of Type 2 hypervisors are VMWare Workstation and Oracle Virtualbox

A Hypervisor with multiple VMs

Physical Node

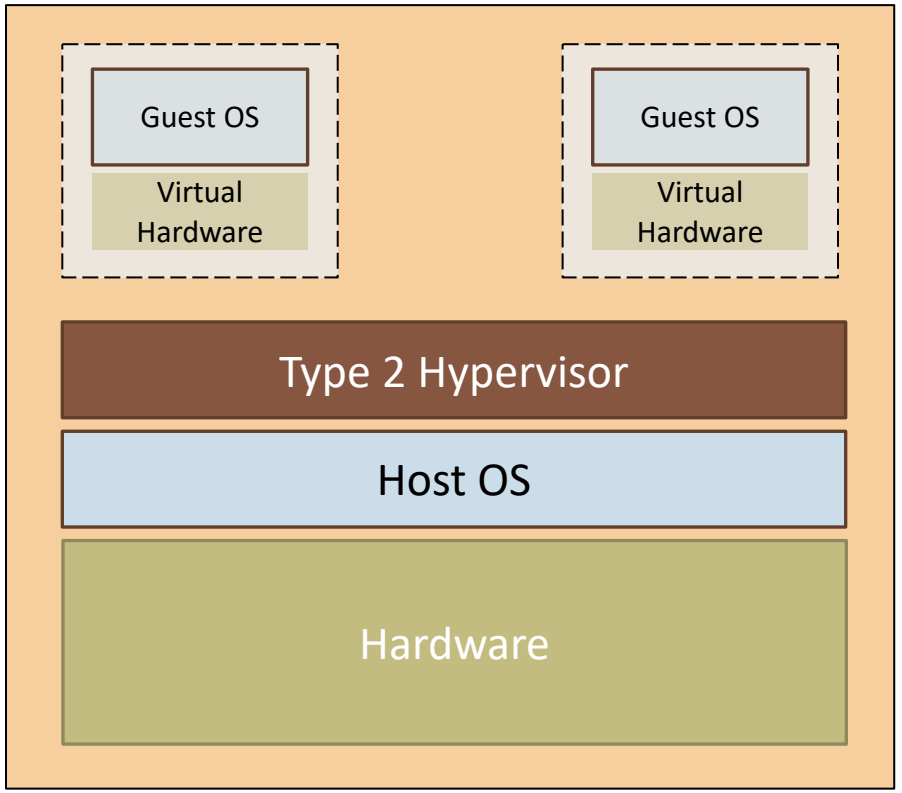
Virtual Machine



A Hypervisor with multiple VMs

Physical Node

Virtual Machine



Type 2 hypervisors run over a host Operating System, like any other process