CMPSCI 677 Distributed and Operating Systems **Spring 2025**

Lecture 10: March 05

Lecturer: Prashant Shenoy Scribe: ***Harshita Vidapanakal (2025), Pranav Pratap Singh (2024)***

10.1 Datacenters

A datacenter is a facility where a large number of servers along with lots of storage run multiple applications(server farm).

A common way to image a Datacenter is a supermarket where shelves are composed of servers.

Datacenters generate a lot of heat and they need cooling infrastructure. They consume lots of power too(electricity).

10.1.1 Architectures

- Traditional:
  - Applications run on physical machines.
  - Manual management of server by System admins.
  - Uses SAN and NAS to hold data.
- Modern:
  - Applications run on VMs.
  - Uses Type1 hypervisor.
  - Pre and post copy migrations is done on VMs.
  - VMs can be resized and migrated.
  - Allows automation.

10.1.2 Virtualization in Data Center

- Virtual Servers Example are consolidated servers where you can replace N old servers with 1 new server .
- Virtual Desktop

  Example: PC in the Cloud - In an enterprise, rather than giving powerful hardware to the employees, employees get a thin client that connects to a powerful VM in the cloud.

Question: How migrate-able is GPU as a resource?
Answer: We can virtualize a GPU and slice it into virtual GPUs (for example, NIVIDIA MIG) but this is fairly recent and can be expensive. Old GPUs do not have this feature and have to be used fully.

10.1.3 Data Center costs

Expensive to run a datacenter.
PUE (Power Usage Effectiveness) = Total Power/IT Power

typically PUE is 1.7
Google's Datacenter has PUE about 1.1

10.2 Cloud Computing

Cloud Computing is where servers and storage are going to be leased by the customer. Cloud providers allow you to rent on demand, not only on an advance, pay-as-you-go model.

Benefits:

- Remote access from anywhere
- Pay as you go
- Highly scalable

Question: Do cloud providers have the ability to handle peak loads?

Answer: They have the instinct to develop the most efficient data centers.

10.2.1 Types

As classified by the block of resource received by the cloud provider

- Infrastructure as a Service (IaaS) : Machines/VMs with network access
- Platform as a Service (PaaS): Platform to run the applications by the provider
- Software as a Service (SaaS): Hosted application by the provider.

10.2.1.1 IaaS

Example: Amazon EC2 - Rents servers and storage to customers

- Leverages virtualization to share each server for multiple customers
- Economy of scale lowers price

Types of IaaS instances:

- On-demand instances: Provisioned on the fly and follow a pay-by-the-minute model. They are maintained until servers are requested to be terminated.
- Reserved instances: User reserves an on-demand server for a long period of time (1 year, 3 years). This commitment provides discount over on0demand pricing.

- Spot Instances: Excess capacity is rented at a high discounted price. However, the access can be revoked by the provider with a very short warning (eg. in the instance where regular customers need the server). Is a cheap method to run large computations in off-peak periods.

10.2.1.2 PaaS

- Cloud resources are offered as a scalable runtime platform. Platform is responsible for deploying, provisioning resources and autoscaling the application
- Developers provide only code. Platforms support code in multiple languages such as Python, Java, Node, .NET
- Billed based on usage/workload (Eg. Number of requests for a web server)
- Eg. Google App Engine, Microsoft Azure

10.2.2 Serverless Computing

- Has similarities with PaaS: Platform is responsible for scaling, deploying and developer is responsible only for the code.
- Difference: Code is written as FaaS (Funtion as a Service) - deployed at function-level
- Multiple functions can be chained together
- Functions are often stateless. Need to write to DB to persist information
- FaaS hence more fine-grained than microservices
- Example: AWS Lambda - FaaS in the cloud

10.2.3 Cloud Models

- Public - The cloud provider owns the resources. Enterprices suspicous of the security.
- Private - Org builds its own private cloud.
- Hybrid - Exceess private load is put into public clouds.

Question: Is a combination of servers on a private cloud and storage in a public cloud considered hybrid?

Answer: Yes, it is. However, the general convention is to have the data stored in the private cloud and use public cloud resources for server/compute.

Question: Where does UNITY service fall into in this classification?

Answer: UNITY does not fall into any of these categories. It is a compute cluster provider which uses distributed scheduler to service jobs. It allows to run only batch jobs.

10.3 Kubernetes (k8s)

Container orchestration is a form of cluster scheduling but rather than scheduling jobs or http requests you're scheduling containers. There is a pool of machines and applications are coming as containers. The goal of the container manager is to now assign this container onto some physical host. When the application is done, the container is terminated. The scheduler does not care about what is inside the container, it just schedules the container on the basis of its resources requirements.

Kubernetes is one of the most popular container orchestration systems. It is based on Google's Borg/Omega cluster managers. In Kubernetes, it is assumed that all applications are containerized. K8s will deploy them onto machines of the cluster. Kubernetes provides the following features:

1. Replication of apps on multiple machines if requested (fault tolerance)
2. Load balance across replicas
3. Can scale up or down dynamically (vary replica pool size, a concept similar to dynamic thread/process pools)
4. Provide automated restart upon detecting failure (self-healing)

10.4 K8s Pods

Kubernetes has the concept of pods. A *pod* is an abstraction where we can have more than one container in it. Containers inside pods share volumes and namespace. Kubernetes doesn't directly deal with containers, it deals with pods. So, pods are the smallest granularity of allocation in k8s.

In a distributed application, because your application has multiple components in the kubernetes world each component has to be containerized first. So, each pod consists of one or more components/containers.

Pod can contain all containers of an application but if a component needs to be scaled, put that component in a separate pod. As a good design principle, each independently scalable component should be put in a different pod. Constructing applications in pod is the job of an application developer. Deploying and scaling it is the responsibility of kubernetes. Pods of an application can span multiple cluster machines.

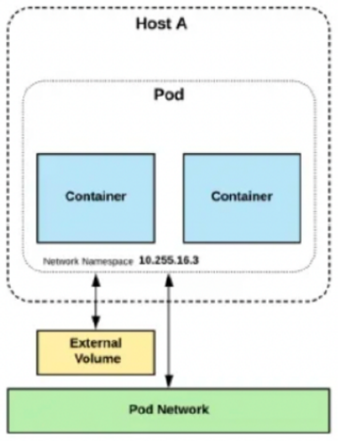

Figure 10.2: Visualization of the relationship between containers, pods, and hosts.

10.5 K8s Services

Pods are used to construct kubernetes services. A service is a method to access a pod's exposed interfaces. Features of services include:

1. Static cluster IP address
2. Static DNS name
3. Services are not ephemeral
4. Collection of pods

Pods are ephemeral. Each has its own IP. They can be migrated to another machine. Pods can communicate with one another using IP.

Question: Can a component run in multiple containers?

Answer: Yes, K8s supports replication.

Question: Is replication done by the user? Or does Kubernetes do it automatically?

Answer: User specifies if replication is allowed and configures the number of replicas. However, orchestration is performed by K8s.

10.6 Control Plane

- apiserver: REST interfaces for clients to access management interface
- etcd: cluster key-value datastore. Strongly consistent, highly durable (uses RAFT consensus)
- controller-manager: replicate pods, monitor for node failures and restart
- scheduler: Assigns newly created pods to servers based on resource constraints
- cloud-controller-manager: Interacts with cloud platforms