- Today, most of the compute & storage intensive work goes on in networked servers.
- There is hardly any work on a large scale that happens on local computers.
- Everything is moving to the cloud.
- All large companies have their own data centers for processing data, or use public cloud offerings.
- All the data present in public clouds like Amazon AWS, Google Compute Engine or Microsoft Azure, resides on VMs. Virtualization is in the heart of public cloud offerings.
- In this class, we will look at the virtualization offered by the public cloud providers, such as Google & Amazon.

- We will see what these companies offer in terms of VMs, containers, serverless, etc.
- Then we will look at how major companies like netflix / twitter / uber use the services for all of their data processing.
- This class will serve as a motivation for the other classes, & will talk about the usefulness & impact of virtualization, before looking into "how" to virtualize in the remainder of the semester.

Timeline:

1. VMs.
2. Containers.
3. Unikernels
4. Serverless
5. New VM technologies.

Twitter ad-analytic framework: managed by Google cloud.

- GCP: VMs

Spot VMs (GCP)

- can be used to create VMs that are fault tolerant.
- The VMs are created on-demand. They are allocated resources when there are available idle resources.
- If the load on servers becomes high, the spot VMs are either paused or killed.
- User applications running on spot VMs should be fault tolerant
- Not good if there are tight SLOs. The tail latency becomes high due to spot VMs.
- 91% lower cost than general-purpose VMs.
- Used by maps, YouTube, etc

- Harvest VMs. (Azure)
  - More flexible than Spot VMs.
  - The resources in HarvestVMs grow & shrink according to the unallocated resources in underlying physical machine
  - Does not get killed. Only killed if the resources go below the minimum required resources of the VM.
  - More utilization of data centers
- Burstable VMs.
  - VMs that are allocated some minimum amount of resources, but that can get more resources - on-demand in a best-effort manner if the application load increases.

→ Ballooning

→ Serverless

Google : Cloud Functions

Amazon : AWS Lambda

Microsoft : Azure Functions.

Functions that run in the cloud offer Faas. You just need to write a function, such as a serverless web app. Then it is associated with an API. The function gets called on the API, & runs in the cloud.

Example :

Coca-Cola used AWS Lambda

use QR-code on phone to scan the coca booth. Automatically shows UI with options. On click, the AWS Lambda function is executed, & drink is poured in the cup.

Only pay for the invocation of function. Payment happens at millisecond granularity.

- Disadvantage:
  No state. It needs to be stored in separate db outside of application.
  Function needs to be self-sufficient in the data it uses. cannot rely on global variables, or data of other functions. Hard in some cases.

Durable Functions: : State ful serverless functions

In some scenarios, Azure manages to hold state for your functions & hence makes it easy for us to use serverless in real world.

VMs

GCE: 2012

Amazon EC2 : 2006

Microsoft Azure : 2008

Charged on per-minute usage.

Containers

Kubernetes: 2014

Docker: 2013

GKE charged in $0.1 per cluster per hour.

Serverless

2017: Google Cloud functions.

- Developer in company.
  - Job: long period of time. computing something
  - How?
    - either buy lot of machine.
    - Amazon EC2.
      - Azure.
      - with price etc.
    - only billed while time is running.
  - Job: not all the time. Only get erratically. 10 times a month.
    - → EC-2. Big cost of keeping it running
    - serverless. Info. from external sources. Bill lower.
  - Containers. Developer working in company. Just send depending.

pre-virtualization. (10 yrs ago.)

1st Jump:

→ on-prem machine. single machine
→ on-prem. V.M.
→ cloud

Each thing. : How people did before then.

Connect to parts of class.

- Last time: history of virtualization.
- We talked about timeline of when VMs introduced, containers introduced & serverless introduced
- Today: motivate virtualization, look at real-world examples & impact of virtualization.

→ Imagine that we have an IT company
  - Accounting dept
  - Developers - backend
  - Marketing dept.
  - Developers - front-end.

- In 90s, there were no machines.
- Every employee: one computer with own monitor, keyboard..
- Each computer: $1000 - $2000 minimum.
- IT support staff: manage s/w on computers, update O.S, etc.
- Initial startup investment: very high.
  ↳ 10s of 1000's of $.

1st Jump of Virtualization.

- on-prem VMs.
- Instead of separate desktops, server room with all machines.
- VMs for every employee depending on requirements.
- Underlying physical machine: manage CPU, mem., storage & networking across VMs.
- Every employee didn't need a separate server.
- However, IT support staff was still needed, to support the on-prem servers.
- Half the servers needed.
- Inventory cost.

Look at when we learn about VMs next week

IInd Jump → CLOUD.

- There was no physical resource needed.
- Simply launch a VM in the cloud, using Amazon EC2, Google cloud platform or Microsoft Azure.
- No IT support staft needed.
- Very low initial investment on resources. Low overhead to start a company.

- In public cloud, almost everything must run in a virtualized setting.

→ Financial year-end = Calculate profits & revenue. e.g. Google.
- Run the software for multiple days & do lot of computation.
- Because of 2nd Jump of virtualization:
→ Launch Amazon EC-2 VMs, costing total be done with it, instead of paying & maintaining for own machines.
- You get paid for every second use of the VM, & it is maintained & manged by

+3 medium: 2vCPUs, 4GB mem, 100 GB SSD: 42$ /month
16 vCPUs, 32 GB mem 200 GB SSD: $252/month. 34¢/hr or $8.3/day.
6¢/hr. or $1.4/day

Cloud providers.

— In fact to reduce costs even further / tailor to our needs, different VMs are present.

① spot VMs
② Harvest VMs
③ Burstable VMs.

→ Virtualizing Memory class. Ballooning

IT team comes up with a complex software product, that depends on a number of packages & their versions, such as python 3, JDK v8.7.2 → 100MB of space, 500 MB DRAM

→ old days: no virtualization: ship the package / sell the software license

— customer m/c different, & would fail.
— days of debugging required.
— The software is computation heavy, but requires only about 100 MB of memory.
— ship it in VM, of 10s of GBs.
— slow to start-up, too heavy.
— Any bugs, re-install VM at the client. end.

— CONTAINERS

- package the software & dependencies, ship to client
- client can run at their end.
- Any crash, same environment & easily reproducible
- Container orchestration frameworks:
  Google Kubernetes Engine.
  Platform-as-a-service.
- Put container & orchestrate with GKE. It places your containers & executes in the cloud on its own
- you don't need to worry about VMs or anything.
- You only mention the # CPUs, mem. & storage needed for app. GKE manages the rest.
- We will learn about containers after VMs.

- Netflix

→ Come up with a software that is only called erratically. Probably 10 times a month, it gets invoked.

e.g. V

- Having a VM running all the time causes you to pay for the VM all year.
- Same with containers.

- Serverless: write the service, stores data in Amazon S3.

what is Amazon S3.

- Write an API for the service.
- Push it to AWS Lambda.
- Whenever API is called, it is run on AWS Lambda.
  - No VM
  - No Container.
  - Function-as-a-service.
  - $1.37/month only

↳ We will learn about serverless towards the end

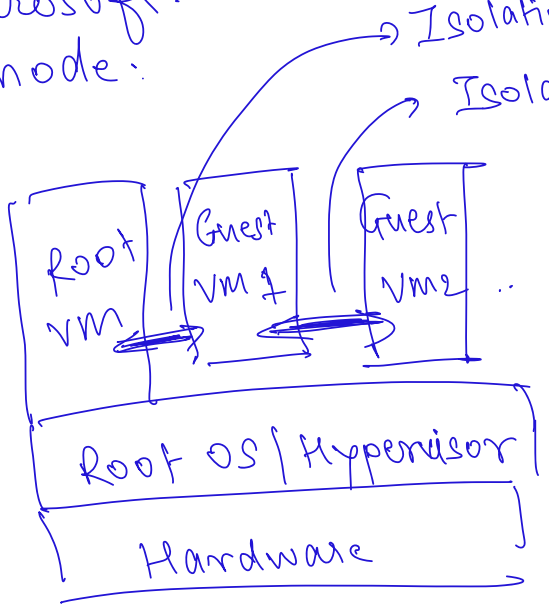
→ COCA-COLA

↳ Coca-cola free-style.

MICROSOFT AZURE

Uses hypervisor named Hyper-V by microsoft.

- Each node:



cross VM attack:

- when victim VM & adversary-controlled VM are on the same host, & adversary-controlled VM snoops or causes slowdown of victim VM.
  - side-channel attacks

Fabric Controller → Sophisticated placement of VMs on hosts. VMs don't know other co-existing VMs on the same host.

- **Root VM**
  - ↳ Special VM that runs special OS

AWS Nitro

September 2020.

Offload virtualization functions to dedicated hardware & software.

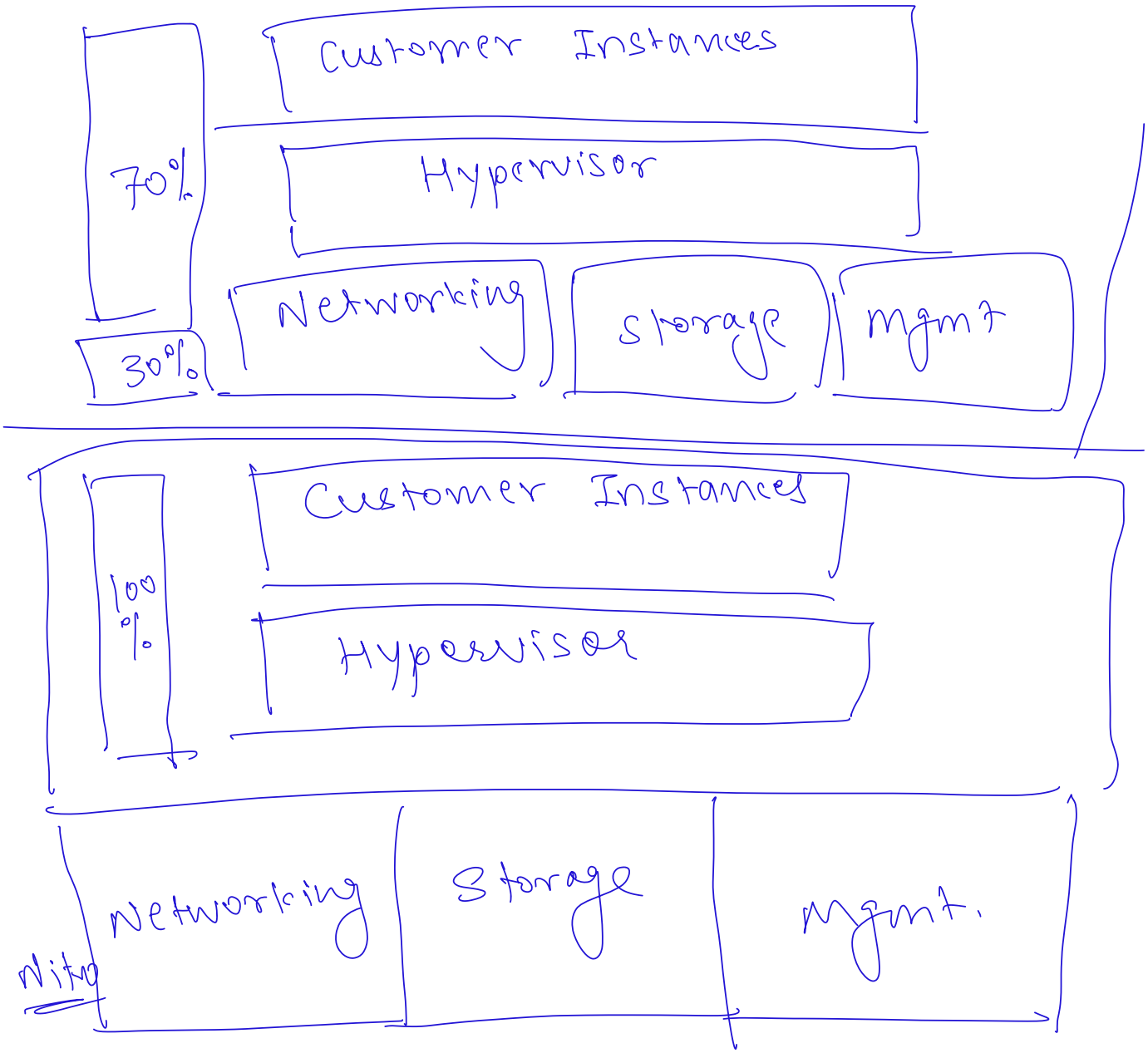
In early days of EC2, Xen hypervisor was used & was modified to support multi-tenancy, to increase utilization & reduce costs.

However, with Xen, as much as 30% of the resources in an instance were allocated to hypervisor.

Nitro components :

1. Nitro cards : Accelerates IO for functions, such as instance storage.
2. Nitro Hypervisor : Complete hypervisor offloaded to hardware.

Nitro enabled Amazon to make hypervisor layer optional & offer baremetal instances.

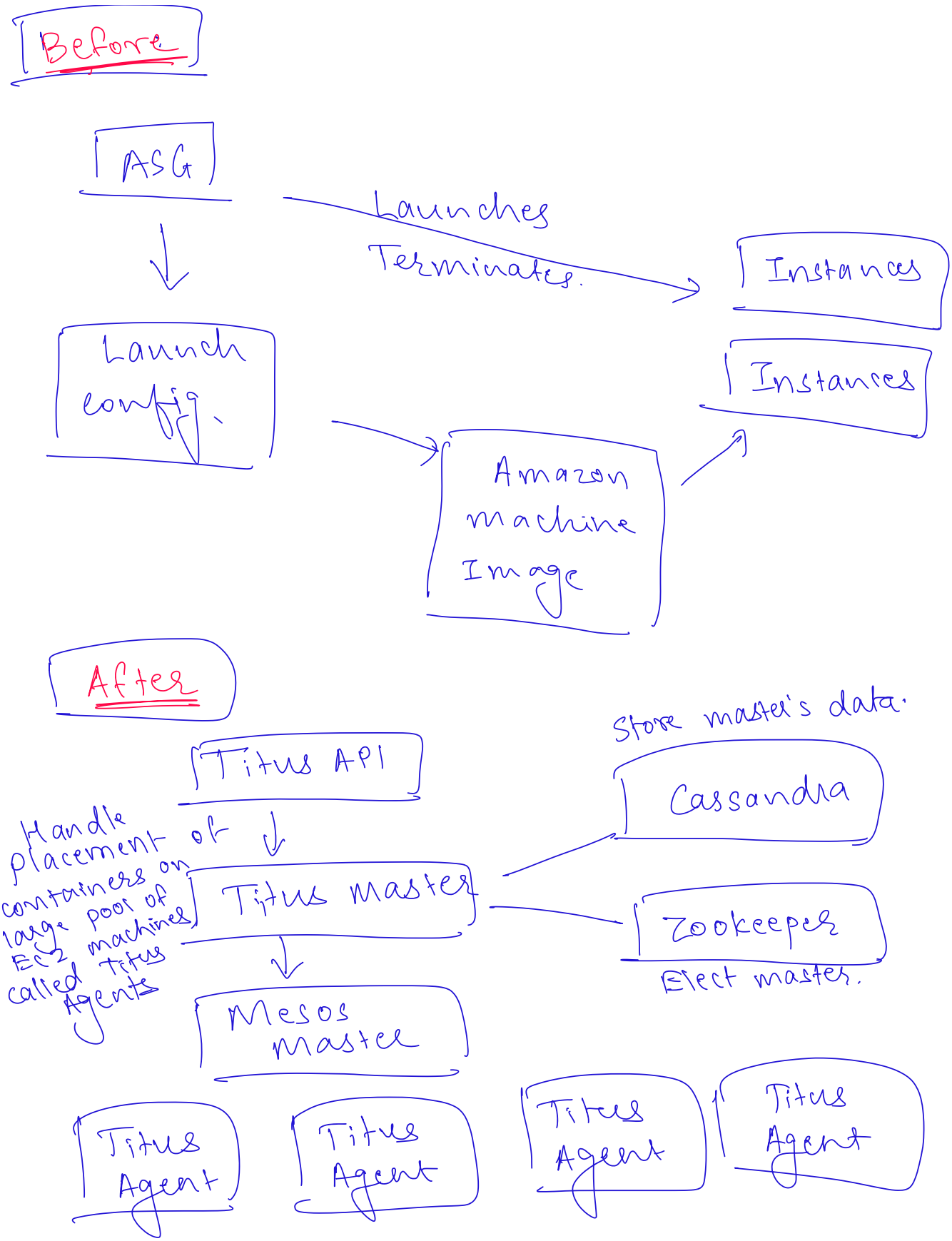


Faster networking : 100 Gb/s.

Faster storage : 4x faster.

Security : Hardware enforced.

Cheaper : Better utilization of resources.


Before
ASG
Launches
Terminates.
Instances
Instances
Launch config.
Amazon machine Image
After
Titus API
Store master's data.
Cassandra
Handle placement of containers on large pool of EC2 machines called Titus Agents
Titus master
Zookeeper
Elect master.
Mesos master
Titus Agent
Titus Agent
Titus Agent
Titus Agent