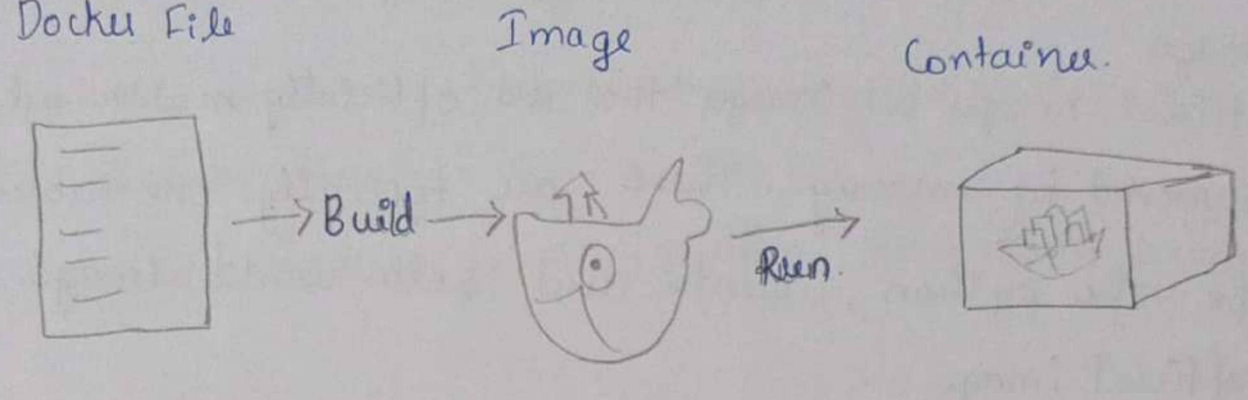
9) Docker terminology:



→ Docker Image: Once the code is ready with the application. and the DockerFile is written (it is a text file with docker commands that are necessary to create an image and install required dependencies). When we build from Docker File we get Docker Image it contains all the dockerfile, libraries and code your application needs to run, all bundled together. To get a new Docker image, youcan either get it from a registry or create your own. There are ten-thousands of images available on Dockerhub. We can also search for images directly from the command line using the docker search.

→ Docker Images are of two types :- i) Parent Image
i) Base Image
ii) Child Image

→ Base images are images that have no parent image, usually images with an OS like Ubuntu, Python, or Debian.

→ Child images are images built on base images and add additional functionalities.

→ There are official and user images which can be base or child images

→ Official images are images that are officially maintained and supported by company. These are typically one word-long Ex: the python, ubuntu and hello-world images are official images

→ User images are created and shared by users. They build on base images and add additional functionality.

→ Once image is created the code is ready to be launched.

* Docker Container: It is a Docker Image brought to life.

Container has:-

i) A Docker Image

ii) An execution environment

iii) A standard set of instructions

Executing a docker image with a dedicated command will create and start container from that image.

• Docker hub: A registry of Docker images. One can host their own registries and use them for pulling images

Step-by-Step Execution:

1. Download Docker Community Edition for free.
2. Create Docker Image by writing and building a Dockerfile.

Command to build image from DockerFile:

```
docker build --tag [image-name]
```

3. Run the container from Docker Image. So that it is up and running

Command: `docker run --publish 8000:8000 imagename`

4> Commit

```
docker commit [container-name] userid/imagename:latest
```

5> We can push to registry.

```
docker push userid/imagename:latest.
```

6> We can pull from registry and verify.

```
docker pull userid/image-name:latest
```

7> Run the image that is pulled.

```
docker run -td -p 8000:8000 --name containername user-id/image-name
```

6c) Novel ideas for a trusted virtual machine monitor.

→ It should support not only traditional operating systems, by exporting the hardware abstraction for open-box platforms, but also the abstractions for closed-box platforms

→ An application should be allowed to build its software stack based on needs

→ Applications requiring a very high-level of security should run under a very thin OS supporting only the functionality required by the application and the ability to boot.

→ At the other end of the spectrum are applications demanding low assurance, but rich set of OS features; such applications need a commodity operating system

→ Provide trusted paths from a user to an application. Such a path allows a human user to determine with certainity the identity of the VM it is interacting with and allows the VM to verify the identity of the human user

→ Deny the platform adminstrator the root access.

→ Support attestation, the ability of an application running in a closed-box to gain trust from a remote party, by crpytographically identifying itself

6b) The design principles:

Scheduling is a critical component of cloud resource management. It is responsible for resource sharing or multiplexing at several levels.

i) It should be efficient

ii) Fair - No priorities all are equal.

iii) Starvation free - Single process can't hold the resources for long and cause other processes or users to starve.

iv) Maximized throughput - No. of jobs completed per unit time.

v) Minimize turnaround time - the time between job submission and it's completion.

vi) For real-time systems - It should be predictable and meet demands.

Meet hard or soft deadlines.

→ It should take care of whom the resource is allocated, how much and at what time it is allocated

It is shared at

i) Server level

ii) VM level

iii) Application level

Types/Levels of scheduling algorithms and objectives? Pg No - 06

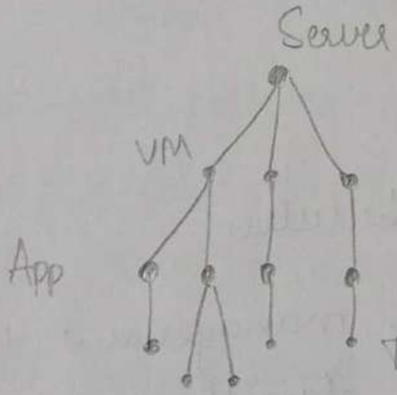
Scheduling algorithms should be able to decide for whom to assign the task, how much and how to meet deadlines.

Resources can be shared at different level.

i) Server level → It should be shared among VM's

ii) VM level → It should be shared among applications

iii) Application → It should be shared among threads



A scheduling algorithm should be fair, efficient, starvation free. - no process should wait indefinitely.

→ Objectives for a batch system:

i) maximize throughput

ii) minimize turnaround time.

iii) meet the deadlines.

iv) be predictable/solvable/tractable and stable

→ Objectives for real-time system.

i) to meet deadlines and to be predictable.

~~a~~ Schedulus for systems supporting mix of tasks are subjed contradictony requirements. Types Of Requirements?

(i) Hard real time requirement

eg: Satellite launching and response should not exceed more than 2ms.

(ii) Soft-real time constrained.

eg: Developer needs to update something should be done within 3hrs.

(iii). No-time constraints.

eg: Backup and Archive

Pre-emptive and non-premptive schedulers.

* Two ditinct dimensions of resource management that should be addressed by scheduling algorithms policy:

i) The amount or quantity of resources allocated.

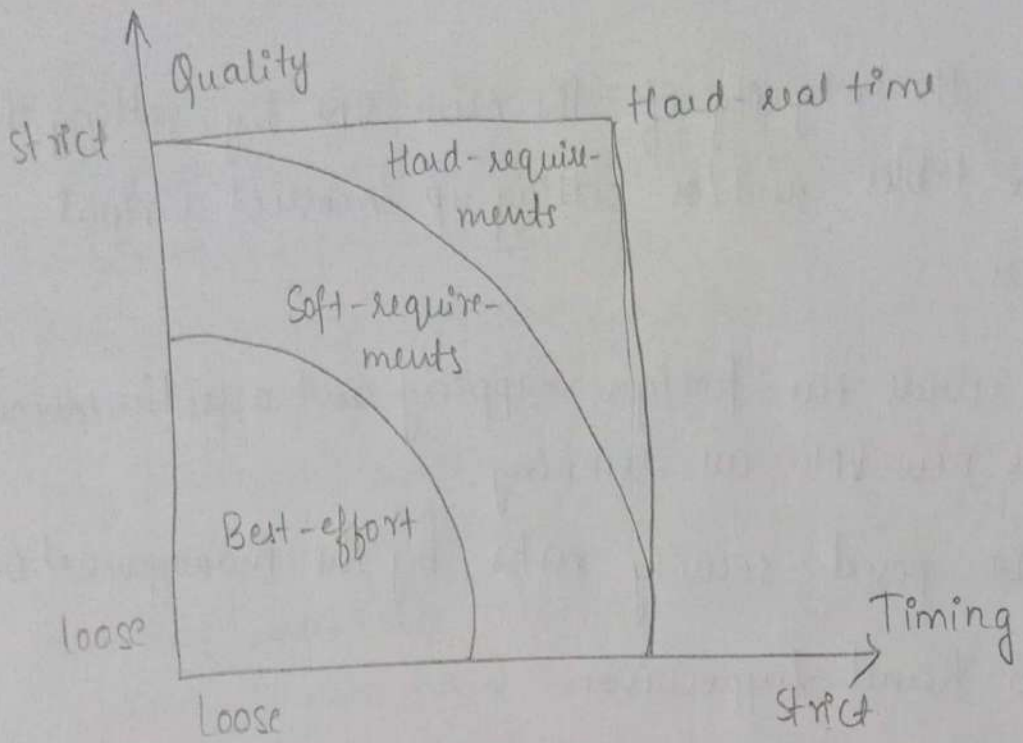
ii) The timing when access to resources are granted.

Classes of resource allocation requirements in the space defined by two dimensions

i) Best-effort

ii) Soft-requirement.

iii) Hard-requirement,



Classes of resource allocation requirements.

i) Best effort policies do not impose requirements regarding either the amount of resources allocated to an application or the timing when application is scheduled.

ii) Soft-requirements allocation policies require statiscally guaranted amounts and timing constraints.

iii) Hard-real time systems are the most challenging because they require strict timing and precise amounts of resources

The QoS requirements differ for different classes of cloud apps and demand scheduling policies. Best effort apps such as batch apps and analytics do not require QoS guarantees. Multimedia apps such as audio & video streaming have soft-real time constraints and require statistically guaranteed max delay & throughput. Apps with hard real-time constraint dont use a public cloud.

Management OS and security threats?

7b) To describe how management OS of micro kernel hypervisor are posing more security threat, compared to monolithic, we take the example of Xen architecture.

We often hear that virtualization enhances security because a hypervisor is considerably smaller than an OS. The hypervisor must rely on management OS to create VMs and to transfer data in and out from a guest VM to storage devices and network interfaces Xen includes not only the hardware and the hypervisor but also the management OS running in the so called Domain-0 (default VM). System vulnerabilities can be introduced by both software components. Xen and management OS. A malicious Domain 0 can play several nasty tricks at a time when it creates Dom-U. Some of the security attacks are:-

i) Refuse to carry out the steps necessary to the start the new VM. An action trat is also called as Denial-of-service attack.

ii) Modify the kernel of guest OS in ways that it will allow a third party to monitor and control the execution of applications running under the new VM.

iii) Undermine the integrity of the new VM by setting the wrong page tables and/or setting up incorrect virtual CPU registers.

iv) Refuse to release the foreign mapping and acess the memory while the new VMs are running.

Hence, the posed security risks by the management os of micro-kernel hypervisor.

Why cloud resource mgmt is important?

7a) It is most important function required for man-made system. It affects three criterias of evaluation for the system:
i) performance
ii) functionality
iii) cost.

Hence resource management is must for complex cloud system. If no resource management than the notion of "Unlimited resources" doesn't make any sense. Such complex system makes impossible to have accurate global state information of of the upredictable interactions with environments. There may be planned or unplanned demands and cloud must handle all those.

Cloud Resource Management Policies?

Policies typically refers to the principal guiding decisions, whereas mechanisms represent means to implement policies.

Policies are grouped into 5 classes:

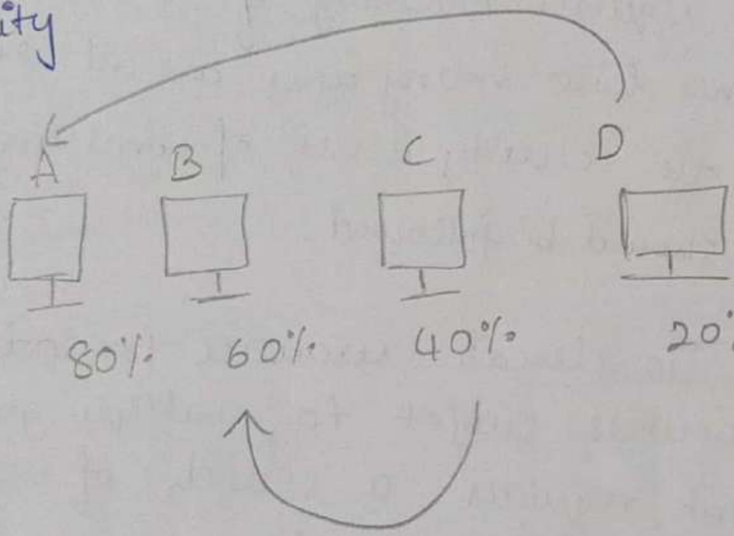
1> Admission control: To prevent system from accepting workloads in violation of high level system policies. A system may not accept any additional workload that would prevent it from completing work already in progress and contracted, limiting workloads requires knowledge of global state of system. It determines how many users are admitted at any instant, what are security issues of admitting new users, user policies should be followed.

2> Capacity allocation: To allocate resources for individual instance. Locating resources subject to multiple global optimization constraints requires a search of very large search space when state of individual system changes rapidly

3> Load Balancing and Energy Optimization:
They are interrelated. We need minimize cost as its commercial or market mode. So switching and maintainence energy should be low, low memory and carbon footprints that encourages green computing. Server energy utilization should be low

For example,
Consider 4 identical server A, B, C and D where relative loads are 80%, 60%, 40%. 20%. of their capacity

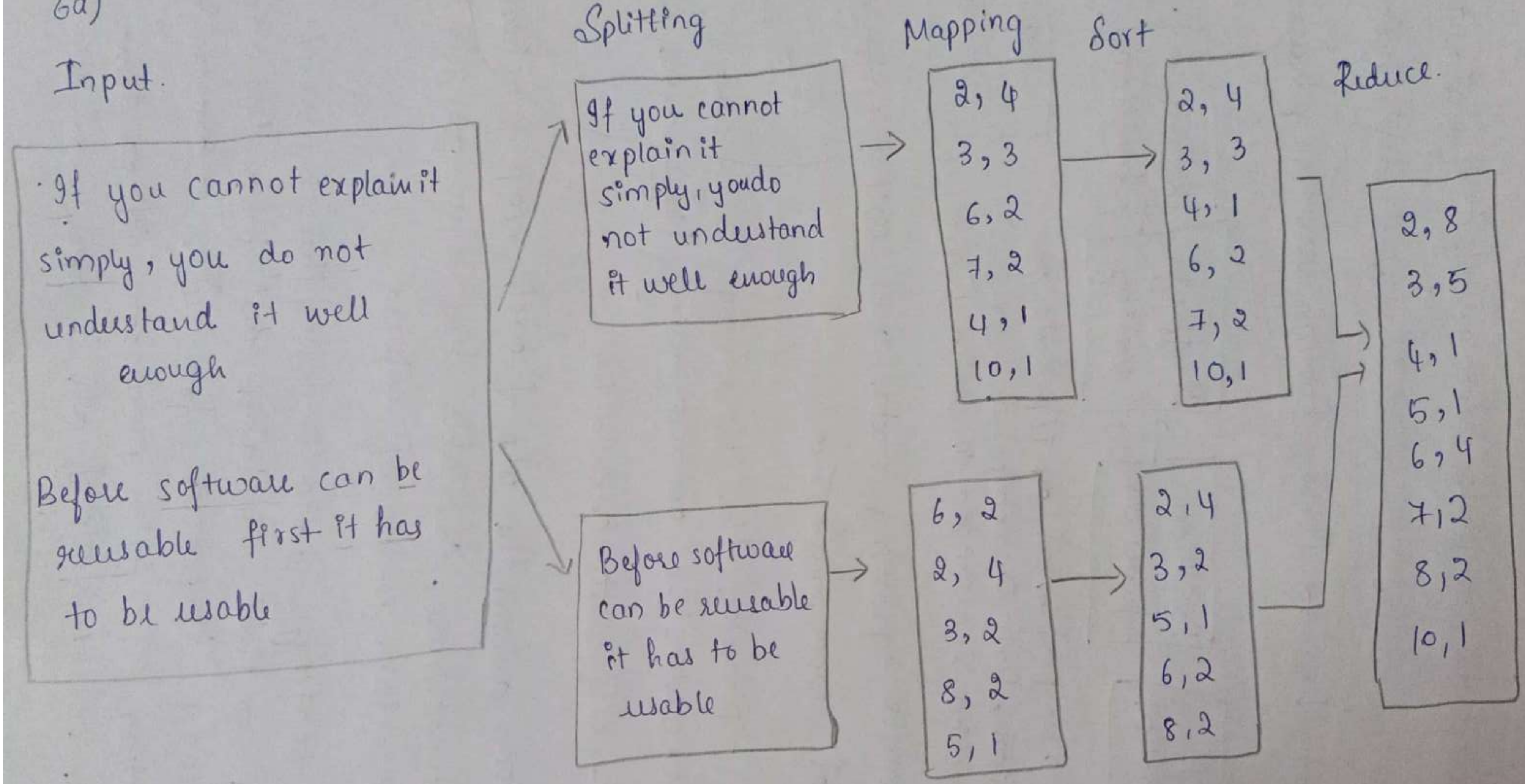
In cloud computing, a critical goal is to set a minimum cost of providing service and minimizing energy consumption. Instead of loads, we want to concentrate and use smaller no. of servers while switching others to standby mode to save energy. In our example from D are migrate to A and from C migrate to B so A and B are up and running to its maximum capacity



C and D will be shut down.

5) Quality of Service (QoS): Guarantee means the ability to satisfy timing or other conditions specified by it. CSP should abide by all the agreements made in Service-Level-Agreement (SLA).

6a)

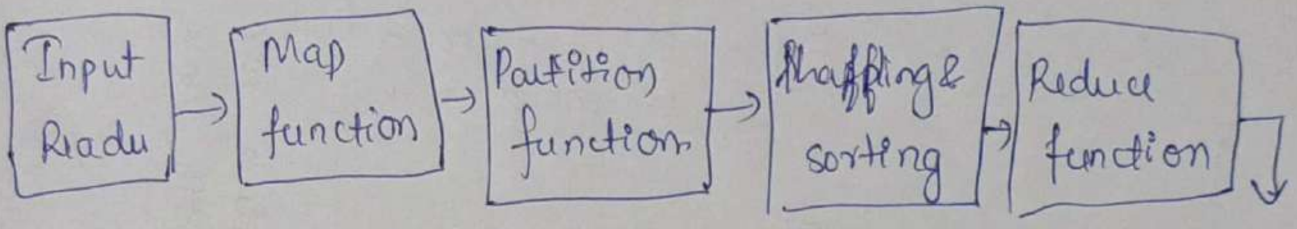

Input.
· If you cannot explain it simply, you do not understand it well enough
Before software can be reusable first it has to be usable
Splitting
If you cannot explain it simply, you do not understand it well enough
Before software can be reusable it has to be usable
Mapping
2, 4
3, 3
6, 2
7, 2
4, 1
10, 1
6, 2
2, 4
3, 2
8, 2
5, 1
Sort
2, 4
3, 3
4, 1
6, 2
7, 2
10, 1
2, 4
3, 2
5, 1
6, 2
8, 2
Reduce.
2, 8
3, 5
4, 1
5, 1
6, 4
7, 12
8, 2
10, 1

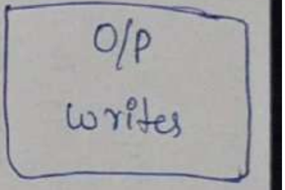
Map-Reduce for giving counts of words with same no. of letters?

Key → no. of letter
Value → count of words
Sort according to key.

7c)

Map reduce used to compute huge amount of data. To handle upcoming data in parallel and distributed form. the dat has to flow from various phases





* Sorting operation is performed on i/p data to reduce frunction. Data is compared using comparision function & arranged in sorted form

* Input reader reads upcoming data and splits into data block of appropriate size

* Map function - processes the upcoming key-value pairs and generates the corresponding key-value pairs, I/o may be different each other

* Partition function assigns the o/p of each map function to appropriate reducer. Available key value provides this function. It return index of reducer

* Shuffling and sorting: data is shuffled between/ within nodes so that it takes out from the nap and get ready to process for ~~redy~~ ready to process for reduce function. This takes much computation time.

* Reduce function combines o/p of all mappers and writes o/p finally