CS565 - CLOUD COMPUTING

Name : P.V. Sriram
Roll : 18015037

1) D - All
2) A - Distributed system, Message Passing
3) B - False
4) A - Utility Computing
5) D - Public
6) D - All
7) D - All
8) D - All
9) D - All
10) D - All of the above

11) Problems with Trap and Emulate process virtualization

(i) Guest OS may realize it is running at lower privilege level.
* Some registers in X86 reflect CPU privilege level
* Guest OS can read these values and get offended

(ii) Some X86 instructions which change hardware state run in both privileged and unprivileged modes
* Will behave differently when guest OS is in ring 0 vs in less privileged ring 1.
* OS behaves incorrectly in ring 1 will not trap to VMM
* Instruction set of X86 is not easily visualizable.

(iii) Eflags register is a set of CPU flags. Consider popf instruction in X86
* Executed in ring 0, all flags set normally
* Executed in ring 1, only some flags set

So, popf is a sensitive instruction, not privileged, doesn't trap behaves differently when executed in diff privilege levels, Guest OS is buggy in ring 1.
* V-rings, no root/non-root modes yet
* hypervisor in ring 0, guest OS in ring 1
* HV doesn't know, so it doesn't try to change setting
* OS doesn't know, so assumes change was successful.

Binary Translation

In binary translation, we rewrite VM binary to never issue those 17 instructions which causes problems in Trap-and-Emulate.

Goal: Full virtualization
= Guest OS
not modified

Approach: Dynamic binary translation

1. Inspect code blocks to be executed
2. If needed, translate to alternate instruction sequence
   Eg: To emulate desired behaviour, possibly even avoiding trap.
3. Otherwise, run at hardware speeds.
   Cache translated blocks to amortize translation cost.

12) There are three key models for device virtualization

a) Pass through model: VMM level drivers configures device access permissions
* VM provided with exclusive access to the device
* VM can directly access the device.
* VMM must have exact type of device as what VM expects.

b) Hypervisor - Direct model: VMM intercepts all device accesses
* Emulate device operation
  a) Translate to generic I/O operation.
  b) Transverse VMM-resident I/O state.
  c) Invoke VMM-resident driver

c) Split-Device Driver model: Device access control split between
* Front end driver in guest VM
* Back end driver in service VM.

13) Linux Containers

A linux containa is a set of 1 or more processes that are isolated from the rest of the system. Containers run in host system kernel, Seperated with policies and can use apps in host system.

* Linux containers are portable between OS variants supporting linux container and there is no overhead with hypervisor and guest OS Kernel.

Advantages of using Containerized Apps over VM-based

* Containers are independent hosts for applications that use a single, stripped-down version of OS to run whereas VMs use full version of an OS.
* Containers run a virtualized workload, processed by an application broken up into microservices, making them more lightweight and flexible than a VM.
* Containers can scale up and down an application quickly and easily.

* Containers are highly suitable for a microservices architecture in which applications are broken into small, self-sufficient components which can be deployed and scaled individually. Containers are an attractive option for deploying and scaling each of these microservices.
* Containers are easily controlled by API and thus are also ideal for automation and CI/CD pipelines

14)

Hotspot detection

Sandpiper implements a hotspot detection algorithm that determines when to migrate VMs.

The hotspot detection component employs a monitoring and profiling engine that gathers usage statistics on various virtual and physical servers and constructs profiles of resource usage. These profiles are used in conjunction with prediction techniques to detect hotspots in the system. When there is increased workload on a service, it should be simply migrated to a less loaded server. Manually initiated migration lacks the agility to respond to sudden workload changes. So, detecting hotspots is an important step to flexibly remap physical resources to virtual servers

Hotspot migration:

Upon detecting hotspots, Sandpiper's migration manager is invoked for hotspot mitigation. This hotspot mitigation algorithm resorts to a heuristic to determine which overloaded VMs to migrate and where such that migration overhead is minimized.

15) Given,

(i) Desired mean response time, $d = 1.5$ s

(ii) The 10 most recent service times (in seconds) are 1.1, 0.9, 1.2, 0.8, 1.3, 0.7, 1.4, 0.6, 1.2, 0.8

(iii) Inter-arrival time follows gamma distribution,

$$\Gamma(a, b) = \Gamma(2, 1)$$

(iv) Peak request arrival time = 1 request per second.

(λ_{peak})

To find the factor by which CPU capacity should be scaled, we need to find the request arrival rate (λ_{cap}) using the following formula.

$$\lambda_{cap} \geq \frac{1}{s + \frac{\sigma_a^2 + \sigma_b^2}{2(d-s)}}$$

where S is the mean service time.

σ_a^2 is variance of inter arrival time

σ_b^2 is variance of inter service time

Mean service time (S) = { mean of 10 recent service times

$$= \frac{1.1 + 0.9 + 1.2 + 0.8 + 1.3 + 0.7 + 1.4 + 0.6 + 1.2 + 0.8}{10}$$

$$= \boxed{1 \text{ sec}}$$

Variance of inter arrival time σ_a^2

$$= ab^2 = 2 \times 1^2 = 2$$

(∵ Variance of $I(\alpha, \beta) = \alpha\beta^2$)

Variance of service time σ_b^2

= Mean of squares of service time } = { square of mean of service time

$$= 1.068 - 1.0 = 0.068$$

$$\Rightarrow \boxed{\sigma_a^2 = 2} \quad \boxed{\sigma_b^2 = 0.068}$$

$$\boxed{S = 1} \quad \boxed{d = 1.5}$$

$\therefore \lambda_{cap}$ follows the inequality,

$$\lambda_{cap} \geq \frac{1}{1 + \frac{2+0.068}{2(1.5-1)}} = \frac{1}{3.068}$$

$$\boxed{\lambda_{cap} \geq \frac{1}{3.068}} \quad -(1)$$

Factor by which CPU capacity should be increased is $\frac{\lambda_{peak}}{\lambda_{cap}} = \frac{1}{\lambda_{cap}}$

As $\lambda_{cap} \geq \frac{1}{3.068}$, $\frac{1}{\lambda_{cap}} \leq 3.068$

So, the current CPU capacity of the server should be scaled by a max factor of 3.068 so as to service peak request arrival time of 1 request per sec.

16) Microservices architecture allows developers to create separate components of an application through building an application from a combination of small services.

Benefits of microservice based architecture

1) Isolation

Microservices are profitable due to this isolation and resilience. If one of the component fails, developers have the option to use another service and application will run independently

2) Scalability

It's easier to scale up or down following the requirement of specific application (element).

3) Productivity

Ability to easily understand when compared to monolith. Beneficial to expand your development.

4) Flexibility and Faster project development

Since services work independently, no need to change whole codebase. You can change one component and then deploy, Helps to choose right tool w.r.t task, without affecting communication.

5) Evolutionary

Perfect choice for developers who are unable to predict the kinds of devices the app is going to run on.

---

17) Factors that should be considered while choosing a right cloud service provider

(i) Regular Compliance → Your businesses processes running on cloud based system are compliant with the standards which your customers require.

(ii) Adaptable to the Infrastructure requirements → Provider should understand your infrastructure requirements and offer solutions corresponding to that without comprimising on business continuity.

(iii) Favourable service level agreements → Establishes relationship between a user and cloud service provides

(iv) Different types of cloud services → Selecting the service that is compatible to the business requirement.

(v) Security → Cloud service should ensure that your data is safe and can traverse across different platform resilient with AUTHENTICATION and ease of availability.

(vi) Wont refuse custom support → Customer service is required

(vii) Flexibility in pricing plans → Costs should also be considered

---

18) Black Box monitoring

Black box monitoring refers to the monitoring of services with a focus on areas such as disk space, CPU usage, memory usage.

Grey Box monitoring

Grey box monitoring implies performing analysis of the system at runtime, which brings new limitations to monitorability (The feasibility of solving the monitoring problem).

---

19) Two approaches to address the problem of networking VMs

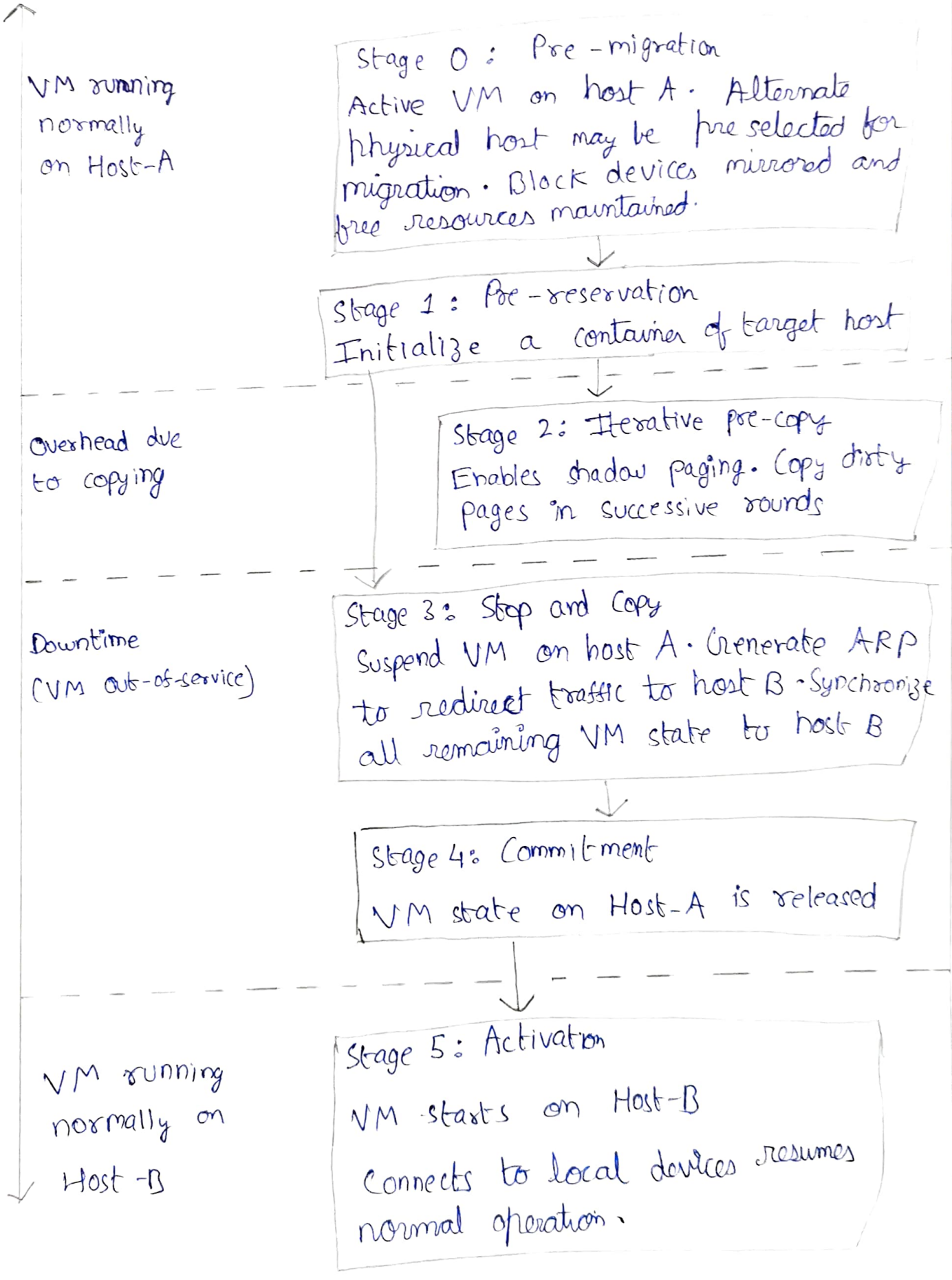
a) One using specialized Hardware

b) Software based approach

Hardware based	Software based
(i) SR-IOV, single root I/O virtualization	(i) Open V Switch
(ii) Sacrifices flexibility and forwarding logic	(ii) Focuses on forwarding flexibility

Hardware based	Software based
(iii) Hits native performance	(iii) The compromise made is to avoid targeting worst case performance.
(iv) The main idea behind is that CPUs are not designed to forward packets	(iv) Smarts of this approach is the switch routing decisions lie in user space
(v) SR-IOV, the physical link itself supports virtualization in hardware	(v) This is where one decides what rules or filters apply to packets of a certain type.
(vi) The NIC provides a physical function (a standard ethernet port). Several virtual functions are also provided	(vi) Perhaps based on network updates from other, possibly virtual, such as in network.
(vii) Each VM is mapped to one of these functions. So, the VMs themselves get NIC hardware resources	(vii) Programmed using open flow.

20) Stages of live VM migration



21) Given,

Start up that wishes to run a service with,

~~cloud service~~ 128 servers (1024 cores)

524 TB storage

Costs are given as follows:

cloud service provider cost = $0.12 / GB per month

= $0.10 / CPU per hour

owning cost = $ 349 K for storage

= $ ~~2205~~ 7.5 K for CPU per month

Now, we need to compare per month costs of these two options

Ownership (Considering operations for M months)

storage → $\frac{349}{M}$

Total → $\frac{1555}{M} + 7.5$

(Considering 0.45 : 0.4 : 0.15 split for hardware : power : network and 3 year lifetime of hardware)

Outsourcing:

$$\text{Storage} = \$0.12 \times 524 \times 1000 \sim \$62K$$

$$\text{Total} = \text{storage} + \text{CPUs} = \$62K + \$0.10 \times 1024 \times 24 \times 30$$
$$= \$136K$$

Breakeven analysis

Now, we shall analyse the required number of months for ownership operations to breakeven with outsourcing operations.

$$\frac{\$349K}{M} < \$62 \Rightarrow M > 5.55 \text{ months (For storage)}$$

$$\frac{\$1555K}{M} + 7.5 < \$136K \Rightarrow M > 12 \text{ months (For total)}$$

∴ I would prefer outsourcing since it is taking more than a year for breakeven point.

22)

Given service period = 30 days

Maximum service hours per day = 15 hours

Total service uptime = 30 days × 15 hours/day

= 450 hours

Cost per day = ₹2500 per day

Total cost = ₹2500 × 30 = ₹75000

Total outrage time = 5 + 0·5 + 1·5 + 5min + 2·5

= 9hrs 35 min

Service availability = $1 - \frac{\text{outrage time}}{\text{uptime (actual)}} = 1 - \frac{9\text{hr } 35\text{min}}{450 \text{ hrs}}$

= 97·92 %

So, the monthly uptime (availability) = 97·92% < 99%

Service credit available : 25% of total cost

= $\frac{1}{4}$ × 75000 ₹18750

∴ Effective cost payable : 75000 − 18750

= ₹ 56250

23)

a) Cost / effective hour

In-house server:

Purchase cost = ₹ 70000

Purchase cost (per hour) = $\frac{70000}{3\times365\times24}$ = ₹ 2.66

Efficiency = 40%

∴ Cost per effective hour = $\frac{2.65}{0.4}$ = 6.65

Power & cooling

∴ Total cost per effective hour = 6.65 + 20 + 30

Management cost

= ₹ 56.65

Cloud server:

Given cost / hr = ₹ 7

Efficiency = 80% ⇒ cost per eff hr = $\frac{7}{0.8}$ = 8.75

The cost per effective hour = 8.75 + 1

Management cost

= 9.75

∴ Total cost / effective hour

cloud = ₹ 9.75

In-house = ₹ 56.65

b)

Cost analysis for 3-years:

In-house server:

Purchase cost : ₹ 70000

Power and cooling cost for 3-years = $30 \times 3 \times 365 \times 24$

= ₹ 7,88,400

Mangement cost = $20 \times 3 \times 365 \times 24$ = ₹ 5,25,600

Total cost : 1384000

Cloud Server:

Cost per hour = ₹7

Cost for 3-years = $7 \times 3 \times 365 \times 24 = 183960$

Management cost = $1 \times 3 \times 365 \times 24 = 26280$

Total cost = $183960 + 26280 = 210240$

Given Revenue per day = 3000

For 3 years = $3000 \times 3 \times 365 = 3285000$

Expected profits =)

In-house = $3285000 - 1384000$ = 19,01,000

Cloud = $3285000 - 210240$ = 30,74,760

c) Let 'x' be modified efficiency

Equating cost/effective-hour for both scenarios

From (a), cost/effective-hour for cloud = 8.75

Cost/effective hour for inhouse = $\frac{70000}{3 \times 365 \times 24 \times x}$

$\Rightarrow \frac{7}{0.8} = \frac{70000}{3 \times 365 \times 24 \times x}$ $\Rightarrow x = \frac{2.66}{8.75} = 0.304$

$\Rightarrow$ x = 30.4%

∴ The required efficiency of inhouse = 30.4%

24)

a) Power consumption in a Physical machine is given by,

$$\Delta P_j = \begin{cases} P_{j,idle} + (P_{j,peak} - P_{j,idle}) \times UT_j & ; UT_j > 0 \\ 0 & ; \text{otherwise} \end{cases}$$

where,

$P_{j,idle}$ → Half of peak power consumption $P_{j,peak}$

UT_j → Percentage of utilization in PM_j

Server's total energy consumption ΔE_j is determined via the energy model represented as;

$$\Delta E_j = \int_t \Delta P(UT_j)\, dt$$

b) Proposed algorithm first distinguishes the PMs having hotspots, then selects some VMs from each hotspot PM to perform migration and determines destination PMs for the VMs. ~~To detect cotetspots~~

Algorithm to predict hotspots:

Input : PM_{active}
Output : Qs PM having hotspot or not
windowLength ← 20; $UT_{pm}(t) = (U_{pm}(t) / Cap_{pm}(t))$,
Set Th_{hot} applying MAD = 1 − 3 * MAD

for each pm ∈ PM_{active} do
begin
if Length · $UTHistory_{pm}(t)$ < windowLength then
if ($UT_{pm}(t) > Th_{static}$) then
return true
else
return False
else
predict $UT_{pm}(t+1)$ using CBS Bi-LSTM
if ($UT_{pm}(t) > Th_{hot}$) && ($UT_{pm}(t+1) > Th_{hot}$) then
return true
else
return False
end

Algorithm to predict cold spots :

We follow same algorithm as hotspot; but we compare UT_{pm} with Th_{cold}

ie) $UT_{pm} > Th_{hot}$ to $UT_{pm} \leq Th_{cold}$

c) PMs energy consumption relies on the utilization of resources mainly memory, network bandwidth, CPU, etc. However, existing research reveal that the CPU utilizes more power than other resources. Hence, we represent the PMs resource usage by its CPU usage. As compared to the energy consumption caused by different algorithms, the suggested approach uses a lesser amount of energy. The reason being that the PMs having their usage lower than the threshold limit are switched to power saving mode. The energy consumption of PMs is further optimized via packing the VMs into most loaded PMs.