11/3/25

1a) A study on Cloud Analyst tools with its steps to install and run cloud analyst

1b) Write a procedure to launch virtual machine using open stack

1(a)

Cloud Analyst

It is a cloud simulator tool developed at the University of Melbourne, built on top of the CloudSim. It provides a graphical user interface to facilitate setup and visualization. This tools helps developers and organizations optimize the deployment of applications across cloud infrastructures by analyzing factors such as request processing time, resource allocation etc..

Tools used in Cloud Analyst

1. CloudSim Framework:
   - A foundational simulation toolkit for modeling cloud computing systems
   - Provides functionalities for simulating data centers, virtual machines (VMs) and cloud resources.

2. Java programming language:
   - The tool is developed using Java, making its compatible with across-platform environments.
   - Supports modular development & customization

3. SimJava

   A discrete event simulation librations used for creating models within cloud analyst.

4. GUI

   built using Java swing, enabling users to configure simulations & visualize results easily

Advantages of Cloud Analyst Simulator

1. Cost Effectiveness: Reduced cost and efficient resource allocation
2. Flexibility and customization: Configurable simulations & Experiment looping
3. Enhanced Decision Making: performance metrics & geographic distribution analysis
4. Educational & Research Benefits: Easy to use Interface & Rapid prototyping and Testing
5. Scalability and Reliability:
   Simulation of Large-scale environments & dynamic simulation elements

5(b) Procedure to launch a virtual machine using openstack

1. Login to openstack Dashboard
   - login with credentials

2. Select Project
   - select appropriate from dropdown menu

3. Check Compute Resource Quota
   - compute > overview [ensure project has sufficient resources] to launch a VM

4. Launch instance
   - click instances & click launch instance button

5. Configure Instance Details
   - instance name, description, availability zone, count

6. Select boot source
   - choose image as boot source

7. Select ~~[illegible]~~ resource needs
   - CPU, RAM

8. Select Network
   - choose n/w for your VM

9. Additional settings:
   - Key Pair: select the uploaded SSH key pair for secure access
   - Click launch instance

[illegible] 11/3.

Lab-2 25/5/23

In the given cloud simulator to host a simple web application ~~Cloud~~ with below given configuration

a) 6 userbases and one data centre with 50 virtual machines with

each unit 1024 mb memory and processor speed as 100 mips.

(b) 4 userbases and 1 datacentre with 25 virtual machines each with 1024mb of memory and processor speed as 100 mips

Analyse the average minimum & maximum response time & data centre & processing time & write the findings.

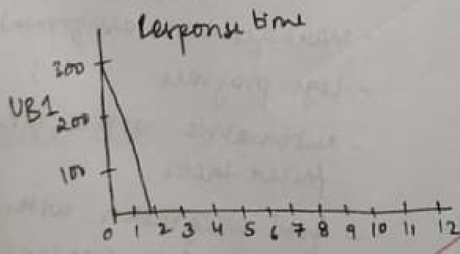
2(b) case study on workflow of amazon webservice

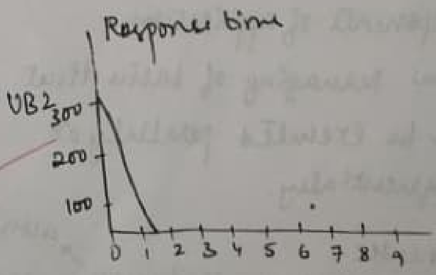
OUTPUT

Response time by region

Userbase	Avg (ms)	Min (ms)	Max (ms
UB1	301.173	225.637	365.204
UB2	299.667	231.637	362.151
UB3	299.668	233.141	366.269
UB4	300.974	234.641	362.151
UB5	300.984	225.641	363.651
UB6	300.533	233.137	363639

User Base Hourly Average Response Times

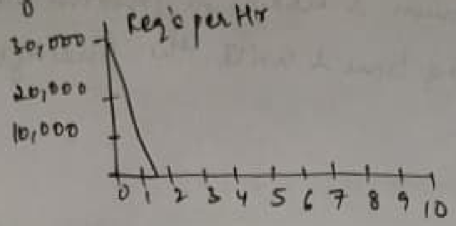




Data center Request servicing Times

Data center	Avg (ms)	Min (ms)	Max (ms)
DC1	0.847	0.07	1.189

Data centre Loading



Cost

Data centre	VM Cost	Data Transfer Cost	Total
DC1	5.018	0.385	5.403

(2b) AWS offers a robust framework framework for managing workflows through its various services notably the Amazon Simple workflow Service (SWF) and AWS step Functions. These services help developers to build, run, and orchestrate complex applications in a scalable manner.

Simple workflow service

Purpose: coordinate distributed components of application. Allow managing of tasks that can be executed parallely or sequentially.

Components:
- Tasks: unit of work – activity tasks, decision tasks
- Workers - execute the tasks (like EC2)
- Domains. containers for related workflows, tasks & actors

Functionality:
- tracks the execution of workflows
- manages task assignments
- logs progress
- automatic retries for failed tasks
- good integration with other AWS services

AWS Step Functions

Purpose: higher level abstraction for building workflows
- state machines

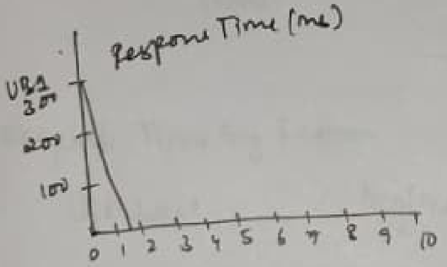
Features
- Visual Workflow Design (GUI)
- Built In Error handling

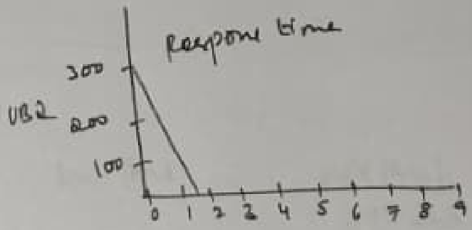
For an order Fulfilment
→ step 1: Validate Payment
step function triggers the lamda function to validate the payment
→ step 2: Check Inventory
step function triggers another lamda function to check the DynamoDB if there is enough stock
→ step 3: shipping order
It triggers the SQS to make sure this order is put into queue & fulfilled
→ step 4: send Notification
Triggers the amazon SNS to send the notification

1(b) Response Time By Region

Userbase	Avg (ms)	Min (ms)	Max (ms)
UB1	300.264	232.837	366.338
UB2	301.318	232.815	375.335
UB3	300.686	237.311	360.319
UB4	300.08	241.833	363.337

User Base Hourly Average Response Time

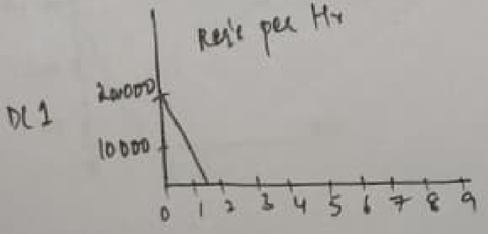




Data center Request servicing Times

Data center	Avg (ms)	Min (ms)	Max (ms)
DC1	0.549	0.037	0.856

Data Center Loading



Cost

Data center	VM Cost	Data Transfer Cost	Total
DC1	2.509	0.255	2.764

✓ 25/3

Lab program 4a) 1/4/25

In given cloud simulataor to host simple web application on cloud with the configuration given below

1) Two datacenters 50 virtual machine each with 1024mb memory and processor speed as 100 mps, set the no. of userbase to 6. Conduct the experiment by changing different load balancing policies and conclude which load balancing policy is better by clearly stating the reason.

4b) A case study on cloud computing networking challenges & oppurtunities for innovation.

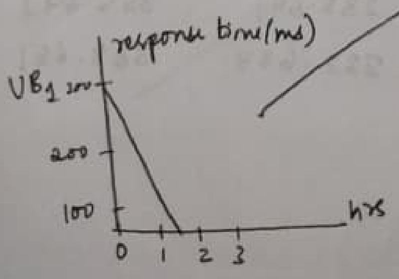
4a)sol. Round Robin

Overall Response Time Summary

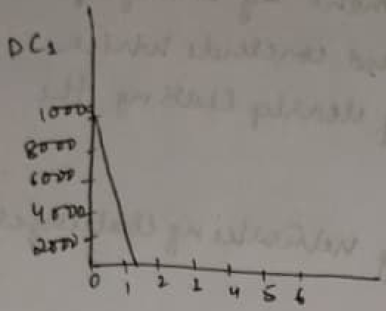
	Average (ms)	Minimum (ms)	Maximum (ms)
Overall Response Time:	300.91	227.14	384.64
Data center Processing Time:	1.16	0.07	1.83

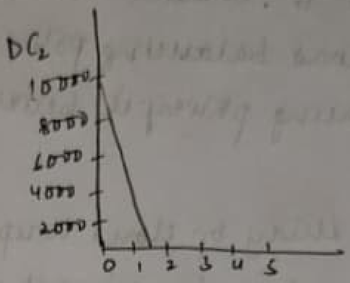
Response Time By Region

Userbase	Avg(ms)	Min (ms)	Max (ms)
UB1	300.33	227.141	379.265
UB2	300.854	242.137	368.763
UB3	300.884	239.76	374.761
UB4	302.37	232.266	365.768
UB5	300.842	237.636	370.26
UB6	300.173	231.643	384.64



Data center	Avg(ms)	min(ms)	max(ms)
DC₁	0.852	0.07	1.186
DC₂	1.438	0.132	1.814





Cost

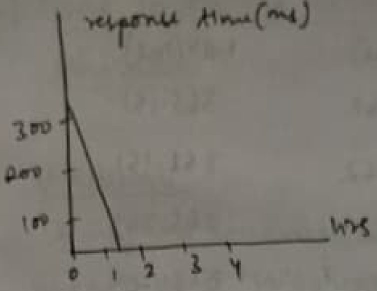
Total Virtual machine cost : $10.04
Total Data Transfer cost : $0.38
Grand Total : $10.42

Data center	VM cost	Data Transfer Cost	Total
DC₁	5.018	0.188	5.207
DC₂	5.018	0.196	5.215

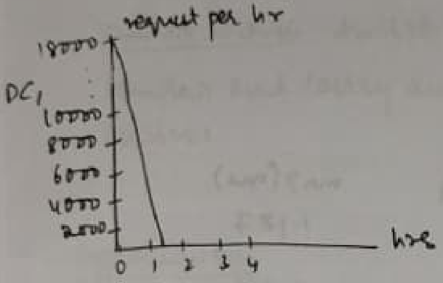
Equally spread current execution load

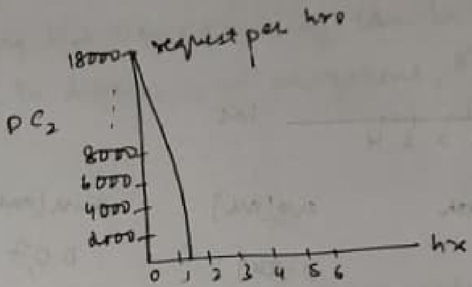
	Average	Minimum	Maximum
overall response time	300.26	22.614	316.27
Data center processing time	1.4	0.07	1.81

Userbase	Average(ms)	Min(ms)	max(ms)
UB₁	300.785	225.137	365.151
UB₂	299.271	231.617	362.151
UB₃	299.455	232.641	366.269
UB₄	301.01	234.141	365.204
UB₅	300.423	235.641	355.643
UB₆	300.631	232.637	363.651



Datacenter	Avg(ms)	Min(ms)	Max(ms)
DC1	0.845	0.07	1.187
DC2	0.449	0.133	1.814





Cost:

Total virtual machine cost : $10.04

Total Data Transfer cost : $0.38

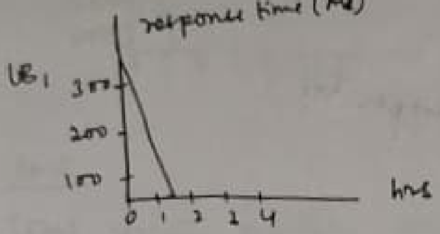
Grand Total : $10.42

Data center	VM Cost	Data Transfer Cost	Total
DC1	0.502	0.193	0.695
DC2	5.018	0.192	5.21

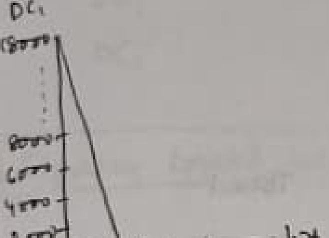
Throttled

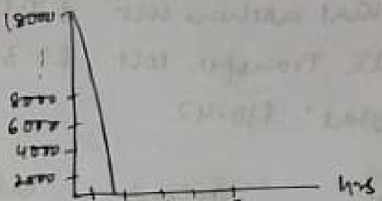
	Average	Min	Max
Overall response time	300.50	226.26	360.27
Data center processing time	1.15	0.07	1.81

Userbase	Avg(ms)	Min(ms)	max(ms)
UB1	301.344	226.262	365.151
UB2	299.984	232.262	361.151
UB3	300.46	233.766	366.269
UB4	301.809	235.266	365.204
UB5	300.741	226.266	363.651
UB6	300.771	233.132	364.264



Datacenter	avg(ms)	min(ms)	max(ms)
DC1	0.849	0.07	1.187
DC2	1.443	0.132	1.814





Cost

Total Virtual Machine cost: $10.04
Data Transfer cost: $0.38
Grand Total: $10.42

Data center	VMcost	Data transfer cost	total
DC1	5.018	0.193	5.212
DC2	5.018	0.191	5.21

Round Robin: Distributing incoming requests evenly across all servers in a circular manner regardless of server load.

Equally spread: Distributes tasks based on current load of each server ensuring fair share of workload.

Throttled: Limits no of requests sent to each server to prevent overloading typically by using predefined limits

4b) Cloud computing networking challenges and opportunities for innovation

Challenges

Latency and Bandwidth: high latency and limited bandwidth impact cloud performance, especially for realtime applications.

Security risks : Data breaches and unauthorised access are major concerns in multi cloud environments

Reliability and Downtime: Network issue can cause downtime

Scalability: sudden spikes can overwhelm cloud n/ws

Vendor lock in: switching b/w cloud providing can be complex and costly due to difference in platforms, APIs & services.

Opportunities

Edge computing: processes the data closer to the users reduces latency.

Global Accessibility: Data and services are accessible from any where with internet connectivity.

Rapid Deployment: Network services and resources can be provisioned quickly, enabling faster innovation & testing.

Enhanced cloud security features: cloud providers offer advanced security tools (firewall, encryption, identity management) as part of their services.

Cost Efficiency: Reduces the need for physical infrastructure lowering capital expenditure and maintenance costs.

Lab program 8/4/25

3a) In a given cloud simulator to host a simulator to host a simple web application on cloud with the configuration given below.

i) Two data centre with 50 virtual Machine each with 1024 mb memory and processor speed 100mbps, set the no. of user bases to 6. Conduct experimentation by changing different service broker policy to analyze average minimum & maximum response.
Also analyze the total cost required to run the application

3(b) A case study on the grep the web application

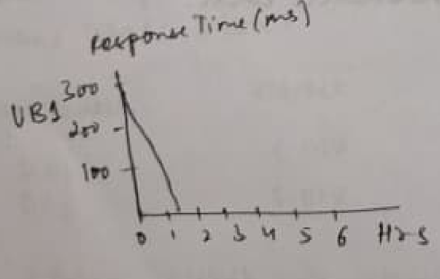
ans(3a) Closest Data Center

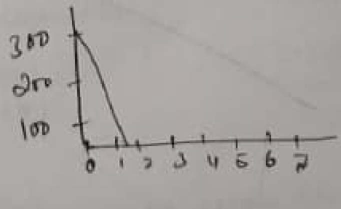
	Average (ms)	Minimum (ms)	Maximum (ms)
Overall Response Time :	300.79	225.64	366.27
Data center Processing Time :	1.14	0.07	1.79

Response Time By Region

Userbase	Avg(ms)	Min(ms)	Max(ms)
UB1	301.415	225.637	365.151
UB2	300.067	232.262	362.776
UB3	299.902	233.141	366.269
UB4	301.7	235.266	362.151
UB5	300.84	225.641	364.276
UB6	300.845	233.137	365.829

User Base Hourly Average Response Time

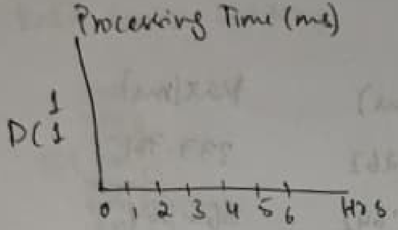


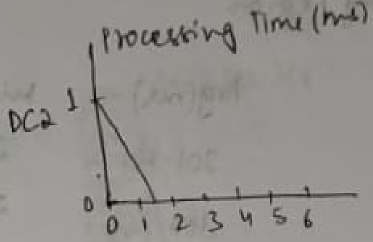


Data Center Request Servicing Times

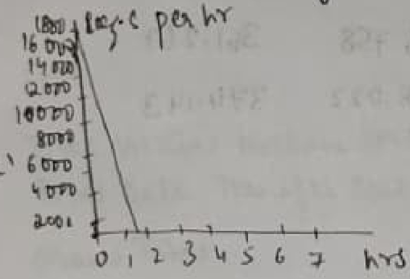
Data Center	Avg(ms)	Min(ms)	Max(ms)
DC1	0.848	0.07	1.187
DC2	1.442	0.132	1.792

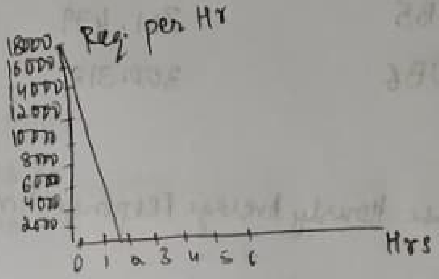
Data Center Hourly Average Processing Times





Data center Loading





Cost

Total Virtual Machine Cost: $10.04

Total Data Transfer Cost: $0.38

Grand Total: $10.42

Data Center	VM Cost	Data Transfer Cost	Total
DC2	5.018	0.189	5.207
DC1	5.018	0.196	5.215

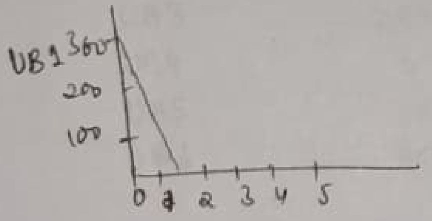
Optimise Response Time

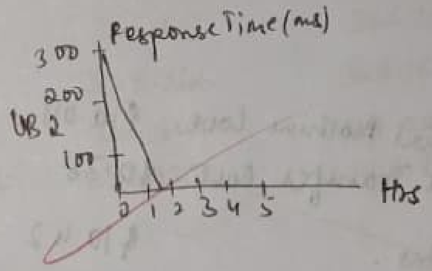
	Average (ms)	Minimum(ms)	Maximum(ms)
Overall Response Time:	300.91	226.26	380.14
Data Center Processing Time	1.15	0.07	1.80

Response Time By Region

Userbase	Avg(ms)	Min(ms)	Max(ms)
UB1	301.581	241.262	377.761
UB2	300.366	243.642	365.764
UB3	300.742	245.76	380.141
UB4	300.955	226.263	363.638
UB5	301.479	236.758	361.267
UB6	300.318	238.022	374.143

User Base Hourly Average Response Times

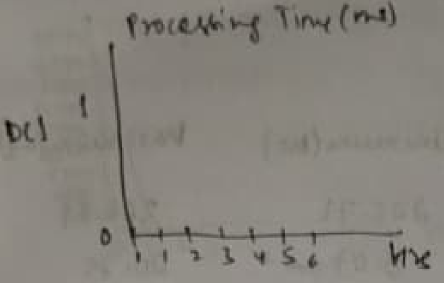


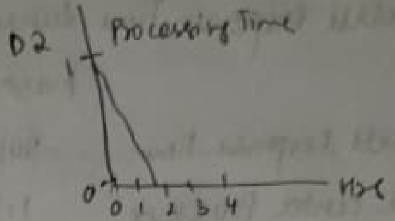


Data Center Request Servicing Times

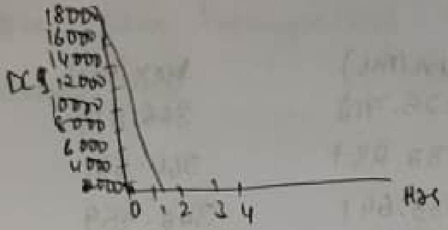
Data Center	Avg(ms)	Min(ms)	Max(ms)
DC1	0.862	0.07	1.191
DC2	01.452	0.132	1.803

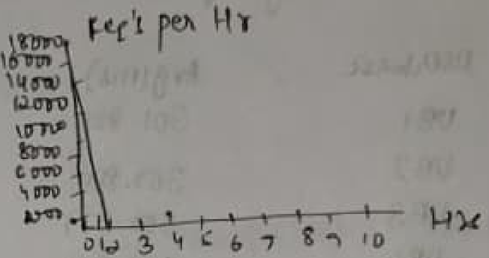
Data Center Hourly Average Processing Times





Data Center Loading





Cost

Total Virtual Machine Cost : $10.04
Total Data Transfer Cost : $0.38
Grand Total $10.42

Data Center	VM Cost	Data Transfer Cost	Total
DC2	5.018	0.189	5.207
DC1	5.018	0.196	5.214

P.T.O

Reconfigure Dynamically with L

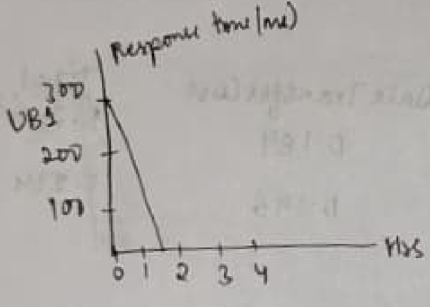
Overall Response Time Summary

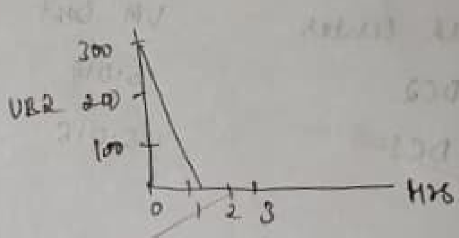
	Average (ms)	Minimum (ms)	Maximum (ms)
Overall Response Time:	301.48	225.71	366.81
Data Center Processing Time	1.82	0.07	20.25

Response Time By Region

Userbase	Avg(ms)	Min(ms)	Max(ms)
UB1	301.801	225.712	366.814
UB2	300.884	232.237	364.051
UB3	300.879	233.641	366.759
UB4	302.251	234.74	366.024
UB5	301.354	226.266	365.12
UB6	301.702	233.786	365.204

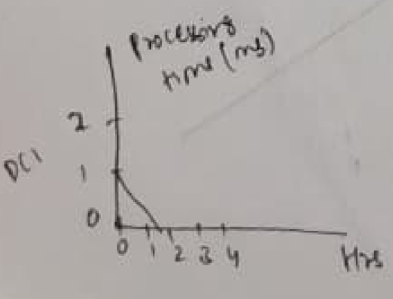
User Base Hourly Average Response Times

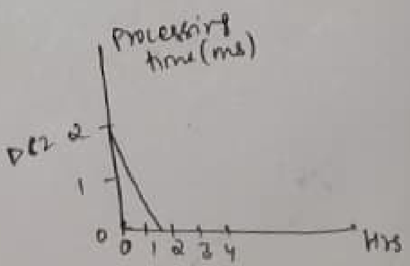




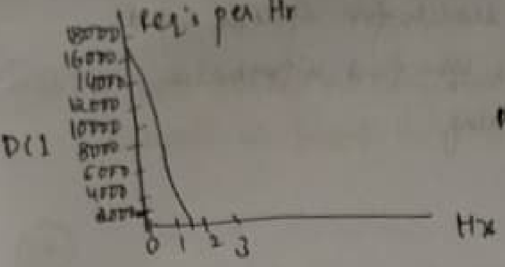
Data Center Request Servicing Times

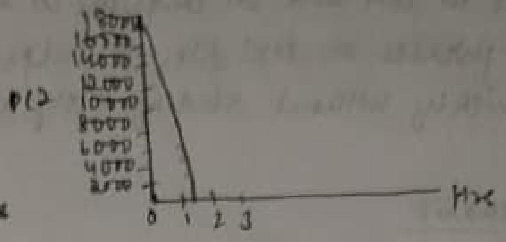
Data Center	Avg(ms)	Min(ms)	Max(ms)
DC1	1.235	0.072	3.577
DC2	2.411	0.135	20.251





Data center loading





Cost

Total Virtual Machine Cost:	$15.43
Total Data Transfer Cost:	$0.38
Grand Total:	$15.82

Data center	VM Cost	Data Transfer Cost	Total
DC2	7.72	0.193	7.913
DC1	7.715	0.192	7.907

Service Broker Policy	Description	Key Considerations
Closest Data Center	Routes user workload to the data center with the least n/w latency. If multiple datacenters have the same proximity it selects one randomly to balance the load	Proximity, N/w Latency
Optimize Response Time	selects the datacenter that offers the lowest response time, considering both n/w latency & datacenter processing time	Response Time, Processing Time
Reconfigure Dynamically	Dynamically adjusts the routing based on current conditions such as load & performance metrics, allowing for real time optimization	Dynamic Load Balancing, Real-time Optimization

3(b) Grep: Case study to find words in Text Files

It is tool used in computers to search for specific words or phrases in text file. It helps you find information quickly without reading everything.

Scenario

You have a folder with several text files about different books, and you want to find all the books that mention the word "magic".

Steps to use 'grep'

1. Open the Terminal
2. Go to Yo Folder (.txt)
3. Search for "magic"

```
grep "magic" *.txt
```

4. See the Results

Extra Tips

- Ignore Case: MAGIC or maGic

```
grep -i "magic" *.txt
```

- Count occurences

```
grep -c "magic" *.txt
```

[illegible] 8/4/25

5(a) Find a procedure to transfer the files from one virtual machine to another virtual machine.

(b) A case study on Performance Evaluation of Distributed File Systems in Cloud Environments.

5a

Ans Open virtual box with specific configurations

1) General → Advanced

   Shared Clipboard: Bidirectional
   Drag 'n' Drop: Bidirectional

2) Network → Attached to: Bridged Adapter

In the terminal

```
1) sudo apt install openssh-server
2) sudo apt install net-tools
3) ifconfig
```

COPY IP configuration of the guest OS

Open WinSCP

Host name: 172.25.6.249

User name: vm

password: 12345

Login ✓

Now the interface allows the transfer ~~the~~ of files from one virtual machine to another.

5b Performance Metrics for Evaluation

1) Throughput: This measures the amount of data that can be processed in a given time frame, typically expressed in megabytes per second. High throughput is essential for applications that require large data transfers.

2) Latency: This is the time it takes for a request to travel from the client to the server & back. Low latency is crucial for user experience, especially in interactive applications.

3) Scalability: This refers to the system's ability to handle increased loads by adding more resources without significant performance degradation.

4) Availability: This metric indicates the system's uptime & reliability. A highly available system ensures that users can access their files whenever needed.

5) Consistency: This measures how up-to-date the data is across different servers. In a distributed system, ensuring that all users see the same data at the same time can be challenging.

Challenges in performance Evaluation

Network Latency: The performance of a DFS is heavily dependent on a network speed & reliability. High latency can lead to delays in file access & retrieval

Data Locality: When data is stored across multiple servers, accessing files that are not located close to the client can result in increased latency. Optimizing data placement is crucial for performance.

Load Balancing: Uneven distribution of data & requests across servers can lead to some servers being overloaded while others are underutilized. Effective load balancing is essential for maintaining performance.

[illegible] 29/4.

6(a) Docker containers

a) Installing Docker Engine on EC2 instance
b) Basic Docker commands. Understanding docker images & building images using Dockerfile. Exposing ports for web microservices, creating and using docker networks. Data persistence using docker volumes.

(a)

1) Installing Dock

2) Docker engine on AWS EC2 instance

3) Docker & images & Dockerfiles
   a. Screenshot of C Program successfully run inside container

4) Exposing ports, docker networks

   a. Sample.html showing the web page on the browser. Screenshot should show public dns of the EC2 host of docker

   b. Screenshot of docker container running nginx (terminal of EC2 host)

   c. Screenshot of python application successfully writing and reading from the MongoDB ~~[illegible]~~ database.

   d. Screenshot showing mongodb being run within network (docker command has to be clearly highlighted)

   e. Screenshot showing python file being run within network and successfuly writing and reading from MongoDB (docker command has to be clearly highlighted)

5) Docker Compose

   a. Screenshot of python-mongodb application running as a docker compose application (logs of the application)

   b. Screenshot of 3 python application writes and reads ~~tot~~ from MongoDB after scaling the python application.

6. Docker Volumes

   a. Screenshot clearly showing command run to mount the host volume and [illegible] manually running the python application inside the container.

Docker Basic Commands

Task 1: Installing Docker Engine on EC2

1. Create EC2 instance
2. Get keypair file and SSH into the instance
3. Install docker engine on the instance
   a. Install Docker Engine on Ubuntu
4. ~~[illegible]~~ make docker "sudo less" command
5. Verify docker engine using
   docker run hello-world

Task 2: Docker images and docker files

1. Explore various images
2. The images in docker hub, need to be pulled into our EC2 instance in order to use them.
   a. docker pull
3. To find ~~[illegible]~~ images that exist on your instance run
   a. docker images
4. Run the container using docker run
5. To list currently running container use
   - docker ps
6. To stop running container use docker stop
7. To remove container run docker rm

8. After you have your Dockerfile, build the image using docker build & run the container

Task 3 : Exposing ports, docker networks

1. Download sample HTML file, modify CRN
2. Create Dockerfile and then docker image having nginx: latest
   - Build docker image & tag/name it your CRN
3. Run the container. Expose the HTTP port & check connectivity

~~Dock~~

Connectivity without docker networks & how docker networks make it much easier to connect within containers.

1. Run mongodb container
   a. Use mongo: latest image
   b. Name the container as mongodb
   c. Docker detached mode
2. Download sample.py. Modify SRN
3. Find IP using docker inspect
4. Create a Dockerfile ~~[illegible]~~

Docker Networking Options

1. Create docker bridge network called my-bridge-network
2. Run mongodb container again
   a. network: my-bridge-network
   b. name: mongodb
   c. Exposed ports: 27017
   d. Image: mongo: latest

Task 4: Docker compose

1. Following files in directory
   a. docker-compose.yml
   b. Dockerfile
   c. sample.py

2. Install docker compose

3. Understand syntax

4. run: docker-compose up

5. scale the python application
   a. docker-compose scale

Task 5: Docker Volumes

Volumes are the preferred mechanism for persisting data generated by and used by Docker containers.

1. Create a new directory on the EC2 host, with your SRN
2. Add the folders inside it
   a. sample.py
   b. details.txt containing name, srn, section, sem
3. Using Ubuntu: 18.04 as base image

4. Mount the host volume you have created & create bash shell into the container

5. Inside container, run the python program to display contents of the list