



VIT®

# **Vellore Institute of Technology**

(Deemed to be University under section 3 of UGC Act, 1956)

# DEADLOCK AVOIDANCE THROUGH EFFICIENT LOAD BALANCING IN CLOUD ENVIRONMENT

By

Jananya Sivakumar (19BCE1602)

Tejas V (19BCE1607)

Sandhya S (19BCE1614)

Shravan Ganapathiraman (19BCE1703)

M.A. RAM(19BCE1162)

Submitted to

Prof. Subbulakshmi P

## Overview:

- Using cloud computing implementation of detection and avoidance algorithm of deadlock to prevent cloud disaster in cloud computing since many users use the same platform concurrently so there is a huge demand for resources which sometimes cause deadlock conditions.
- Deadlock should be minimized to get better multiprocessing efficiency. The aim of this paper is therefore to analyze different simulation tools for various disaster recovery in Cloud computing.
- This examination empowers one to appreciate the selection of instruments for their applications. This learning further gives data about parameters that are bolstered by the devices and thus opens up research roads towards those parameters which are not yet esteemed in the devices.
- The condition of- specialty of the innovation centers around information handling to manage enormous measures of information.
- Distributed computing is a rising innovation, which empathizes one to achieve the previously mentioned goal, driving towards improved business execution

## OBJECTIVES:

- The objective of this review paper is to summarize the powerful data backup recovery techniques that are used in the cloud computing domain.
- Further, this paper likewise furnishes the foreseen outcomes with the execution of the proposed calculation.
- The halt evasion upgrades the quantity of occupations to be adjusted by cloud specialist co-op and along these lines improving working execution and the matter of the cloud specialist organization.

## **DESCRIPTION:**

- Deadlock is defined as the permanent blocking of a set of processes that compete for system resources, including database records or communication lines.
- Unlike other problems in multiprogramming systems, there is no efficient solution to the deadlock problem in the general case.
- The main applicability of deadlock avoidance is that it is not necessary to preempt and roll back mechanisms, as in deadlock mechanisms.
- Also one of the other applications of deadlock avoidance is decreased restriction than deadlock prevention which makes it easier to implement it.
- The approach to application of deadlock avoidance is to manipulate to find at least one safe path but knowing that future resource requirements must be known and processes might be waiting for a long time.
- One of the applications of deadlock avoidance is to allocate all required resources to the process before the start of its execution, this way hold and wait conditions are eliminated but it will lead to low device utilization.

## **MOTIVATION:**

- The performance of this model is influenced by various factors like Scalability, Reliability, Security, Disaster Management .
- The advancement in the Cloud computing model has led to an increase in demand for its services.
- An efficient and effective performance of this model is essential to have better business returns.
- The performance of this model is influenced by various factors like Scalability, Reliability, Security, Disaster Management etc.
- Disaster is one of the factors which deteriorate Cloud performance.

- Though it is an unpredictable event it can be prevented if the system is designed with efficient Scheduling, Resource allocation, Load Balancing etc. Disaster management is a challenging task, it provides a platform for a Cloud Service provider to exhibit the best service.

## Implementation details:

### How to Make Amazon EC2 Instance

Welcome to the new instances experience!

We're redesigning the EC2 console to make it easier to use. To switch between the old console and the new console, use the New EC2 Experience toggle above the navigation panel. We'll release updates continuously based on customer feedback.

**Instances (1) Info**

Filter instances Clear filters

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Publ...
i-0ed53486d7b2d498b	i-0ed53486d7b2d498b	Running	t2.micro	2/2 checks passed	No alarms	us-east-2b	ec2-...

Select an instance above

Log in to AWS and go to services where we have to go to launch instances

Step 1: Choose an Amazon Machine Image (AMI)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.

Search for an AMI by entering a search term e.g. "Windows"

Cancel and Exit

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

Quick Start

My AMIs

Amazon Linux Free tier eligible

Amazon Linux 2 AMI (HVM), SSD Volume Type - ami-077e31c4939f6a2f3 (64-bit x86) / ami-07a3e3eda401fbcaa (64-bit Arm)

Amazon Linux 2 comes with five years support. It provides Linux kernel 4.14 tuned for optimal performance on Amazon EC2, systemd 219, GCC 7.3, Glibc 2.26, Binutils 2.29.1, and the latest software packages through extras. This AMI is the successor of the Amazon Linux AMI that is approaching end of life on December 31, 2020 and has been removed from this wizard.

Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

macOS Big Sur 11.3.1 - ami-0aab315fd6492be58

The macOS Big Sur AMI is an EBS-backed, AWS-supported image. This AMI includes the AWS Command Line Interface, Command Line Tools for Xcode, Amazon SSM Agent, and Homebrew. The AWS Homebrew Tap includes the latest versions of multiple AWS packages included in the AMI.

Root device type: ebs Virtualization type: hvm ENA Enabled: Yes

macOS Catalina 10.15.7 - ami-050efa637ab09b285

The macOS Catalina AMI is an EBS-backed, AWS-supported image. This AMI includes the AWS Command Line Interface, Command Line Tools for Xcode, Amazon SSM Agent, and Homebrew. The AWS Homebrew Tap includes the latest versions of multiple AWS packages included in the AMI.

Select

64-bit (x86)

64-bit (Arm)

Select

64-bit (Mac)

Select

64-bit (Mac)

Choose amazon linux 2 AMI

**Step 2: Choose an Instance Type**

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. [Learn more](#) about instance types and how they can meet your computing needs.

Filter by: All instance families ▾ Current generation ▾ Show/Hide Columns

Currently selected: t2.micro (- ECUs, 1 vCPUs, 2.5 GHz, -, 1 GiB memory, EBS only)

Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance	IPv6 Support
t2	t2.nano	1	0.5	EBS only	-	Low to Moderate	Yes
t2	t2.micro Free tier eligible	1	1	EBS only	-	Low to Moderate	Yes
t2	t2.small	1	2	EBS only	-	Low to Moderate	Yes
t2	t2.medium	2	4	EBS only	-	Low to Moderate	Yes
t2	t2.large	2	8	EBS only	-	Low to Moderate	Yes
t2	t2.xlarge	4	16	EBS only	-	Moderate	Yes
t2	t2.2xlarge	8	32	EBS only	-	Moderate	Yes
t3	t3.nano	2	0.5	EBS only	Yes	Up to 5 Gigabit	Yes

Cancel Previous Review and Launch Next: Configure Instance Details

**Configure instance details**

**Step 4: Add Storage**

Your instance will be launched with the following storage device settings. You can attach additional EBS volumes and instance store volumes to your instance, or edit the settings of the root volume. You can also attach additional EBS volumes after launching an instance, but not instance store volumes. [Learn more](#) about storage options in Amazon EC2.

Volume Type	Device	Snapshot	Size (GiB)	Volume Type	IOPS	Throughput (MB/s)	Delete on Termination	Encryption
Root	/dev/xvda	snap-06937a7c52a33c946	8	General Purpose SSD (gp2)	100 / 3000	N/A	<input checked="" type="checkbox"/>	<input type="checkbox"/> Not Encrypted

Add New Volume

Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage. [Learn more](#) about free usage tier eligibility and usage restrictions.

Cancel Previous Review and Launch Next: Add Tags

Add storage of your instance according to your need

**Step 5: Add Tags**

A tag consists of a case-sensitive key-value pair. For example, you could define a tag with key = Name and value = Webserver.

A copy of a tag can be applied to volumes, instances or both.

Tags will be applied to all instances and volumes. [Learn more](#) about tagging your Amazon EC2 resources.

**Key** (128 characters maximum) **Value** (256 characters maximum)

	Instances	Volumes	Network Interfaces
Tejas	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

**Add another tag** (Up to 50 tags maximum)

**Cancel** **Previous** **Review and Launch** **Next: Configure Security Group**

Add tags to make your instance descriptive

**Step 6: Configure Security Group**

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. [Learn more](#) about Amazon EC2 security groups.

**Assign a security group:**  Create a new security group  
 Select an existing security group

**Security group name:** launch-wizard-1

**Description:** launch-wizard-1 created 2020-10-09T15:21:16.540+05:30

Type	Protocol	Port Range	Source	Description
SSH	TCP	22	Custom 0.0.0.0/0	e.g. SSH for Admin Desktop

**Add Rule**

**Warning**  
Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

**Cancel** **Previous** **Review and Launch**

Confirm security group to allow public or private access to your instance

**Step 7: Review Instance Launch**

Please review your instance launch details. You can go back to edit changes for each section. Click **Launch** to assign a key pair to your instance and complete the launch process.

**AMI Details**

**Amazon Linux 2 AMI (HVM), SSD Volume Type - ami-077e31c4939f6a2f3**

Free tier eligible

Amazon Linux 2 comes with five years support. It provides Linux kernel 4.14 tuned for optimal performance on Amazon EC2, systemd 219, GCC 7.3, Glibc 2.26, Binutils 2.29.1, and the latest software packages through extras. This AMI is the successor of the Amazon Linux AMI that is a...  
Root Device Type: ebs Virtualization type: hvm

**Instance Type**

Instance Type	ECUs	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance
t2.micro	-	1	1	EBS only	-	Low to Moderate

**Security Groups**

Security group name: launch-wizard-2

**Network Performance**

**Launch**

## Review of the instance

**Step 7: Review Instance Launch**

**AMI Details**

**Amazon Linux 2 AMI (HVM), SSD Volume Type - ami-077e31c4939f6a2f3**

Free tier eligible

Amazon Linux 2 comes with five years support. It provides Linux kernel 4.14 tuned for optimal performance on Amazon EC2, systemd 219, GCC 7.3, Glibc 2.26, Binutils 2.29.1, and the latest software packages through extras. This AMI is the successor of the Amazon Linux AMI that is a...  
Root Device Type: ebs Virtualization type: hvm

**Instance Type**

Instance Type	ECUs	vCPUs
t2.micro	-	1

**Security Groups**

Security group name: launch-wizard-2  
Description: launch-wizard-2 created

**Select an existing key pair or create a new key pair**

A key pair consists of a public key that AWS stores, and a private key file that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about [removing existing key pairs from a public AMI](#).

Create a new key pair  
Key pair name: Tejasji  
**Download Key Pair**

You have to download the **private key file (.pem file)** before you can continue. Store it in a secure and accessible location. You will not be able to download the file again after it's created.

**Network Performance**

**Launch Instances**

Create a key pair, download it and then convert it to ppk form using putty

The screenshot shows the AWS EC2 Management console. On the left, there's a sidebar with options like EC2 Dashboard, Events, Tags, Limits, Instances (selected), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Images, AMIs, and Elastic Block Store. The main area is titled 'Instances (2) Info' and shows two running instances. The first instance has an ID of i-0ed55486d7b2d498b, is a t2.micro type, and has a status check of 2/2 checks passed. The second instance has an ID of i-0c498d6223b8f880f, is also a t2.micro type, and has a status check of 2/2 checks passed. Both instances are in the us-east-2b availability zone. A message at the bottom says 'Select an instance above'.

Copy the DNS name from the instance created

The screenshot shows the PuTTY Configuration window. The left sidebar lists categories: Logging, Terminal, Window, Selection, Colours, Connection, Data, Proxy, SSH (selected), Kex, Host keys, Cipher, Auth (selected), TTY, X11, Tunnels, Bugs, and More bugs. The right pane shows configuration for the selected 'Auth' category. It includes sections for 'Options controlling SSH authentication' (checkboxes for 'Display pre-authentication banner (SSH-2 only)' and 'Bypass authentication entirely (SSH-2 only)'), 'Authentication methods' (checkboxes for 'Attempt authentication using Pageant', 'Attempt TIS or CryptoCard auth (SSH-1)', and 'Attempt "keyboard-interactive" auth (SSH-2)'), 'Authentication parameters' (checkboxes for 'Allow agent forwarding' and 'Allow attempted changes of username in SSH-2'), and a 'Private key file for authentication:' field containing 'C:\Users\socka\Downloads\Jananya.ppk' with a 'Browse...' button. At the bottom are 'About', 'Help', 'Open', and 'Cancel' buttons.

Using putty gen generate the key

Instance summary for i-0d30f6c32e024950c (JananyaSivakumar) [Info](#)

Updated less than a minute ago

Instance ID	Public IPv4 address	Private IPv4 addresses
i-0d30f6c32e024950c (JananyaSivakumar)	-	172.31.45.68 ip-172-31-45-68.us-east-2.compute.internal
Instance state	Private IPv4 DNS copied	
Stopped	<input checked="" type="checkbox"/> Private IPv4 DNS copied	
Instance type	VPC ID	
t2.micro	vpc-5c168737	
AWS Compute Optimizer finding	Subnet ID	
<a href="#">Opt-in to AWS Compute Opti recommendations.</a>   <a href="#">Learn more</a>	subnet-6384da2f	
<a href="#">Details</a>	Security	Monitoring
<a href="#">Instance details</a> <a href="#">Info</a>		disabled
Platform	Termination protection	
Amazon Linux (Inferred)		

Differences available

All changes

The key will be generated

Putty Configuration

Category:

- Keyboard
- Bell
- Features
- Window
  - Appearance
  - Behaviour
  - Translation
  - Selection
  - Colours
- Connection
  - Data
  - Proxy
  - Telnet
  - Rlogin
  - SSH
    - Kex
    - Host keys
    - Cipher
    - Auth
      - Auth
      - PTY
      - X11
      - Tunnels
      - Bugs
      - More bugs
    - Serial

Options controlling SSH authentication

Display pre-authentication banner (SSH-2 only)  
 Bypass authentication entirely (SSH-2 only)

Authentication methods

Attempt authentication using Pageant  
 Attempt TIS or CryptoCard auth (SSH-1)  
 Attempt "keyboard-interactive" auth (SSH-2)

Authentication parameters

Allow agent forwarding  
 Allow attempted changes of username in SSH-2

Private key file for authentication:  
C:\Users\Shakthi T\Downloads\testserver. [Browse...](#)

About Help Open Cancel

Activate the server from AWS via putty by entering the DNS and then going auth and select the ppk file generated

- Banker's Algorithm :

If Request $i$  <= Need $i$

Goto step (2) ; otherwise, raise an error condition, since the process has exceeded its maximum claim.

2) If Request $i$  <= Available

Goto step (3); otherwise, Pi must wait, since the resources are not available.

3) Have the system pretend to have allocated the requested resources to process Pi by modifying the state as follows:

Available = Available –

Request $i$  Allocation $i$  =

Allocation $i$  + Request $i$  Need $i$  =

Need $i$ – Request $i$

1) Let Work and Finish be vectors of length 'm' and 'n' respectively.

2) Initialize: Work = Available

3) Finish[i] = false; for i=1, 2, 3, 4...n

2) Find an i such that both

a) Finish[i] = false

b) Need $i$  <= Work if no such i exists goto step (4)

3) Work = Work + Allocation[i] Finish[i] = true goto step (2)

4) if Finish [i] = true for all i then the system is in a safe state

### Code:

```
#include<stdio.h>
```

```
int current[5][5], maximum_claim[5][5],  
available[5]; int allocation[5] = {0, 0, 0, 0, 0};
```

```
int maxres[5], running[5], safe = 0;
int counter = 0, i, j, exec, resources, processes,
k = 1; int main() {
printf("\nEnter number of processes:
"); scanf("%d", &processes);
for (i = 0; i < processes; i++) {
running[i] = 1; counter++;
} printf("\nEnter number of resources:
"); scanf("%d", &resources);
printf("\nEnter Claim
Vector:");
for (i = 0; i < resources; i++) { scanf("%d",
&maxres[i]); }
printf("\nEnter Allocated Resource
Table:\n");
for (i = 0; i < processes; i++) {
for(j = 0; j < resources; j++) { scanf("%d",
&current[i][j]); } } printf("\nEnter Maximum Claim
Table:\n");
for (i = 0; i < processes;
i++) { for(j = 0; j <
resources; j++) {
scanf("%d", &maximum_claim[i][j]);
} } printf("\nThe Claim Vector is: ");
for (i = 0; i < resources;
i++) { printf("\t%d",
maxres[i]); }
```

```
printf("\nThe Allocated Resource  
Table:\n"); for (i = 0; i < processes; i++) {
```

```
for (j = 0; j < resources;
j++) { printf("\t%d",
current[i][j]); } printf("\n");
printf("\nThe Maximum Claim
Table:\n"); for (i = 0; i < processes;
i++) {
for (j = 0; j < resources; j++) {
printf("\t%d", maximum_claim[i][j]);
} printf("\n");
for (i = 0; i < processes;
i++) { for (j = 0; j <
resources; j++) {
allocation[j] += current[i][j]; }
}
printf("\nAllocated
resources."); for (i = 0; i <
resources; i++)
{ printf("\t%d", allocation[i]);
} for (i = 0; i < resources;
i++)
{ available[i] = maxres[i] - allocation[i];
} printf("\nAvailable resources.");
for (i = 0; i < resources; i++) { printf("\t%d",
available[i]); } printf("\n");
while (counter != 0) {
safe = 0; for (i = 0; i < processes; i++)
```

```
{ if (running[i])  
{ exec = 1;  
for (j = 0; j < resources; j++) {
```

```

if (maximum_claim[i][j] - current[i][j] >
available[j]) { exec = 0; break;
} }
if (exec) {
printf("\nProcess%d is executing\n", i +
1); running[i] = 0;
counter--
; safe =
1;
for (j = 0; j < resources;
j++) { available[j] +=
current[i][j]; } break;
} } }

if (!safe) { printf("\nThe processes are in unsafe
state.\n"); break; }
else { printf("\nThe process is in safe
state"); printf("\nAvailable vector:");
for (i = 0; i < resources; i++) { printf("\t%d",
available[i]); } printf("\n");
} }

return 0;
}

```

Output :

```
Enter number of processes: 2
Enter number of resources: 2
Enter Claim Vector:3
2
Enter Allocated Resource Table:
5
4
5
3

Enter Maximum Claim Table:
5
6
7
4

The Claim Vector is: 3      2
The Allocated Resource Table:
5      4
5      3

The Maximum Claim Table:
5      6
7      4

Allocated resources: 10      7
```

i-0d30f6c32e024950c (JananyaSivakumar)

Public IPs: 3.16.148.50 Private IPs: 172.31.45.68

```
Allocated resources: 10      7
Available resources: -7      -5

The processes are in unsafe state.
[ec2-user@ip-172-31-45-68 ~]$ gcc jananya.c -o jananya
[ec2-user@ip-172-31-45-68 ~]$ ./jananya

Enter number of processes: 5
Enter number of resources: 4
Enter Claim Vector:8
5
9
7

Enter Allocated Resource Table:
2 0 1 1 0 1 2 1 4 0 0 3 0 2 1 0 1 0 3 0

Enter Maximum Claim Table:
3 2 1 4 0 2 5 2 5 1 0 5 1 5 3 0 3 0 3 3

The Claim Vector is: 8      5      9      7
The Allocated Resource Table:
2      0      1      1
0      1      2      1
4      0      0      3
0      2      1      0
1      0      3      0

The Maximum Claim Table:
```

i-0d30f6c32e024950c (JananyaSivakumar)

Public IPs: 3.16.148.50 Private IPs: 172.31.45.68

```

The Maximum Claim Table:
 3   2   1   4
 0   2   5   2
 5   1   0   5
 1   5   3   0
 3   0   3   3

Allocated resources:  7   3   7   5
Available resources: 1   2   2   2

Process3 is executing

The process is in safe state
Available vector:  5   2   2   5

Process1 is executing

The process is in safe state
Available vector:  7   2   3   6

Process2 is executing

The process is in safe state
Available vector:  7   3   5   7

Process4 is executing

The process is in safe state
Available vector:  7   5   6   7

Process5 is executing

```

i-0d30f6c32e024950c (JananyaSivakumar)

Public IPs: 3.16.148.50 Private IPs: 172.31.45.68

```

Process2 is executing

The process is in safe state
Available vector:  7   3   5   7

Process4 is executing

The process is in safe state
Available vector:  7   5   6   7

Process5 is executing

The process is in safe state
Available vector:  8   5   9   7
[ec2-user@ip-172-31-45-68 ~]$ 

```

i-0d30f6c32e024950c (JananyaSivakumar)

Public IPs: 3.16.148.50 Private IPs: 172.31.45.68

## Throttle Algorithm:

```
#include <stdio.h>
```

```
#include
```

```
<semaphore.h?
```

```
#define MAX 50
```

```
#include <time.h>
```

```
void
```

```
inserti();
```

```
void
```

```
deletei();
```

```
void
display();
int
queue_array[MAX];
int rear = - 1;
int fronti = -1;

int tempresources[MAX];
void swap(int *xp, int *yp)
{ int temp = *xp;
*xp = *yp;
*yp = temp;
```

```
}

void inserti(x)
{ int add_item=x;
if (rear == MAX -
1)
printf("Queue Overflow \n");
else { if (fronti == - 1) /*If queue is initially
empty */ fronti = 0;
printf("Insert the job in queue : ");
scanf("%d", &add_item); rear = rear
+ 1; queue_array[rear] = add_item;
}
} /* End of insert()

*/ void deletei() {
if (fronti == - 1 || fronti > rear)
{
printf("Queue Underflow
\n"); return ; } else {
fronti = fronti + 1; }

} void display()
{
int i;
if (fronti == - 1)
printf("Queue is empty
\n"); else { printf("Queue is
: \n"); for (i = fronti; i <=
rear; i++) printf("%d ",
queue_array[i]);
```

```
printf("\n");
} }

struct semaphore

{ int s;

};

struct instances {

int

instances[MAX];

};

void wait(struct semaphore

s) { s.s=0;

}

void signal(struct semaphore

s) { s.s=1;

}

int main()

{ int tr,i,j;

printf("Enter total number of resources

"); scanf("%d",&tr);

Int

totalresources[tr],priority[10],execution_time[10],sort[10],p[10],arrival_

tim e[10];

for(i=0;i<tr;i++)

{

printf("Enter resources of instance

%d\n",i+1); scanf("%d".&totalresources[i]);

Tempresources[i]=totalresources[i];

}
```

```
int tj;
printf("Enter    no    of
jobs\n");
scanf("%d",&tj);//no    of
jobs int a;
for(i=0;i<tr;i++)
{printf("Enter the arrival time of the job
%d\n",i+1); Scanf("%d",&arrival_time[i]);
}
for(i=0;i<tj;i++)
{printf("Enter job no %d\n",i+1);
P[i]=i;
Printf("Enter priority of job
%d\n",i+1); scanf("%d",&priority[i]);
}
for(i=0;i<tj;i++)
{printf("Enter execution time of job
%d\n",i+1); scanf("%d",&execution_time[i]);
tempresources[i]=totalresources[i];
}
for (i = 0; i < tj-1; i++) // Last i elements are already in
place for (j = 0; j < tj-i-1; j++)
if (arrival_time[j] > arrival_time[j+1]) {
swap(&arrival_time[j], &arrival_time[j+1]);
swap(&p[j],&p[j+1]);
swap(&priority[j],&priority[j+1]);
swap(&execution_time[j],&execution_time[j+
1]);
}
```

```

} if (&arrival_time[j]==&arrival_time[j+1]) {
if(&priority[j]>priority[j+1]){
swap(&arrival_time[j], &arrival_time[j+1]);
swap(&p[j],&p[j+1]);
swap(&priority[j],&priority[j+1]);
swap(&execution_time[j],&execution_time[j+
1]);
}
for (i=0;i<tj;i++)
{ //1 2 3 4 5 6
inserti(p[i]);
} struct instances
jobresources[tj]; for
(i=0;i<tj;i++)
{//how much instances
needed printf("for
Job%d\n",i+1);
for (j=0;j<tr;j++)
{printf("resources of type %d needed
\n",j+1);
scanf("%d",&jobresources[i].instances[j]);
}
display();
struct semaphore
s; clock_t t;
t = clock();
for (i=0;i<tj;i++)
{

```

```
printf("for  
Job%d\n",i+1); int c=0;
```

```

for (j=0;j<tr;j++)
{
{ if (totalresources[j]<jobresources[i].instances[j])
{
deletei()
; Break;
}
else
{c++;
if (c==tr) {
wait(s); //critical section
printf("executing process%d \n",i+1);
signal(s); }
else { printf("not executed process%d \n",i+1);

} }

t = clock() - t;
double time_taken = ((double)t)/CLOCKS_PER_SEC;
printf("Job%d took %f seconds to execute
\n",i+1,time_taken);

```

```

Last login: Mon Jun  7 14:50:42 2021 from ec2-13-233-177-0.ap-south-1.compute.amazonaws.com
[ec2-user@ip-172-31-44-163 ~]$ gcc -o throttle3 throttle3.c
[ec2-user@ip-172-31-44-163 ~]$ ./throttle3
Enter total number of resources 3
Enter resources of instance 1
5
Enter resources of instance 2
6
Enter resources of instance 3
7
Enter no of jobs
2
Enter the arrival time of the job 1
2
Enter the arrival time of the job 2
3
Enter the arrival time of the job 3
3
Enter job no 1

```

```
Enter the arrival time of the job 3
3
Enter job no 1
Enter priority of job 1
1
Enter job no 2
Enter priority of job 2
2
Enter execution time of job 1
2
Enter execution time of job 2
3
Insert the job in queue : 1
for Job1
resources of type 1 needed
4
resources of type 2 needed
3
resources of type 3 needed
2
for Job2
resources of type 1 needed
3
resources of type 2 needed
1
```

```
Enter execution time of job 2
3
Insert the job in queue : 1
for Job1
resources of type 1 needed
4
resources of type 2 needed
3
resources of type 3 needed
2
for Job2
resources of type 1 needed
3
resources of type 2 needed
2
resources of type 3 needed
1
Queue is :
1
for Job1
executing process1
for Job2
executing process2
Job3 took 0.000008 seconds to execute
[ec2-user@ip-172-31-44-163 ~]$
```

## DEADLOCK USING PTHREADS:

```
#include<pthread.h>
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
pthread_mutex_t resource1,resource2;
int test=0;
void *proc1()
{
    printf("\nThis is proc1 using rs1");
    pthread_mutex_lock(&resource1);
    usleep(200);
    printf("\nproc1 trying to get rs2...");
    pthread_mutex_lock(&resource2);
    test++;
    printf("\nproc1 got rs2!!!");
    pthread_mutex_unlock(&resource2);
    pthread_mutex_unlock(&resource1);
return 0;
}

void *proc2()
{
    printf("\nThis is proc2 using rs2");
    pthread_mutex_lock(&resource2);
    usleep(200);
    printf("\nproc2 trying to get rs1...");
    pthread_mutex_lock(&resource1);
    test--;
    printf("\nproc2 got rs1!!!");
    pthread_mutex_unlock(&resource1);
    pthread_mutex_unlock(&resource2);
return 0;
}

int main(){
    pthread_t t1,t2;
    pthread_mutex_init(&resource1, NULL);
    pthread_mutex_init(&resource2, NULL);

    pthread_create(&t1,NULL, proc1 , NULL);
    pthread_create(&t2,NULL, proc2 , NULL);
```

```
    pthread_join(t1,NULL);
    pthread_join(t2,NULL);
// will never arrive here
    pthread_mutex_destroy(&resource1);
    pthread_mutex_destroy(&resource2);
}
```

#### DEADLOCK AVOIDANCE USING PTHREADS:

```
#include<stdio.h>
#include<pthread.h>
#include<stdlib.h>
pthread_mutex_t
read_mutex;
pthread_mutex_t
write_mutex; void *
write(void *temp) {
char *ret;
FILE
*file1;
char *str;
```

```
pthread_mutex_lock(&write_mutex
x); sleep(5);
pthread_mutex_lock(&read_mutex
x);
printf("\nFile locked, please enter the message
\n"); str=(char *)malloc(10*sizeof(char));
file1=fopen("temp","w");
scanf("%s",str);
fprintf(file1,"%s",str);
fclose(file1);
pthread_mutex_unlock(&read_mutex);
pthread_mutex_unlock(&write_mutex);
printf("\nUnlocked the file you can read it now
\n"); return ret; }

void * read(void *temp)
{ char *ret; FILE *file1;
char *str;
pthread_mutex_lock(&write_mutex
x); sleep(5);
pthread_mutex_lock(&read_mutex
x); printf("\n Opening file \n");
file1=fopen("temp","r");
str=(char
*)malloc(10*sizeof(char));
fscanf(file1,"%s",str);
printf("\n Message from file is %s
\n",str); fclose(file1);
pthread_mutex_unlock(&read_mutex);
```

```
pthread_mutex_unlock(&write_mute
x); return ret; }

main() {
pthread_t
thread_id,thread_id1;
pthread_attr_t attr;
int ret;
void
*res;

ret(pthread_create(&thread_id,NULL,&write,NU
LL);
ret(pthread_create(&thread_id1,NULL,&read,N
ULL);

printf("\n Created thread");
pthread_join(thread_id,&res);
pthread_join(thread_id1,&res)
: }
```

Save the code as mutex\_nodeadlock.c, compile it using  
"-lpthread" flag Execution command- \$ cc -lpthread  
mutex\_nodeadlock.c -o create\_nodeadlock \$ ./create\_nodeadlock

## Screenshot and Output:

```
void * write(void *temp) {
    pthread_mutex_lock(&write_mutex);
    printf("\nFile locked, please enter the message \n");
    str=(char *)malloc(sizeof(char));
    file1=fopen("temp","w");
    scanf("%s",str);
    fprintf(file1,"%s",str);
    fclose(file1);
    pthread_mutex_unlock(&write_mutex);
    pthread_mutex_lock(&read_mutex);
    printf("\nUnlocked the file you can read it now \n");
    return ret;
}

void * read(void *temp) {
    char *ret;
    FILE *file1;
    char *str;
    pthread_mutex_lock(&read_mutex);
    sleep(5);
    pthread_mutex_lock(&read_mutex);
    printf("\n Opening file \n");
    file1=fopen("temp","r");
    str=(char *)malloc(sizeof(char));
    fscanf(file1,"%s",str);
    printf("\n Message from file is %s \n",str);
    fclose(file1);

    pthread_mutex_unlock(&read_mutex);
    pthread_mutex_unlock(&write_mutex);
    return ret;
}

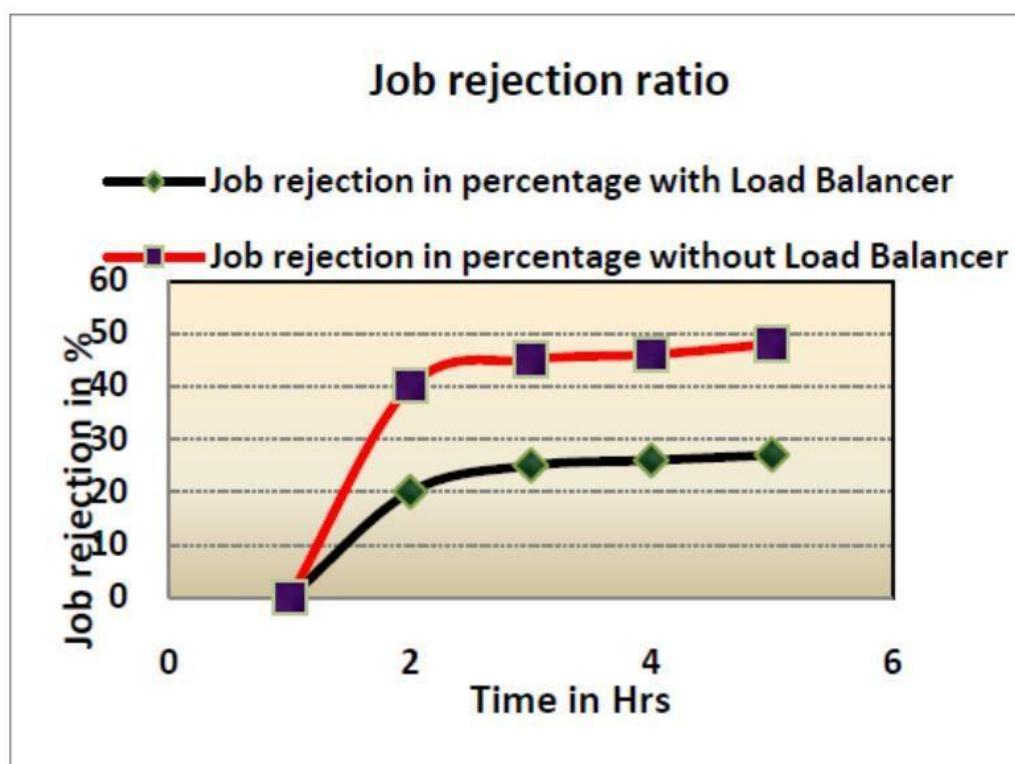
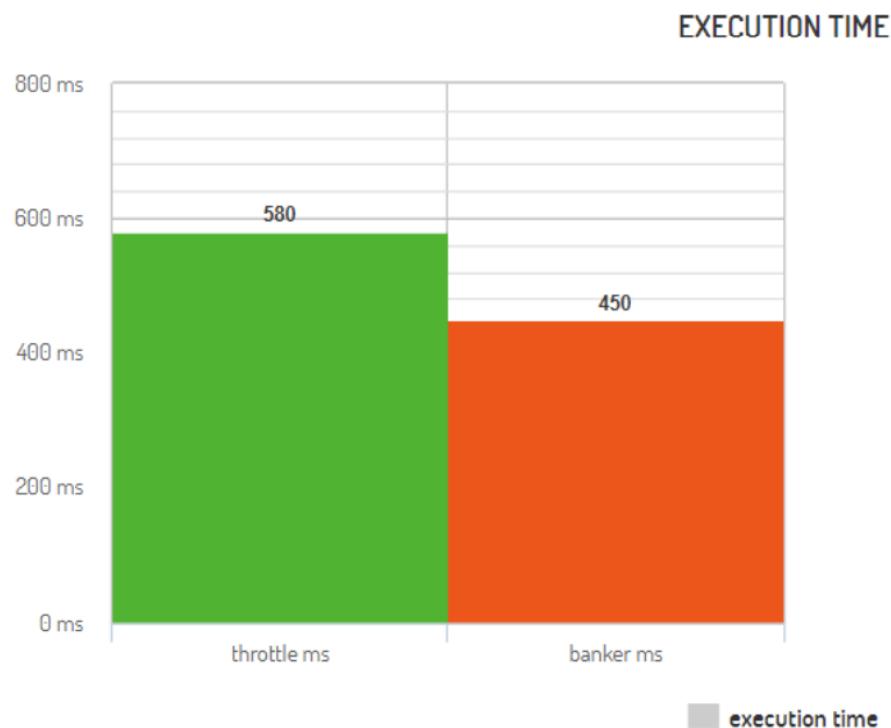
main() {
    pthread_t thread_id,thread_id1;
    pthread_attr_t attr;
    int ret;
    void *res;
    ret=pthread_create(&thread_id,NULL,&write,NULL);
    ret=pthread_create(&thread_id1,NULL,&read,NULL);
    printf("\n Created thread");
    pthread_join(thread_id,&res);
    pthread_join(thread_id1,&res);
}
```

```
Created thread
Opening file

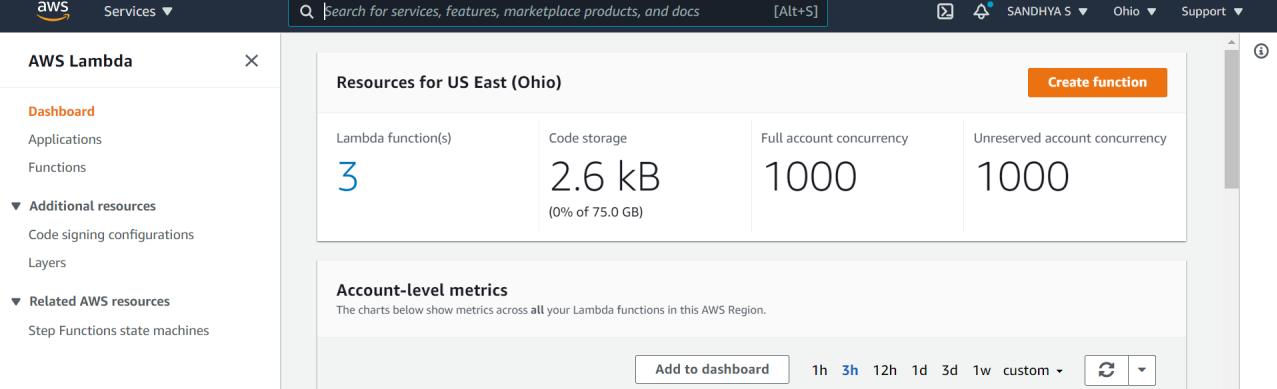
File locked, please enter the message

Message from file is Hello
```

## Comparative Analysis of Algorithm:

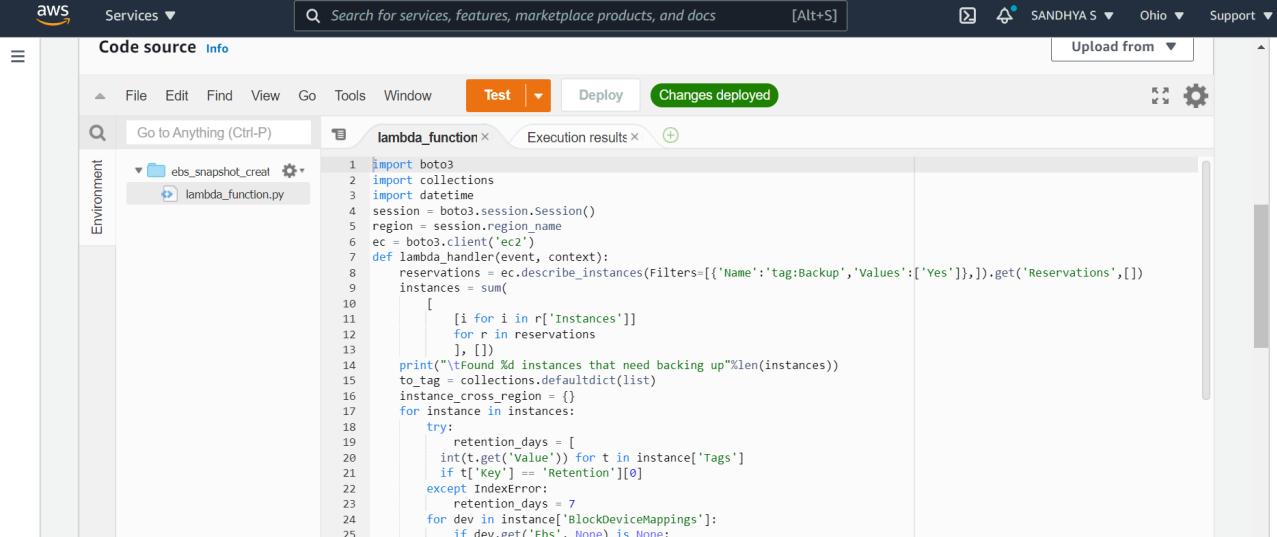


## Disaster Recovery execution in AWS Cloud :



The screenshot shows the AWS Lambda console interface. On the left, there's a sidebar with options like Dashboard, Applications, Functions, Additional resources, and Related AWS resources. The main area displays 'Resources for US East (Ohio)' with summary statistics: 3 Lambda function(s), 2.6 kB of code storage (0% of 75.0 GB), Full account concurrency at 1000, and Unreserved account concurrency at 1000. A large orange 'Create function' button is visible in the top right. Below the summary, there's a section for 'Account-level metrics' with a chart and time selection buttons (1h, 3h, 12h, 1d, 3d, 1w, custom). A 'Create dashboard' button is also present.

Creating function using AWS lambda through EC2 Server to execute codes



The screenshot shows the AWS Lambda code editor for a function named 'lambda\_function'. The code is written in Python and is intended to create EBS snapshots. It uses the 'boto3' library to interact with the AWS services. The code defines a lambda handler 'lambda\_handler' that takes an event and context. It retrieves instances from the 'ec2' service, filters them by tag 'Backup', and then iterates through them to check their retention days. If retention is set to 7 days or more, it creates a new snapshot and updates the instance's tags to reflect the snapshot ID.

```
1 import boto3
2 import collections
3 import datetime
4 session = boto3.session.Session()
5 region = session.region_name
6 ec = boto3.client('ec2')
7 def lambda_handler(event, context):
8     reservations = ec.describe_instances(Filters=[{'Name': 'tag:Backup', 'Values': ['yes']}]).get('Reservations', [])
9     instances = sum([
10         [i for i in r['Instances']]
11         for r in reservations
12     ], [])
13     print(f"\tFound {len(reservations)} instances that need backing up")
14     to_tag = collections.defaultdict(list)
15     instance_cross_region = {}
16     for instance in instances:
17         try:
18             retention_days = [
19                 int(t.get('Value')) for t in instance['Tags']
20                 if t['Key'] == 'Retention'][0]
21             except IndexError:
22                 retention_days = 7
23             for dev in instance['BlockDeviceMappings']:
24                 if dev.get('Ebs', None) is None:
25                     continue
26                 if retention_days >= 7:
27                     volume_id = dev['Ebs'].get('VolumeId')
28                     snapshot_id = ec.create_snapshot(VolumeId=volume_id, Description=f"Snapshot created by Lambda function {context.function_name} on {datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S')}")
```

After creation of hosting server with JSON as policy ebs-snapshot-creator as snapshot

SANDHYA S ▾ Ohio ▾ Support ▾

aws Services ▾

Search for services, features, marketplace products, and docs [Alt+S]

Go to Anything (Ctrl-P)

lambda\_function Execution result

Execution results

Test Event Name creator

Response null

Function Logs

```
START RequestId: 20f782c6-d18a-4635-93d1-88687c959908 Version: $LATEST
    Found 1 instances that need backing up
*****1*****
    Found EBS volume vol-0da4e5e38558eed5 (/dev/xvda) on instance i-0993173acaa34201c
**"debug1**
Debug 2*****
cross_region us-east-2
    Snapshot snap-0ce724fbca22840d9 created in us-east-2 of [- vol-0da4e5e38558eed5 (/dev/xvda)]
    Retaining snapshot snap-0ce724fbca22840d9 of volume vol-0da4e5e38558eed5 from instance i-0993173acaa34201c () for 5 days
END RequestId: 20f782c6-d18a-4635-93d1-88687c959908
REPORT RequestId: 20f782c6-d18a-4635-93d1-88687c959908 Duration: 649.75 ms Billed Duration: 650 ms Memory Size: 128 MB Max Memory Used: 85 MB
```

Status: Succeeded | Max memory used: 85 MB | Time: 649.75 ms

Environment

ebs\_snapshot\_create lambda\_function.py

Request ID 20f782c6-d18a-4635-93d1-88687c959908

## The execution of the code where the backup is being retained

Screenshot of the AWS Management Console showing the EC2 Snapshots page. The left sidebar shows navigation options like Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Capacity Reservations, Images, AMIs, Elastic Block Store, Volumes, Snapshots (selected), Lifecycle Manager, Network & Security, Security Groups, Elastic IPs, Placement Groups, Key Pairs, and Network Interfaces. The main content area shows a table titled 'Owned By Me' with columns: Name, CreatedOn, Sandhya trial, Snapshot ID, Size, and Description. A search bar at the top right says 'Search for services, features, marketplace products, and docs [Alt+S]'. The table lists several snapshots, including one selected ('snap-0ce724fbca22840d') which is highlighted with a blue border. Below the table, a detailed view for the selected snapshot is shown with tabs for Description, Permissions, and Tags. The 'Description' tab is active, showing the Snapshot ID as 'snap-0ce724fbca22840d', Status as 'completed', Progress as '100%', and Capacity as '8 GiB'.

Name	CreatedOn	Sandhya trial	Snapshot ID	Size	Description
	2021-06-07		snap-062a2249e70...	8 GiB	- vol-0da4e55e38558eed5 (/dev/xvda)
Selected	2021-06-09		snap-0ce724fbca22...	8 GiB	- vol-0da4e55e38558eed5 (/dev/xvda)
	2021-06-09		snap-0efa588da25a...	8 GiB	- vol-0da4e55e38558eed5 (/dev/xvda)
	2021-06-08		snap-025e5e01ac9...	8 GiB	- vol-0da4e55e38558eed5 (/dev/xvda)
	2021-06-08		snap-0e540dea954...	8 GiB	- vol-0da4e55e38558eed5 (/dev/xvda)
	2021-06-07		snap-0b2c3cd9b28...	8 GiB	- vol-0da4e55e38558eed5 (/dev/xvda)
	2021-06-07		snap-03b6d6d4449...	8 GiB	- vol-0da4e55e38558eed5 (/dev/xvda)
	2021-06-07		snap-096a193f89b2...	8 GiB	- vol-0da4e55e38558eed5 (/dev/xvda)

**Snapshot: snap-0ce724fbca22840d**

Description	Permissions	Tags
Snapshot ID: snap-0ce724fbca22840d		
Status: completed		
	Progress: 100%	
	Capacity: 8 GiB	

As we can see the status of the snapshot shows completed

The screenshot shows the AWS Lambda code editor interface. The top navigation bar includes the AWS logo, 'Services ▾', a search bar ('Search for services, features, marketplace products, and docs [Alt+S]'), and user information ('SANDHYA S ▾ Ohio ▾ Support ▾'). Below the navigation is a tab bar with 'Code' (selected), 'Test', 'Monitor', 'Configuration', 'Aliases', and 'Versions'. A toolbar below the tabs includes 'File', 'Edit', 'Find', 'View', 'Go', 'Tools', 'Window', 'Test' (highlighted in orange), 'Deploy' (disabled), and 'Changes not deployed'. On the left, a sidebar shows the 'Environment' section with a dropdown for 'ebs\_snapshot\_manager' and a file list containing 'lambda\_function.py'. The main code editor area displays the following Python code:

```

1 import boto3
2 import re
3 import datetime
4 import time
5 ec = boto3.client('ec2')
6 iam = boto3.client('iam')
7 def lambda_handler(event, context):
8     account_ids = list()
9     try:
10         iam.get_user()
11     except Exception as e:
12         today = datetime.datetime.now()
13         print("\nDelete snapshots < %s " % today)
14         filters = [
15             {'Name': 'tag:Type', 'Values': ['Automated']},
16         ]
17         snapshot_response = ec.describe_snapshots(OwnerIds=account_ids, Filters=filters)
18         for snap in snapshot_response['Snapshots']:
19             skipping_this_one = False
20             snapshot_delete_on = ""
21             for tag in snap['Tags']:

```

## Ebs-snapshot-manager execution code

The screenshot shows the AWS EBS console. The left sidebar includes 'Capacity Reservations', 'Images' (AMIs), 'Elastic Block Store' (Volumes, Snapshots, Lifecycle Manager), 'Network & Security' (Security Groups, Elastic IPs, Placement Groups, Key Pairs, Network Interfaces), and 'Load Balancing' (Load Balancers, Target Groups). The main area shows a table of snapshots under the 'Schemas' heading. The table has columns: Name, CreatedOn, Snapshot ID, Size, and Description. A filter bar at the top allows filtering by tags and attributes or searching by keyword. The table lists 23 snapshots, all of which are labeled as 'Automated' and were created on June 7, 2021. The last row shows summary details: BackupCrossRegion (us-east-2), CreatedOn (2021-06-09), DeleteOn (2021-06-14), and Type (Automated).

Name	CreatedOn	Snapshot ID	Size	Description
	2021-06-07	snap-062a2249e70...	8 GiB	- vol-0da4e55e38558eed5 (/dev/xvda)
■	2021-06-09	snap-0ce724fbca22...	8 GiB	- vol-0da4e55e38558eed5 (/dev/xvda)
	2021-06-09	snap-0efa588da25a...	8 GiB	- vol-0da4e55e38558eed5 (/dev/xvda)
	2021-06-08	snap-025e50e01ac9...	8 GiB	- vol-0da4e55e38558eed5 (/dev/xvda)
	2021-06-08	snap-0e540dea954...	8 GiB	- vol-0da4e55e38558eed5 (/dev/xvda)
	2021-06-07	snap-0b2c3cd9b28...	8 GiB	- vol-0da4e55e38558eed5 (/dev/xvda)
	2021-06-07	snap-03b6d6d4449...	8 GiB	- vol-0da4e55e38558eed5 (/dev/xvda)
	2021-06-07	snap-096a193f89b2...	8 GiB	- vol-0da4e55e38558eed5 (/dev/xvda)
BackupCrossRegion		us-east-2		
CreatedOn		2021-06-09		
DeleteOn		2021-06-14		
Type		Automated		

Creation of snapshot after the code and the tags and key being generated

SANDHYA S ▾ Ohio ▾ Support ▾

Capacity Reservations

Images

AMIs

Elastic Block Store

Volumes

Snapshots

Lifecycle Manager

Network & Security

Security Groups New

Elastic IPs New

Placement Groups

Key Pairs

Network Interfaces New

Load Balancing

Load Balancers

Target Groups New

Create Snapshot Actions ▾

Owned By Me Filter by tags and attributes or search by keyword

Name	CreatedOn	Sandhya trial	Snapshot ID	Size	Description
2021-06-07	snap-062a2249e70...	8 GiB	- vol-0da4e55e38558eed5 (/dev/xvda)		
2021-06-09	snap-0ce724fbc22...	8 GiB	- vol-0da4e55e38558eed5 (/dev/xvda)		
2021-06-09	snap-0efa588da25a...	8 GiB	- vol-0da4e55e38558eed5 (/dev/xvda)		
2021-06-08	snap-025e5e01ac9...	8 GiB	- vol-0da4e55e38558eed5 (/dev/xvda)		
2021-06-08	snap-0e540dea954...	8 GiB	- vol-0da4e55e38558eed5 (/dev/xvda)		
2021-06-07	snap-0b2c3cd9b28...	8 GiB	- vol-0da4e55e38558eed5 (/dev/xvda)		
2021-06-07	snap-03b6d6d4449...	8 GiB	- vol-0da4e55e38558eed5 (/dev/xvda)		
2021-06-07	snap-096a193f80b2...	8 GiB	- vol-0da4e55e38558eed5 (/dev/xvda)		

Show Column Hide Column Show Column Show Column

We try changing the value of deleted on to see the update in snapshot test





Function name  
Enter a name that describes the purpose of your function.

Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime [Info](#)  
Choose the language to use to write your function.

Permissions [Info](#)  
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

▼ Change default execution role

Execution role  
Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).

Create a new role with basic Lambda permissions  
 Use an existing role  
 Create a new role from AWS policy templates

Existing role  
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.

▼ C

## Creation of the lambda function ebs-cross-region-backup

```

1 import boto3
2 import re
3 import datetime
4 session = boto3.session.Session()
5 source_region = session.region_name
6 ec = boto3.client('ec2')
7 iam = boto3.client('iam')
8 def lambda_handler(event, context):
9     account_ids = list()
10    try:
11        iam.get_user()
12    except Exception as e:
13        today_fmt = datetime.date.today().strftime('%Y-%m-%d')
14        filters = [
15            {'Name': 'tag:CreatedOn', 'Values': [today_fmt]},
16            {'Name': 'tag>Type', 'Values': ['Automated']},
17            {'Name': 'tag:BackupCrossRegion', 'Values': ['*']}
18        ]
19        snapshot_response = ec.describe_snapshots(OwnerIds=account_ids, Filters=filters)
20        for snap in snapshot_response['Snapshots']:
21            print("\tCopying %s created from %s of [%s]" % (snap['SnapshotId'], source_region, snap['Description']))
22            target_regions = [
23                t.get('Value') for t in snap['Tags']
24            if t['Key'] == 'BackupCrossRegion'][0]
25            for target in target_regions.split(','):
26                print("\t\tto %s" % target)
27                addl_ec = boto3.client('ec2', region_name=target)
28                addl_snap = addl_ec.copy_snapshot(

```

**Execution result:**

- Test Event Name:** region
- Response:** null
- Function Logs:**
  - START RequestId: d37ce63a-9b5b-4199-997d-35999afde098 Version: \$LATEST
  - Copying snap-0ce724fbca22840d9 created from us-east-2 of [- vol-0da4e55e38558eed5 (/dev/xvda)] to us-east-2
  - Copying snap-0efa588da25ade44d created from us-east-2 of [- vol-0da4e55e38558eed5 (/dev/xvda)] to us-east-2
  - END RequestId: d37ce63a-9b5b-4199-997d-35999afde098
  - REPORT RequestId: d37ce63a-9b5b-4199-997d-35999afde098 Duration: 1846.11 ms Billed Duration: 1847 ms Memory Size: 128 MB Max
- Request ID:** d37ce63a-9b5b-4199-997d-35999afde098

## Ebs-cross-region-backup code and execution

**Functions (3)**

Function name	Description	Package type	Runtime	Code size	Last modified
ebs_snapshot_cross_region	Zip	Python 3.8	765.0 byte	4 days ago	
ebs_snapshot_creator	Zip	Python 3.8	1.1 kB	4 days ago	
ebs_snapshot_manager	Zip	Python 3.8	723.0 byte	4 days ago	

## Creation of the three lambda functions

The screenshot shows the AWS CloudWatch Rules console. On the left, a navigation sidebar lists various AWS services like Log groups, Metrics, Events, and Lambda Insights. The 'Events' section is expanded, and 'Rules' is selected. The main pane displays the 'Rules > DailyBackup' page. The 'Summary' section includes the ARN (arn:aws:events:us-east-2:968659209465:rule/DailyBackup), a cron schedule (Cron expression: 2 1 \* \* ? \*), and a list of the next 10 triggers. The rule is marked as 'Enabled'. A description states '01:02 GMT Run Snapshot Backup and Cleanup the deadlock'. A 'Monitoring' link is present.

The screenshot shows the 'Step 1: Create rule' wizard. The left sidebar is identical to the previous screenshot. The main area is titled 'Step 1: Create rule' and contains instructions to 'Create rules to invoke Targets based on Events happening in your AWS environment.' It has two main sections: 'Event Source' and 'Targets'. In 'Event Source', there are two options: 'Event Pattern' (radio button) and 'Schedule' (radio button, which is selected). Under 'Schedule', there are two sub-options: 'Fixed rate of' (radio button) and 'Cron expression' (radio button, selected). The 'Cron expression' field contains '2 1 \* \* ? \*'. Below this is a link to 'Learn more about CloudWatch Events schedules'. In the 'Targets' section, there are two entries for 'Lambda function': one for 'ebs\_snapshot\_creator' and another for 'ebs\_snapshot\_manager'. Each entry has a 'Function\*' dropdown set to the respective name, and 'Configure version/alias' and 'Configure input' buttons.

## Creation of trigger for ebs-snapshot-creator and ebs-snapshot-manager

The screenshot shows the 'Step 2: Configure rule details' wizard. The left sidebar is identical. The main area is titled 'Step 2: Configure rule details' and 'Rule definition'. It includes fields for 'Name\*' (set to 'DailyBackup'), 'Description' (set to '01:02 GMT Run Snapshot Backup and Cleanup the deadlock'), and 'State' (checkbox checked, set to 'Enabled'). A note at the bottom states 'CloudWatch Events will add necessary permissions for target(s) so they can be invoked when this rule is triggered.' At the bottom right are 'Cancel', 'Back', and 'Update rule' buttons.

## Configuring the trigger as a daily backup

**Step 1: Create rule**

Create rules to invoke Targets based on Events happening in your AWS environment.

**Event Source**

Build or customize an Event Pattern or set a Schedule to invoke Targets.

Event Pattern i  Schedule i

Fixed rate of  Select ▼

Cron expression

Learn more about CloudWatch Events schedules.

[Show sample event\(s\)](#)

**Targets**

Select Target to invoke when an event matches your Event Pattern or when schedule is triggered.

**Lambda function**

Function\*

[Configure version/alias](#)

[Configure input](#)

[Add target\\*](#)

\* Required

[Cancel](#) [Configure details](#)

## Creation of trigger or rule for ebs-cross-region-backup

**Success**

Rule CrossRegionBackup was updated.

**CloudWatch Events is now Amazon EventBridge**

Amazon EventBridge (formerly CloudWatch Events) provides all functionality from CloudWatch Events and also launched new features such as Custom event buses, 3rd party event sources and Schema registry to better support our customers in the space of event-driven architecture and applications.

[Amazon EventBridge documentation](#)

**Rules**

Rules route events from your AWS resources for processing by selected targets. You can create, edit, and delete rules.

[Create rule](#) [Actions ▾](#)

Status	All	Name	Description
Status	Name		

## After creation of both the rules/trigger



aws Services ▾

Search for services, features, marketplace products, and docs [Alt+S]

SANDHYA S ▾ Ohio ▾ Support ▾

Capacity Reservations

Images

AMIs

Elastic Block Store

Volumes

**Snapshots**

Lifecycle Manager

Network & Security

Security Groups New

Elastic IPs New

Placement Groups

Key Pairs

Network Interfaces New

Load Balancing

Load Balancers

**Create Image from EBS Snapshot**

Name	Os proj	Description							
Architecture	x86_64	Virtualization type	Hardware-assisted virtualization						
Root device name	/dev/sda1	Kernel ID	Use default						
RAM disk ID	Use default	Boot Mode	Use default						
Block Device Mappings									
Volume Type	Device	Snapshot	Size (GiB)	Volume Type	IOPS	Throughput (MB/s)	Delete on Termination	Encrypted	
Root	/dev/sda1	snap-062a2249e7070bf64	8	General Purpose S	100 / 3000	N/A	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Not Encrypted
<b>Add New Volume</b>									

**Cancel** **Create**

Creating image for EBS snapshot

# Create Image from EBS Snapshot

X



Create Image request received.

View pending image [ami-04afe43ded36a879a](#)

[Close](#)

Image being requested and received

The screenshot shows the AWS EC2 console with the 'Images' section selected. On the left, there's a sidebar with 'Instances' (including 'Instances', 'Instance Types', 'Launch Templates', 'Spot Requests', 'Savings Plans', 'Reserved Instances', 'Dedicated Hosts', 'Capacity Reservations'), 'Images' (with 'AMIs' highlighted), and 'Elastic Block Store' (with 'Volumes', 'Snapshots', 'Lifecycle Manager'). The main area has a search bar at the top and a table below it. The table has columns: Name, AMI Name, AMI ID, Source, Owner, Visibility, Status, and Creation Date. It shows two rows: one for 'Os proj' (AMI ID: ami-04afe43ded36a879a) and another for 'recovery' (AMI ID: ami-0845f8a924df1a34d). Both rows have 'Private' as the owner, 'available' as the status, and dates from June 10, 2021, and June 6, 2021. A message at the bottom says 'Select an AMI above'.

Name	AMI Name	AMI ID	Source	Owner	Visibility	Status	Creation Date
Os proj	ami-04afe43ded36a879a	968659209465/...	968659209465	Private	available	June 10, 2021 at...	
recovery	ami-0845f8a924df1a34d	968659209465/...	968659209465	Private	available	June 6, 2021 at...	

The recovered backup being generated

**Eb-snapshot-creator code :**

```
import boto3
import
collections
import datetime
session =
boto3.session.Session()
region = session.region_name
ec = boto3.client('ec2')
```

```
def lambda_handler(event, context):
    reservations =
        ec.describe_instances( Filters=[
            {'Name': 'tag:Backup', 'Values': ['Yes']},
        ])
    ).get(
        'Reservations',
        []
    )
    instances =
        sum([
            [i for i in
                r['Instances']] for r in
            reservations
        ], [])
    print("\tFound %d instances that need backing up" %
        len(instances))
    to_tag = collections.defaultdict(list)
    instance_cross_region
    = {} for instance in
    instances:
        try:
            retention_days =
                int(t.get('Value')) for t in
                instance['Tags'] if t['Key'] ==
            'Retention'][0]
        except
            IndexError:
                retention_days =
```

7

```
for dev in
instance['BlockDeviceMappings']: if
dev.get('Ebs', None) is None:
    Continue
```

```

vol_id =
dev['Ebs']['VolumeId']
dev_name =
dev['DeviceName']
print("\tFound EBS volume %s (%s) on instance %s" % (vol_id,
dev_name, in stanc['InstanceId']))

# figure out instance name & if cross region backup
wanted instance_name = ""
cross_region = ""

for tag in instance["Tags"]:
if tag['Key'] == 'Name':
instance_name = tag["Value"]
if tag["Key"] == 'BackupCrossRegion':
cross_region = tag["Value"]
description = '%s - %s (%s)' % (instance_name, vol_id,
dev_name) snap = ec.create_snapshot(
VolumeId=vol_id,
Description=descripti
on
)
if (snap):
print("\tSnapshot %s created in %s of [%s]" % (snap["SnapshotId"],
region, d escription))
to_tag[retention_days].append(snap['SnapshotId'])
print(
"\tRetaining snapshot %s of volume %s from instance %s (%s) for %d
days"
% (
snap["SnapshotId"]

```

'], vol\_id,

```

instance['InstanceId
'], instance_name,
retention_days,
))

today_fmt = datetime.date.today().strftime("%Y-%m-%d")

delete_date = datetime.date.today() +
datetime.timedelta(days=retention_days)

delete_fmt =
delete_date.strftime('%Y-%m-%d')

ec.create_tags(
Resources=[snap['SnapshotId'
], ], Tags=[

{'Key': 'CreatedOn', 'Value': today_fmt},
{'Key': 'DeleteOn', 'Value': delete_fmt},
{'Key': 'Type', 'Value': 'Automated'},
]
)
if cross_region:
    ec.create_tags(
Resources=[snap['SnapshotId'
], ], Tags=[

{'Key': 'BackupCrossRegion', 'Value': cross_region}
])

```

### **Eb-snapshot-manager code :**

```

import boto3
import re
import
datetime

```

```
import time
ec = boto3.client('ec2')

iam =
boto3.client('iam')

def lambda_handler(event, context):
    account_ids =
list() try:
    iam.get_user()
except Exception as e:
    today = datetime.datetime.now()
    print("\tDelete snapshots < %s " %
today) filters = [
{'Name': 'tag>Type', 'Values': ['Automated']}]
snapshot_response = ec.describe_snapshots(OwnerIds=account_ids,
Filters
=filters)

for snap in
snapshot_response['Snapshots']:
    skipping_this_one = False
    snapshot_delete_on = ""
    for tag in snap['Tags']:
        if tag['Key'] == 'KeepForever':
            skipping_this_one =
True if tag['Key'] ==
'DeleteOn':
            snapshot_delete_on =
tag['Value'] if skipping_this_one
            == True:
```

```
print("\tSkipping snapshot %s marked KeepForever" %  
(snap['SnapshotId'])) else:  
if snapshot_delete_on == "":
```

```

print("\tThere is no DeleteOn Tag for this backup: %s" %
(snap['SnapshotId'])
)
continue

snapshot_expires = datetime.datetime.strptime(snapshot_delete_on,
"%Y-
%m-%d")

if snapshot_expires < today:
    print("\tDeleting %s with expiry date %s" % (snap['SnapshotId'],
snapshot_d elete_on))
    ec.delete_snapshot(SnapshotId=snap['Snapsh
otId']) else:
    print("\tKeeping %s until %s" % (snap['SnapshotId'],
snapshot_delete_on))

```

### **Eb-snapshot-cross-regions code :**

```

import
boto3
import re
import datetime
session = boto3.session.Session()

source_region =
session.region_name ec =
boto3.client('ec2')

iam = boto3.client('iam')

def lambda_handler(event,
context): account_ids = list()

try:
    iam.get_user()
except Exception as e:
    today_fmt =

```

`datetime.date.today().strftime('%Y-%m-%d')` filters =

[

```

        {'Name': 'tag:CreatedOn', 'Values': [today_fmt]},
        {'Name': 'tag>Type', 'Values': ['Automated']},
        {'Name': 'tag:BackupCrossRegion', 'Values': ['*']}
    ]
snapshot_response =
ec.describe_snapshots(OwnerIds=account_ids, Filters=filters)
for snap in snapshot_response['Snapshots']:
    print("\tCopying %s created from %s of [%s]" % (snap['SnapshotId'],
source
_region,
snap['Description']))
target_regions = [
t.get('Value') for t in snap['Tags']
if t['Key'] ==
'BackupCrossRegion'][0] for
target in target_regions.split(','):
    print("\t\tto %s" % target)
addl_ec = boto3.client('ec2',
region_name=target) addl_snap =
addl_ec.copy_snapshot(
SourceRegion=source_region,
SourceSnapshotId=snap['SnapshotId'],
Description=snap['Description'],
DestinationRegion=target
)
addl_ec.create_tags(
Resources=[addl_snap['SnapshotId']],
Tags=snap['Tags']
)

```

```
addl_ec.delete_tags(  
    Resources=[addl_snap['SnapshotI  
d']],
```

```
Tags=[{"Key": "BackupCrossRegion"}]  
)  
addl_ec.create_tags(  
Resources=addl_snap['Snapshotl  
d'],  
Tags=[{'Key': 'BackupFromRegion', 'Value': source_region}]  
)
```

## **Conclusion:**

The advancement in the Cloud computing model has led to an increase in demand for its services. An efficient and effective performance of this model is essential to have better business returns. The performance of this model is influenced by various factors like Scalability, Reliability, Security, Disaster Management etc. Disaster is one of the factors which deteriorate the Cloud performance. Though it is an unpredictable event it can be prevented if the system is designed with efficient Scheduling, Resource allocation, Load Balancing etc. Disaster management is a challenging task, it provides a platform for a Cloud Service provider to exhibit the best service. It is essential for the Cloud Service providers to apply appropriate metrics to monitor the Cloud performance. Due to the growing demand for the cloud services a short period of downtime would result in significant financial loss. Disaster is one of the several factors which influence the downtime to occur in this computing environment. In the proposed Throttle Algorithm to balance the load in the computing model the Deadlock Probability is Less and Job Rejection Ratio is Less. The algorithm maintains the status of the resources periodically. It rejects the jobs exceeding the Job queue size.

## **Scopes of Improvement :**

In the future we would like to remove the queue limit and decrease the execution time.

## **References:**

1. Mahitha.O , Suma. V , “Deadlock Avoidance through Efficient Load Balancing to Control Disaster in Cloud Environment.”, Proceedings of the IEEE - 31661 International Conference on Computer Science and Information Technology, July 4 - 6, 2013.
2. Mr.Akshay A. Gharat, Mr. Devendra E. Mhamunkar , “Disaster Recovery in Cloud Computing”, In: 2010 10th Annual International Symposium on Applications and the Internet IJARCET, pp. 304–307 2015
3. Chintureena Thingom,”A Study on Tools for Cloud Disaster Management”, International Journal of Interdisciplinary and Multidisciplinary Studies (IJIMS), 2014, Vol 1, No.4, 82- 86.
4. Mahitha.O , Suma. V, “Effective Disaster Management to Enhance the Cloud Stability”,International Conference on Electrical Engineering and Computer Science (IEEE),2nd june 2013, pp.50-54,2013,Bangalore,India.
5. Sinung Suakanto, Suhono H. Supangkat, Suhardi, Roberd Saragih, Tunggul Arief Nugroho, I Gusti Bagus Baskara Nugraha, “Environmental and Disaster Sensing Using Cloud Computing Infrastructure”, In: International Conference on Cloud Computing and Social Networking (ICCCSN), pp. 1–6 (2012)

