Media Streaming with IBM Cloud Video Streaming

Phase 5 Submission Document

Media Streaming :

Stíeaming media is multimedia foí playback using an offline oí online media playeí. ľechnically, the stíeam is deliveéed and consumed in a continuous manneí fíom a client, with little oí no inteímediate stoíage in netwoík elements. Stíeaming íefeís to the deliveíy method of content, íatheí than the content itself.

Objects:

Media streaming in cloud computing involves delivering audio and video content to users over the internet from cloud-based servers. Here's how it works:

Content Storage: Media files (audio and video) are stored in cloud storage solutions like Amazon S3, Google Cloud Storage, or Azure Blob Storage. These files are ofter stored in different formats and bitrates to accommodate various user devices and network conditions

Content Encoding: Before streaming, the media files may be encoded into different quality levels (bitrates and resolutions) to support adaptive streaming. This allows the service to adjust the quality in real-time based on the viewer's internet connection and device capabilities.

Content Delivery: Content is delivered to end-users through Content Delivery Networks (CDNs) for efficient and scalable distribution. CDNs cache content in multiple locations around the world to reduce latency and improve delivery speed.

Content Delivery: Content is delivered to end-users through Content Delivery Networks (CDNs) for efficient and scalable distribution. CDNs cache

content in multiple locations around the world to reduce latency and improve delivery speed.

Streaming Protocols: Common streaming protocols used in cloud media streaming include HTTP Live Streaming (HLS), Dynamic Adaptive Streaming over HTTP (DASH), and Real-Time Messaging Protocol (RTMP). These protocols enable adaptive streaming and live streaming capabilities.

Transcoding: Transcoding can be performed on the fly in the cloud to adapt content for different devices and network conditions. For example, a mobile device may receive a lower bitrate stream, while a smart TV receives a higher quality stream.

Security: Cloud media streaming services typically include security measures such as encryption (e.g., HTTPS for content delivery), access control, and digital rights management (DRM) to protect the content from unauthorized access.

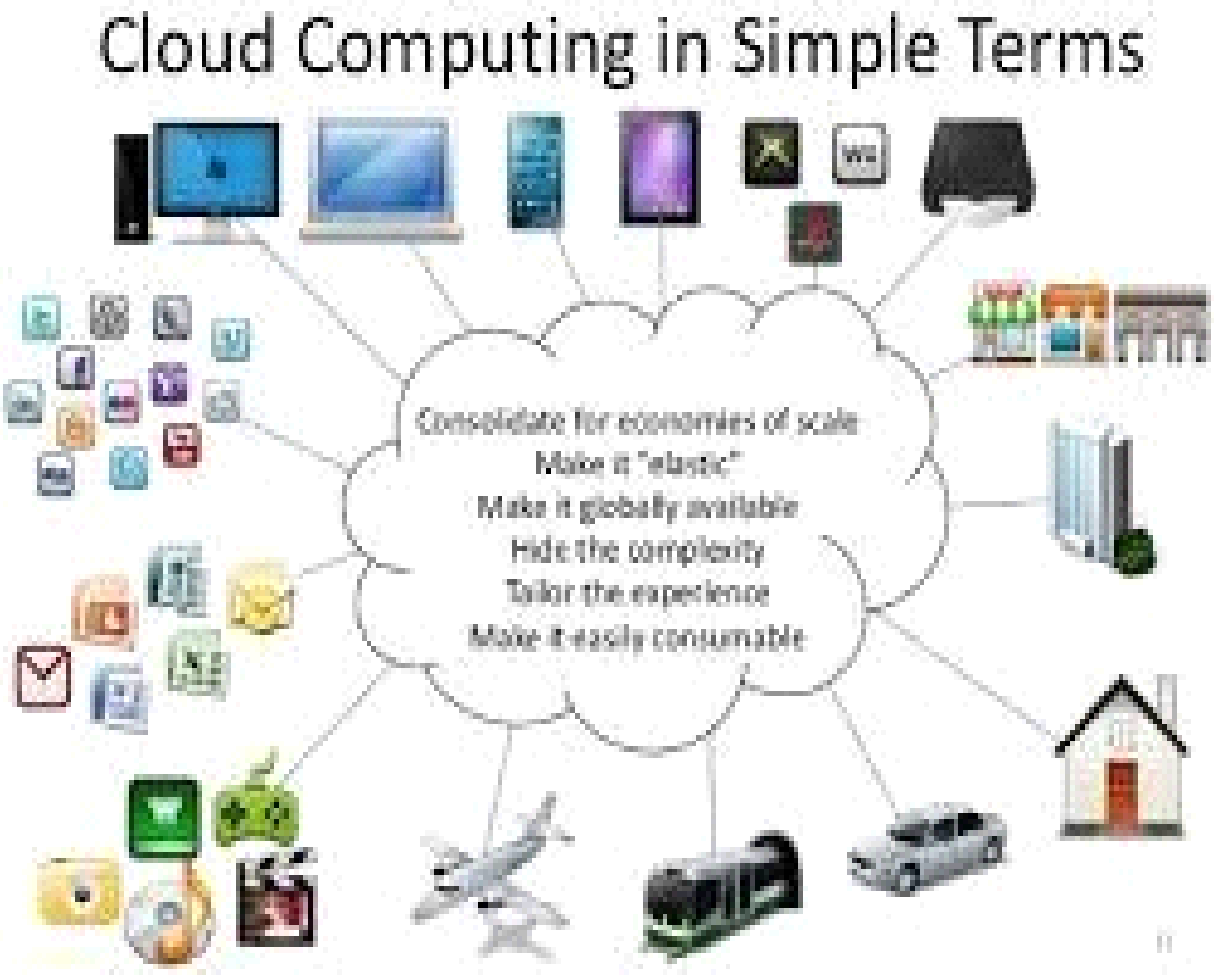
Scalability: Cloud computing allows for easy scalability. When there is a surge in demand for streaming content, cloud resources can be dynamically provisioned to handle the increased load, and scaled down during periods of lower demand.

Monitoring and Analytics: Cloud-based media streaming services often include tools for monitoring the performance of streams and analyzing viewer behavior.

Cost Management: Cloud computing offers cost flexibility, as you pay for the resources you use. This can be cost-effective for media streaming services, especially when dealing with varying traffic patterns.

Global Reach: Cloud providers have data centers worldwide, allowing content to be delivered efficiently to a global audience. This is crucial for media streaming with an international user base.

Design thinking process:



It's a type of computing that relies on sharing computing resources rather than having local servers or personal devices to handle applications. The word cloud, (also phrased as "the cloud")is used as a metaphor for "the Internet," so the phrase cloud computing means "a type of Internet-based computing," where different services — such as servers, storage and applications

— are delivered to an organization's computers and devices through the Internet.

Cloud Computing Terminologies

Cluster

A group of linked computers that work together as if they were a single computer, for high

availability and/or load balancing.

Elastic computing

The ability to dynamically provision and deprovision computing and storage resources to stretch to the demands of peak usage, without the need to worry about capacity planning and engineering around uneven usage patterns.

Docker

Open-source software that automates the deployment of applications inside virtualized software containers.

External cloud Public or private cloud services that are provided by a third party outside the organization.

Advantages

Cost Efficient

- Cloud computing is probably the most cost efficient method to use, maintain and upgrade.
- The cloud, on the other hand, is available at much cheaper ratesand hence, can significantly lower the company's IT expenses.

Backup and Recovery

- Since all your data is stored in the cloud, backing it up and restoring the same is relatively much easier than storing the same on a physical device.
- Hence, this makes the entire process of backup and recovery muchsimpler than other traditional methods of data storage.

Quick Deployment

Cloud technology for Media Streaming:

We live in a digital world where the linear television system that involves traditional broadcasting is outdated. People now prefer OTT platforms that give them the liberty to watch their favorite content anytime and anywhere.

Media and entertainment companies must keep up with customers' demands and expectations. They must stay loyal to these customers and give them more of what they want. The need for cloud computing in the media and entertainment industry is increasing exponentially

Use case;Netflix

Netflix is an example of a cloud-based streaming OTT provider. Netflix consumes a large amount of data which it processes through a Cloud Media Player and a Cloud- based Media Server. These services include streaming, transcoding, video processing, and much more. You've seen some cloud video streaming sites, like Hulu and YouTube.

Cloud Based Audio Streaming

Stream audio to your device from the cloud with music services like Apple Music and Spotify. These apps allow you to listen anytime and enjoy a large library of high-quality songs. Some cloud-based music streaming services also

provide cloud storage for audio files so you can upload your recordings remotely or record them

Use Case: Spotify

A great example of a cloud-based music streaming service is Spotify. It allows you to listen to music as an audio player, streamed over the internet, and stored in “a cloud.”

Spotify is also great for storing your playlists and songs so that you can listen to them again later.

Offline mode is qexcellent for those who like to use public transport!

```
from flask
import
Flask,
Response
import
boto3
from botocore.exceptions import

NoCredentialsErrorapp = Flask(

name)

# Set your AWS S3 credentials
S3_BUCKET_NAME =
'your-s3-bucket-
name'
AWS_ACCESS_KEY =
'your-aws-access-key'
AWS_SECRET_KEY =
'your-aws-secret-key'

# Initialize AWS S3 client
s3 = boto3.client('s3', aws_access_key_id=AWS_ACCESS_KEY,
aws_secret_access_key=AWS_SECRET_KEY)

# Function to
stream video
from S3def
stream_vide
```

```
o(video_key)
:
    try:
        response =
        s3.get_object(Bucket=S3_BUCKET_NAME,
        Key=video_key)data = response['Body'].read()
        return data
    except NoCredentialsError:
        return "AWS credentials not available"

# Route for streaming video
@app.route('/str
eam/<video_key
>')def
stream(video_k
ey):
    video_data = stream_video(video_key)
    return Response(video_data, content_type="video/mp4")

if
    na
    me_____== '_main_':app.run(debug=True)
```

Support process :

tep 1: Project Setup

Initializes NPM: Create and Locate your project folder into the terminal & type the command

npm init -y

It initializes our node application & makes a package.json file.

Install Dependencies: Locate your root project directory into the terminal and type the command

npm install express

To install Express as dependencies inside your project

Create Server File: Create an ‘app.js’ file, inside this file require the Express Module and File System Module, and create a constant ‘app’ for creating an instance of the express module.

```
const express = require('express');

const fs = require('fs');

const app = express();
```

Step 2: Setup Home Page

Create Index.html file: Let’s create a simple HTML file consists a single line inside the body tag which is basically used to set the video path and width of the video. We will set the video path to the ‘videoplayer’ and later in step 3, we will set up this ‘videoplayer’ request inside our server file(app.js) to stream the video.

index.html

```
<!DOCTYPE html>

<html lang="en">

<head>

   <title>Video Player</title>

</head>

<body>
```

```
    <video src="/videoplayer" width="1080px"
        controls></video>
</body>
</html>
```

Send the file to the home page: Now create a get request for the home page which just send this index.html to the browser.

```
app.get('/', (req, res) => {
  res.sendFile(__dirname + '/index.html');
})
```

Step 3: Setup videoplayer route: Now create a get request for the videoplayer.

```
app.get('/videoplayer', (req, res) => {

})
```

Inside this route follow the below steps.

Create a const range: This contains the range value that we get from the header.

```
const range = req.headers.range
```

Set Video Path: Here we have to create a const videoPath and set it to the location of the video.

```
const videoPath = pathOfVideo
```

replace pathOfVideo with the location of your video file.

Set Video Size: Now we have to use the fs module statSync method and pass the video path then calculate the size.

```
const videoSize  = fs.statSync(videoPath).size
```

Set Chunk Size: Use just need to use the below code to set chunk size for streaming in Node.js.

```
const chunkSize = 1 * 1e6;
```

Create const start and end: They are used to calculate the starting and end position of the video

```
const start = Number(range.replace(/\D/g, ""))
const end = Math.min(start + chunkSize, videoSize - 1)
```

Calculate the content length and set it to a variable:

```
const contentLength = end - start + 1;
```

Set header for playing video:

```
const headers = {
   "Content-Range" : `bytes ${start}-${end}/${videoSize}`,
   "Accept-Ranges" : "bytes",
   "Content-Length" : contentLength,
   "Content-Type" : "video/mp4"
}

res.writeHead(206, headers)
```

Create Video Stream and Pipe it with the response:

```
const stream = fs.createReadStream(videoPath, {start, end})

stream.pipe(res)
```

Complete Code: Below is the complete code to build Note Taking Application using Node.js:

```
const express = require('express');
const fs = require('fs');
const app = express();

app.get('/', (req, res) => {
   res.sendFile(__dirname + '/index.html');
})

app.get('/videoplayer', (req, res) => {
   const range = req.headers.range
   const videoPath = './video.mp4';
   const videoSize = fs.statSync(videoPath).size
   const chunkSize = 1 * 1e6;
   const start = Number(range.replace(/\D/g, ""))
   const end = Math.min(start + chunkSize, videoSize - 1)
   const contentLength = end - start + 1;
   const headers = {
      "Content-Range": `bytes ${start}-${end}/${videoSize}`,
      "Accept-Ranges": "bytes",
      "Content-Length": contentLength,
```

```
        "Content-Type": "video/mp4"
    }
    res.writeHead(206, headers)
    const stream = fs.createReadStream(videoPath, {
        start,
        end
    })
    stream.pipe(res)
})
app.listen(3000);
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Video Player</title>
</head>
<body>
    <video src="/videoplayer" width="1080px"
        controls></video>
</body>
</html>
```

THANK YOU