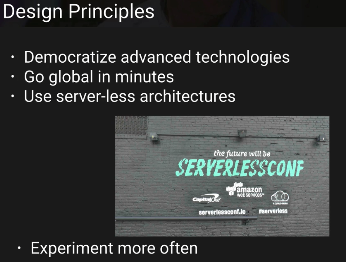
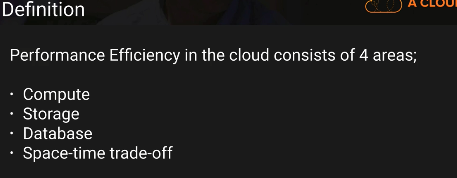


Democratize advanced technologies – Making use of the cloud services instead of learning and designing solutions.. make use of the cloud provider features..



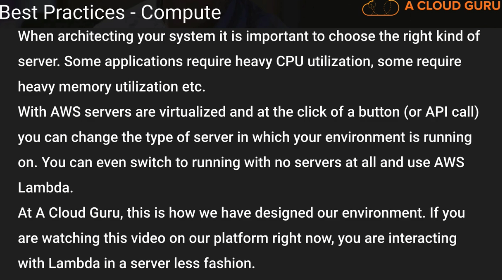


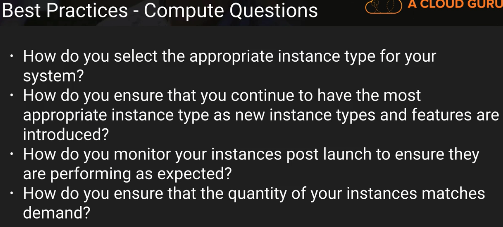
Lambda –

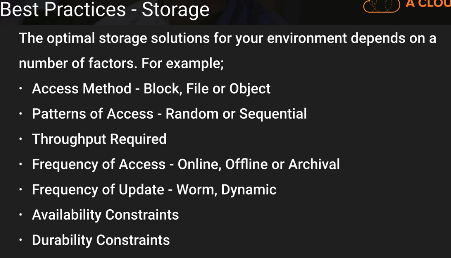
only pay for the execution cost..

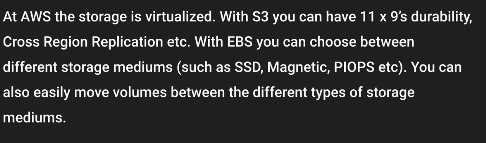
no need to pay for the servers..

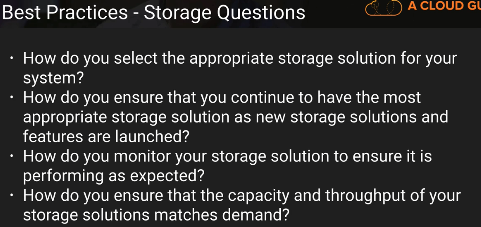
no operational cost…

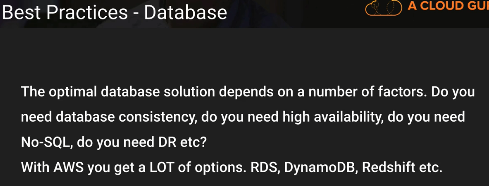




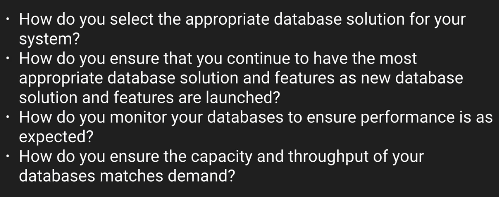


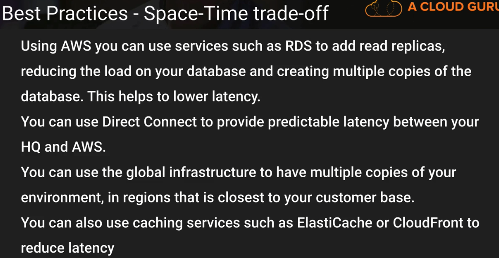


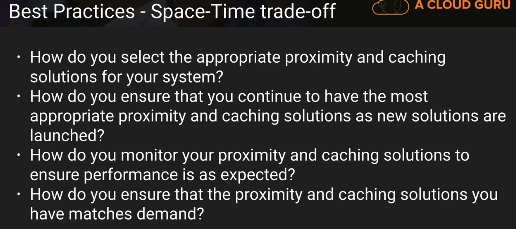


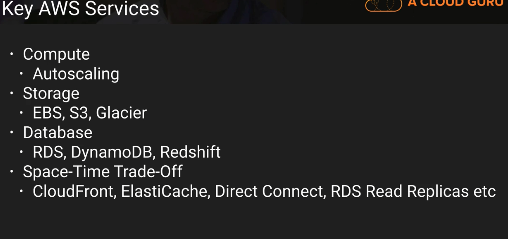


Database questions ????









The performance efficiency pillar focuses on the efficient use of computing resources to meet requirements and how to maintain that efficiency as demand changes and technologies evolve

**Design Principles**

Democratize advanced technologies

Use cloud for new technology, no need to spend time learning and setting up.

m focus on product development rather than resource provisioning and management

Go global in minutes

Easily deploy your system in multiple AWS Regions around the world with just a few click

Use serverless architectures

In the cloud, serverless architectures remove the need for you to run and maintain servers to carry out traditional compute activities

Example : Storage service acts as static website

No need to manage servers

Experiment more often

With virtual and automatable resources, you can quickly carry out comparative testing using different types of instances, storage, or configurations

Mechanical sympathy

**Definition**

Performance efficiency in the cloud is composed of four areas: • Selection • Review • Monitoring • Trade-offs

**Selection**

Cloud v/s on-premises

In the following sections, we look at the four main resource types that you should consider: compute, storage, database, and network.

**Compute**

In AWS, compute is available in three forms: instances, containers, and functions

EC2

ECS

LAMBDA

Elasticity Elasticity allows you to match the supply of resources you have against demand for them. Instances, containers, and functions all provide mechanisms for elasticity either in combination with Auto Scaling or as a feature of the service itself.

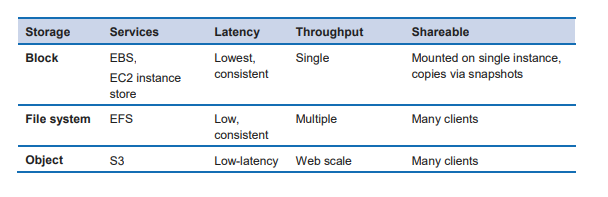
Key AWS Services The key AWS service for elastic compute solutions is Auto Scaling because you use it to match the supply of your resources against demand for them.

**Storage**

Characteristics

When you select a storage solution, you should consider the different characteristics that you require, such as ability to share, file size, cache size, latency, throughput, and persistence of data. Then match the requirements you want to the AWS service that best fits your needs: Amazon S3, Amazon Glacier, Amazon Elastic Block Store (Amazon EBS), Amazon Elastic File System (Amazon EFS), or Amazon EC2 instance store.

Performance can be measured by looking at throughput, input/output operations per second (IOPS), and latency



cid:image002.png@01D412E8.58F6F800

**Key AWS Services – S3**

EBS, EFS, EC2 instance store, Glacier

Configuration Options

SSD-backed storage for transactional workloads – DB and boot volumes

HDD-backed storage for throughput-intensive workloads such as MapReduce and log processing

S3

S3 + cloudfront - route data over an optimized network path

Access Patterns

Creating a RAID 0 (zero) array allows you to achieve a higher level of performance for a file system than you can provision on a single volume. Consider using RAID 0 when I/O performance is more important than fault tolerance. For example, you could use it with a heavily used database where data replication is already set up separately

**Database**

Efficiency Factors

factors to consider when selecting your database approach

When selecting your database technology, keep in mind these four factors: • Access patterns • Characteristics • Configuration options • Operational effort

Access Patterns

The way that you access data will affect how the database solution performs.

Characteristics

Consider the different characteristics you require (for example, availability, consistency, partition tolerance, latency, durability, scalability, and query capability) so that you can select the appropriate approach to use (for example, relational, NoSQL, warehouse).

Configuration

Options Database solutions generally have configuration options that allow you to optimize for the type of workload. Consider the configuration options for your selected database approach such as storage optimization, database level settings, memory, and cache.

Operational Effort

You should consider the effort required to operate your technology choice

Database Technologies

We commonly see four technology approaches used to address database requirements:

• Relational Online Transaction Processing (OLTP)

• Non-relational databases (NoSQL)

• Data warehouse and Online Analytical Processing (OLAP)

• Data indexing and searching

**Data Indexing and Searching**

Technologies such as search engines are particularly efficient for these use cases. Use search engines when you want search or report with low latency and high throughput.

Apache Lucene is a search engine platform

Elasticsearch provides an easy-to-use platform that can automatically discover and index documents at a really large scale

**Network**

AWS offers product features (for example, enhanced networking instance types, Amazon EBS optimized instances, Amazon S3 transfer acceleration, and dynamic CloudFront) to optimize network traffic.

AWS also offers networking features (for example, Amazon Route 53 latency routing, Amazon Virtual Private Cloud (Amazon VPC) endpoints, and AWS Direct Connect) to reduce network distance or jitter.

Location

Choose the appropriate Region or Regions for your deployment based on some key elements: • Where your users are located: Choosing a Region close to your application’s users ensures lower latency when they use the application. • Where your data is located: For data-heavy applications, the major bottleneck in latency is when data is transferred to the computing part of the application. Application code should execute as close to the data as possible.

Placement Groups

Amazon EC2 provides placement groups for networking. A placement group is a logical grouping of instances within a single Availability Zon

Placement groups are recommended for applications that benefit from low network latency, high network throughput, or both

Edge Locations

Delivery Network (CDN) and Domain Name System (DNS)

Product Features

You should consider product features such as

EC2 instance network capability,

enhanced networking instance types,

Amazon EBS-optimized instances,

**Amazon S3 transfer acceleration**, - feature that lets external users benefit from the networking optimizations of CloudFront to upload data to Amazon S3. This makes it easy to transfer large amounts of data from remote locations that don’t have dedicated connectivity to the AWS Cloud

dynamic CloudFront to optimize network traffic

Networking Features

Latency-based routing (LBR) for Route 53

AWS Direct Connect provides dedicated connectivity to the AWS environment, from 50 Mbps up to 10 Gbps

Amazon VPC endpoints provide reliable connectivity to AWS services

Key services

VPC

Route 53

Direct Connect

**Review**

To adopt a data-driven approach to architecture you should implement a performance review process that considerers the following:

Infrastructure as code - Define your infrastructure as code using approaches such as AWS CloudFormation templates

Deployment pipeline - Use a continuous integration/continuous deployment (CI/CD) pipeline to deploy your infrastructure.

Well-defined metrics: Set up your metrics and monitoring to capture key performance indicators (KPIs).

Performance test automatically - As part of your deployment process, automatically trigger performance tests after the quicker running tests have passed successfully. The automation should create a new environment, set up initial conditions such as test data, and then execute a series of benchmarks and load tests.

Alternatively, you could execute performance tests overnight using Amazon EC2 Spot Instances

Load generation

You should create a series of test scripts that replicate synthetic or prerecorded user journeys

As much as possible, you want your test scripts to replicate the behavior of usage in production

Performance visibility:

Key metrics should be visible to your team, especially metrics against each build version.

Visualization:

Use visualization techniques that make it clear where performance issues, hot spots, wait states, or low utilization is occurring.

**Benchmarking**

Benchmarking uses synthetic tests to provide you with data on how components perform

Benchmarking is generally quicker to set up than load testing and is used when you want to evaluate the technology for a particular component.

Benchmarking is often used at the start of a new project, when you don’t yet have a whole solution that you could load test.

Key AWS Services The key AWS services supporting benchmarking are **AWS CodeDeploy and AWS CloudFormation.**

**Load Testing**

Load Testing Load testing uses your actual workload so you can see how your whole solution performs in a production environment.

Load tests should be done using synthetic or sanitized versions of production data (remove sensitive or identifying information).

Key AWS Services The key AWS service supporting load testing is CloudWatch, which allows you to collect metrics on how your whole architecture is performing during the load test.

**Monitoring**

After you have implemented your architecture you will need to monitor its performance so that you can remediate any issues before your customers are aware of them

CloudWatch provides the ability to monitor and send notification alarms. You can use automation to work around performance issues by triggering actions through Amazon Kinesis, Amazon Simple Queue Service (Amazon SQS), and AWS Lambd

Active and Passive

Active monitoring simulates user activity in scripted user journeys across critical paths in your product. AM should be continuously performed in order to test the performance and availability of a workload

Passive monitoring is commonly used with web-based workloads. PM collects performance metrics from the browser

You should use PM to understand these issues:

• **User experience performance**: PM provides you with metrics on what your users are experiencing, which gives you a continuous view into how production is working, as well as a view into the impact of changes over time.

• **Geographic performance variability**: If a workload has a global footprint and users access the application from all around the world, using PM can enable you to spot a performance problem affecting users in a specific geography.

• **The impact of API use**: Modern workloads use internal APIs and third-party APIs. PM provides the visibility into the use of APIs so you can identify performance bottlenecks that originate not only from internal APIs but also from third-party API providers.

Phases

Phases Monitoring at AWS consists of five distinct phases, which are explained in more detail in the

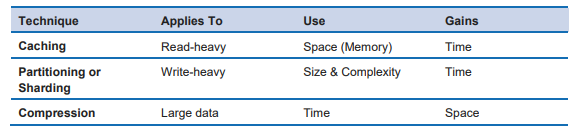
Generation – scope of monitoring, metrics, and thresholds 2. Aggregation – creating a complete view from multiple source 3. Real-time processing and alarming – recognizing and responding 4. Storage – data management and retention policies 5. Analytics – dashboards, reporting, and insights

**Trade-offs**

Depending on your situation you could trade consistency, durability, and space versus time or latency, to deliver higher performance.

Depending on your situation you could trade consistency, durability, and space versus time or latency, to deliver higher performance.

The following sections detail some of the trade-offs you can make and how you can implement them.



cid:image004.png@01D412F7.11A6E780

**Caching**

Most workloads rely on a dependent component such as a service or database that offers a source of truth or a consolidated view of data. Generally, these architecture components are harder to scale and represent a significant proportion of the cost of the workload. You can improve performance efficiency by using caching to trade off against freshness or memory used. These techniques generally update asynchronously or periodically. The trade-off is that your data isn’t always fresh and, therefore, not always consistent with the source of truth.

Application Level

You can make this trade-off at a code level by using application-level caches or memorization

**user sessions** in an application

Database Level

Database replicas

Geographic Level

CloudFront- cdn

Key Services

**ElastiCache,** which provides a general-purpose application cache

**CloudFront,** which allows you to cache information closer to your users.

**Partitioning or Sharding**

Db - rdb

When you hit the limits of vertical scaling, you can use a different approach called data partitioning or sharding. With this model, data is split across multiple database schemas, each running in its own autonomous primary DB instance.

Key AWS Services The key AWS service for partitioning or sharding is Amazon DynamoDB, which manages table partitioning for you automatically

**Compression**

Compressing data trades computing time against space and can greatly reduce storage and networking requirements.

CloudFront supports compression at the edge

The key AWS service for compression is CloudFront, which supports compression at the edge.

**Buffering**

Buffering uses a queue to accept messages (units of work) from producers.

A buffer is a mechanism to ensure that applications can communicate with each other when they are running at different rates over tim

By using a buffer, you can decouple the throughput rate of producers from that of consumers

**Amazon SQS** provides a queue that allows a single consumer to read individual messages. **Amazon Kinesis** provides a stream that allows many consumers to read the same messages.

The key AWS service for buffering is Amazon SQS, which allows you to decouple producers from consumers using a queue.