Chapter 10
Amazon ElastiCache

THE AWS CERTIFIED SOLUTIONS ARCHITECT ASSOCIATE EXAM OBJECTIVES COVERED IN THIS CHAPTER MAY INCLUDE, BUT ARE NOT LIMITED TO, THE FOLLOWING:

Domain 1.0: Designing highly available, cost-efficient, fault-tolerant, and scalable systems

✓ **Identify and recognize cloud architecture considerations, such as fundamental components and effective designs.**

Content may include the following:

- Planning and design
- Architectural trade-off decisions
- Best practices for AWS architecture
- Elasticity and scalability

Domain 3.0: Data Security

✓ **3.1 Recognize and implement secure practices for optimum cloud deployment and maintenance.**

Content may include the following:

- AWS administration and security services

✓ **3.2 Recognize critical disaster recovery techniques and their implementation.**

Introduction

This chapter focuses on building high-performance applications using in-memory caching technologies and Amazon ElastiCache. By using the Amazon ElastiCache service, you can offload the heavy lifting involved in the deployment and operation of cache environments running Memcached or Redis. It focuses on key topics you need to understand for the exam, including:

- How to improve application performance using caching
- How to launch cache environments in the cloud
- What are the basic differences and use cases for Memcached and Redis?
- How to scale your cluster vertically
- How to scale your Memcached cluster horizontally using additional cache nodes
- How to scale your Redis cluster horizontally using replication groups
- How to back up and recover your Redis cluster
- How to apply a layered security model

In-Memory Caching

One of the common characteristics of a successful application is a fast and responsive user experience. Research has shown that users will get frustrated and leave a website or app when it is slow to respond. In 2007, testing of Amazon.com's retail site showed that for every 100ms increase in load times, sales decreased by 1%. Round-trips back and forth to a database and its underlying storage can add significant delays and are often the top contributor to application latency.

Caching frequently-used data is one of the most important performance optimizations you can make in your applications. Compared to retrieving data from an in-memory cache, querying a database is an expensive operation. By storing or moving frequently accessed data in-memory, application developers can significantly improve the performance and responsiveness of read-heavy applications. For example, the application session state for a large website can be stored in an in-memory caching engine, instead of storing the session data in the database.

For many years, developers have been building applications that use cache engines like Memcached or Redis to store data in-memory to get blazing fast application performance. Memcached is a simple-to-use in-memory key/value store that can be used to store arbitrary types of data. It is one of the most popular cache engines. Redis is a flexible in-memory data structure store that can be used as a cache, database, or even as a message broker. Amazon ElastiCache allows developers to easily deploy and manage cache environments running either Memcached or Redis.

Amazon ElastiCache

Amazon ElastiCache is a web service that simplifies the setup and management of distributed in-memory caching environments. This service makes it easy and cost effective to provide a high-performance and scalable caching solution for your cloud applications. You can use Amazon ElastiCache in your applications to speed the deployment of cache clusters and reduce the administration required for a distributed cache environment.

With Amazon ElastiCache, you can choose from a Memcached or Redis protocol-compliant cache engine and quickly launch a cluster within minutes. Because Amazon ElastiCache is a managed service, you can start using the service today with very few or no modifications to your existing applications that use Memcached or Redis. Because Amazon ElastiCache is protocol-compliant with both of these engines, you only need to change the endpoint in your configuration files.

Using Amazon ElastiCache, you can implement any number of caching patterns. The most common pattern is the cache-aside pattern depicted in Figure 10.1. In this scenario, the app server checks the cache first to see if it contains the data it needs. If the data does not exist in the cache node, it will query the database and serialize and write the query results to the cache. The next user request will then be able to read the data directly from the cache instead of querying the database.

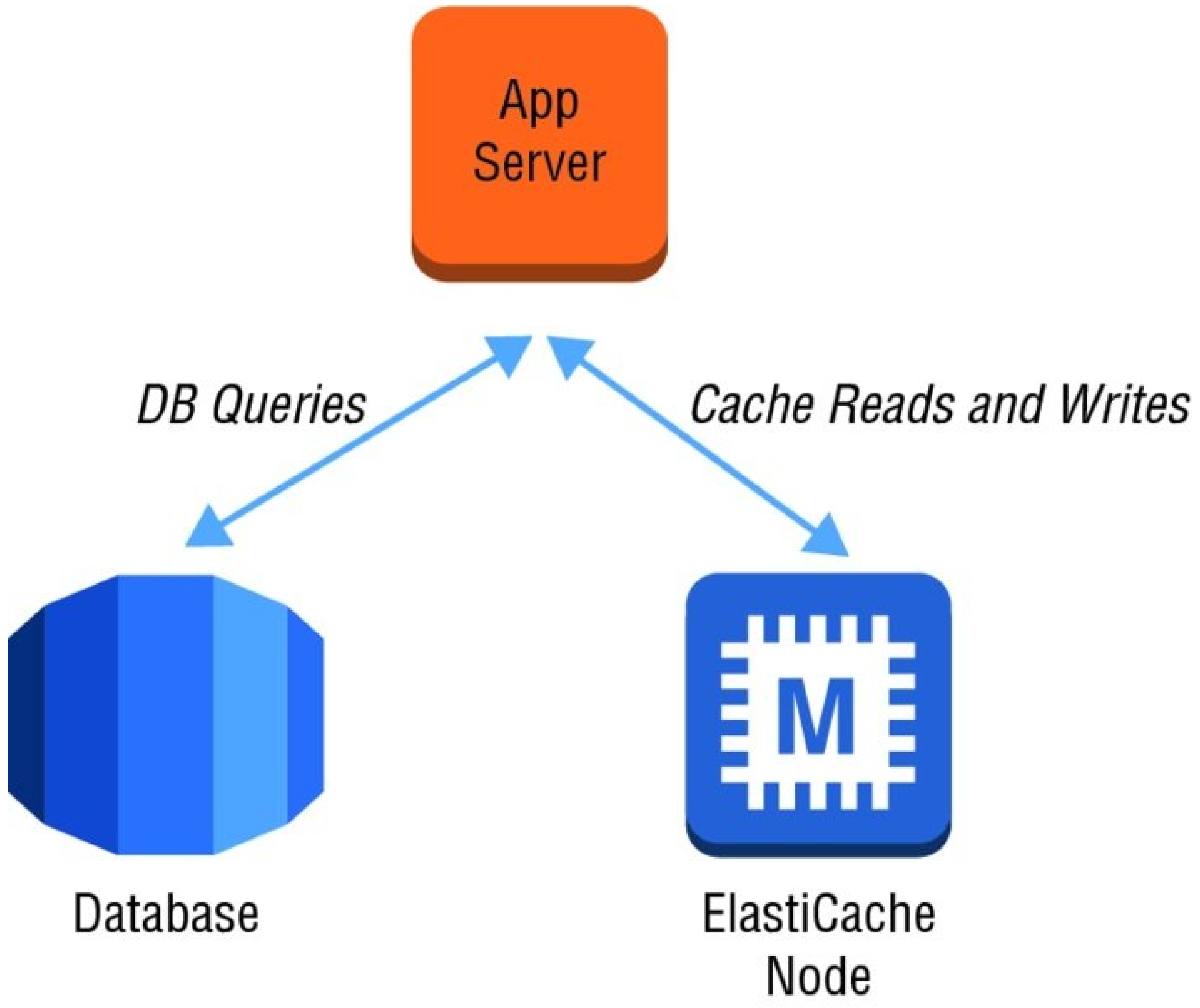

FIGURE 10.1 Common caching architecture

While it is certainly possible to build and manage a cache cluster yourself on Amazon Elastic Compute Cloud (Amazon EC2), Amazon ElastiCache allows you to offload the heavy lifting of installation, patch management, and monitoring to AWS so you can focus on your application instead. Amazon ElastiCache also provides a number of features to enhance the reliability of critical deployments. While it is rare, the underlying Amazon EC2 instances can become impaired. Amazon ElastiCache can automatically detect and recover from the failure of a cache node. With the Redis engine, Amazon ElastiCache makes it easy to set up read replicas and fail over from the primary to a replica in the event of a problem.

Data Access Patterns

Retrieving a flat key from an in-memory cache will always be faster than the most optimized database query. You should evaluate the access pattern of the data before you decide to store it in cache. A good example of something to cache is the list of products in a catalog. For a busy website, the list of items could be retrieved thousands of times per second. While it makes sense to cache the most heavily requested items, you can also benefit from caching items that are not frequently requested.

There are also some data items that should not be cached. For example, if you generate a unique page every request, you probably should not cache the page results. However, even though the page changes every time, it does make sense to cache the components of the page that do not change.

Cache Engines

Amazon ElastiCache allows you to quickly deploy clusters of two different types of popular cache engines: Memcached and Redis. At a high level, Memcached and Redis may seem similar, but they support a variety of different use cases and provide different functionality.

Memcached Memcached provides a very simple interface that allows you to write and read objects into in-memory key/value data stores. With Amazon ElastiCache, you can elastically grow and shrink a cluster of Memcached nodes to meet your demands. You can partition your cluster into shards and support parallelized operations for very high performance throughput. Memcached deals with objects as blobs that can be retrieved using a unique key. What you put into the object is up to you, and it is typically the serialized results from a database query. This could be simple string values or binary data.

Amazon ElastiCache supports a number of recent versions of Memcached. As of early 2016, the service supports Memcached version 1.4.24, and also older versions going back to 1.4.5. When a new version of Memcached is released, Amazon ElastiCache simplifies the upgrade process by allowing you to spin up a new cluster with the latest version.

Redis In late 2013, Amazon ElastiCache added support to deploy Redis clusters. At the time of this writing, the service supports the deployment of Redis version 2.8.24, and also a number of older versions. Beyond the object support provided in Memcached, Redis supports a rich set of data types likes strings, lists, and sets.

Unlike Memcached, Redis supports the ability to persist the in-memory data onto disk. This allows you to create snapshots that back up your data and then recover or replicate from the backups. Redis clusters also can support up to five read replicas to offload read requests. In the event of failure of the primary node, a read replica can be promoted and become the new master using Multi-AZ replication groups.

Redis also has advanced features that make it easy to sort and rank data. Some common use cases include building a leaderboard for a mobile application or serving as a high-speed message broker in a distributed system. With a Redis cluster, you can leverage a publish and subscribe messaging abstraction that allows you to decouple the components of your applications. A publish and subscribe messaging architecture gives you the flexibility to change how you consume the messages in the future without affecting the component that is producing the messages in the first place.

Nodes and Clusters

Each deployment of Amazon ElastiCache consists of one or more *nodes* in a *cluster*. There are many different types of nodes available to choose from based on your use case and the necessary resources. A single Memcached cluster can contain up to 20 nodes. Redis clusters are always made up of a single node; however, multiple clusters can be grouped into a *Redis replication group*.

The individual node types are derived from a subset of the Amazon EC2 instance type families, like t2, m3, and r3. The specific node types may change over time, but today they range from a t2.micro node type with 555MB of memory up to an r3.8xlarge with 237GB of memory, with many choices in between. The t2 cache node family is ideal for development and low-volume applications with occasional bursts, but certain features may not be available. The m3 family is a good blend of compute and memory, while the r3 family is optimized for memory-intensive workloads.

Depending on your needs, you may choose to have a few large nodes or many smaller nodes in your cluster or replication group. As demand for your application changes, you may also add or remove nodes from time to time. Each node type comes with a preconfigured amount of memory, with a small amount of the memory allocated to the caching engine and operating system itself.

Design for Failure

While it is unlikely, you should plan for the potential failure of an individual cache node. For Memcached clusters, you can decrease the impact of the failure of a cache node by using a larger number of nodes with a smaller capacity, instead of a few large nodes.

In the event that Amazon ElastiCache detects the failure of a node, it will provision a replacement and add it back to the cluster. During this time, your database will experience increased load, because any requests that would have been cached will now need to be read from the database. For Redis clusters, Amazon ElastiCache will detect failure and replace the primary node. If a Multi-AZ replication group is enabled, a read replica can be automatically promoted to primary.

Memcached Auto Discovery

For Memcached clusters partitioned across multiple nodes, Amazon ElastiCache supports *Auto Discovery* with the provided client library. Auto Discovery simplifies your application code by no longer needing awareness of the infrastructure topology of the cache cluster in your application layer.

Using Auto Discovery

The Auto Discovery client gives your applications the ability to identify automatically all of the nodes in a cache cluster and to initiate and maintain connections to all of these nodes. The Auto Discovery client is available for .NET, Java, and PHP platforms.

Scaling

Amazon ElastiCache allows you to adjust the size of your environment to meet the needs of workloads as they evolve over time. Adding additional cache nodes allows you to easily expand horizontally and meet higher levels of read or write performance. You can also select different classes of cache nodes to scale vertically.

Horizontal Scaling Amazon ElastiCache also adds additional functionality that allows you to scale horizontally the size of your cache environment. This functionality differs depending on the cache engine you have selected. With Memcached, you can partition your data and scale horizontally to 20 nodes or more. With Auto Discovery, your application can discover Memcached nodes that are added or removed from a cluster.

A Redis cluster consists of a single cache node that is handling read and write transactions. Additional clusters can be created and grouped into a Redis replication group. While you can only have one node handling write commands, you can have up to five read replicas handling read-only requests.

Vertical Scaling Support for vertical scaling is more limited with Amazon ElastiCache. If you like to change the cache node type and scale the compute resources vertically, the service does not directly allow you to resize your cluster in this manner. You can, however, quickly spin up a new cluster with the desired cache node types and start redirecting traffic to the new cluster. It's important to understand that a new Memcached cluster always starts empty, while a Redis cluster can be initialized from a backup.

Replication and Multi-AZ

Replication is a useful technique to provide rapid recovery in the event of a node failure, and also to serve up very high volumes of read queries beyond the capabilities of a single node. Amazon ElastiCache clusters running Redis support both of these design requirements. Unlike Redis, cache clusters running Memcached are standalone in-memory services without any redundant data protection services.

Cache clusters running Redis support the concept of *replication groups*. A replication group consists of up to six clusters, with five of them designated as read replicas. This allows you to scale horizontally by writing code in your application to offload reads to one of the five clones (see Figure 10.2).

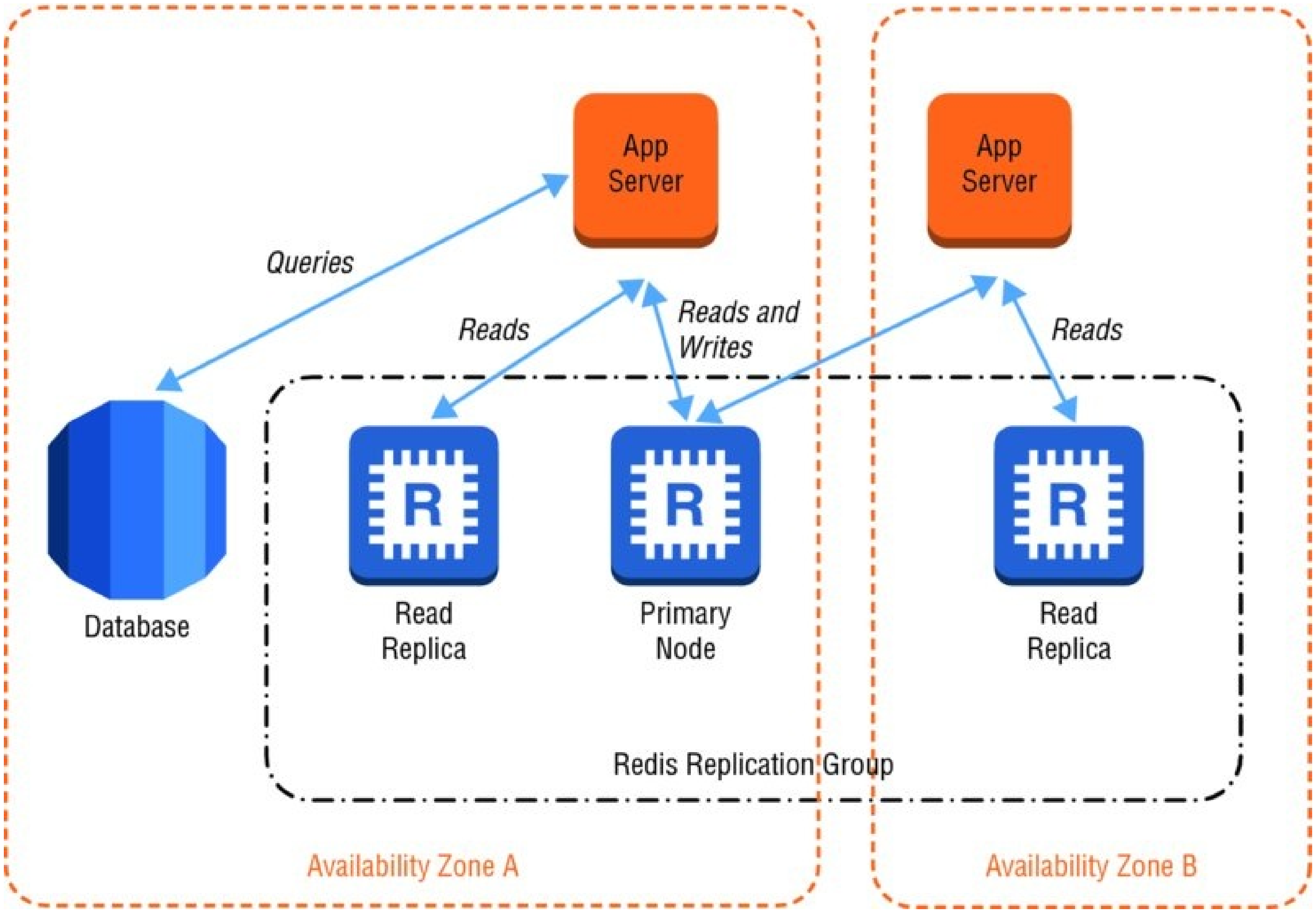

FIGURE 10.2 Redis replication group

Multi-AZ Replication Groups

You can also create a *Multi-AZ replication group* that allows you to increase availability and minimize the loss of data. Multi-AZ simplifies the process of dealing with a failure by automating the replacement and failover from the primary node.

In the event the primary node fails or can't be reached, Multi-AZ will select and promote a read replica to become the new primary, and a new node will be provisioned to replace the failed one. Amazon ElastiCache will then update the Domain Name System (DNS) entry of the new primary node to allow your application to continue processing without any configuration change and with only a short disruption.

Understand That Replication Is Asynchronous

It's important to keep in mind that replication between the clusters is performed asynchronously and there will be a small delay before data is available on all cluster nodes.

Backup and Recovery

Amazon ElastiCache clusters running Redis allow you to persist your data from in-memory to disk and create a *snapshot*. Each snapshot is a full clone of the data that can be used to recover to a specific point in time or to create a copy for other purposes. Snapshots cannot be created for clusters using the Memcached engine because it is a purely in-memory key/value store and always starts empty. Amazon ElastiCache uses the native backup capabilities of Redis and will generate a standard Redis database backup file that gets stored in Amazon Simple Storage Service (Amazon S3).

Snapshots require compute and memory resources to perform and can potentially have a performance impact on heavily used clusters. Amazon ElastiCache will try different backup techniques depending on the amount of memory currently available. A best practice is to set up a replication group and perform a snapshot against one of the read replicas instead of the primary node.

In addition to manually initiated snapshots, snapshots can be created automatically based on a schedule. You can also configure a window for the snapshot operation to be completed and specify how many days of backups you want to store. Manual snapshots are stored indefinitely until you delete them.

Backup Redis Clusters

Use a combination of automatic and manual snapshots to meet your recovery objectives for your Redis cluster. Memcached is purely in-memory and does not have native backup capabilities.

Whether the snapshot was created automatically or manually, the snapshot can then be used to create a new cluster at any time. By default, the new cluster will have the same configuration as the source cluster, but you can override these settings. You can also restore from an RDB file generated from any other compatible Redis cluster.

Access Control

Access to your Amazon ElastiCache cluster is controlled primarily by restricting inbound network access to your cluster. Inbound network traffic is restricted through the use of security groups. Each security group defines one or more inbound rules that restrict the source traffic. When deployed inside of a Virtual Private Cloud (VPC), each node will be issued a private IP address within one or more subnets that you select. Individual nodes can never be accessed from the Internet or from Amazon EC2 instances outside the VPC. You can further restrict network ingress at the subnet level by modifying the network Access Control Lists (ACLs).

Access to manage the configuration and infrastructure of the cluster is controlled separately from access to the actual Memcached or Redis service endpoint. Using the AWS Identity and Access Management (IAM) service, you can define policies that control which AWS users can manage the Amazon ElastiCache infrastructure itself.

Some of the key actions an administrator can perform include CreateCacheCluster, ModifyCacheCluster, or DeleteCacheCluster. Redis clusters also support CreateReplicationGroup and CreateSnapshot actions, among others.

Summary

In this chapter, you learned about caching environments within the cloud using Amazon ElastiCache. You can quickly launch clusters running Memcached or Redis to store frequently used data in-memory. Caching can speed up the response time of your applications, reduce load on your back-end data stores, and improve the user experience.

With Amazon ElastiCache, you can offload the administrative tasks for provisioning and operating clusters and focus on the application. Each cache *cluster* contains one or more *nodes*. Select from a range of *node types* to give the right mix of compute and memory resources for your use case.

You can expand both Memcached and Redis clusters vertically by selecting a larger or smaller node type to match your needs. With Amazon ElastiCache and the Memcached engine, you can also scale your cluster horizontally by adding or removing nodes. With Amazon ElastiCache and the Redis engine, you can also scale horizontally by creating a *replication group* that will automatically replicate across multiple read replicas.

Streamline your backup and recovery process for Redis clusters with Amazon ElastiCache's consistent operational model. While Memcached clusters are in-memory only and cannot be persisted, Redis clusters support both automated and manual *snapshots*. A snapshot can then be restored to recover from a failure or to clone an environment.

You can secure your cache environments at the network level with security groups and network ACLs, and at the infrastructure level using IAM policies. Security groups will serve as your primary access control mechanism to restrict inbound access for active clusters.

You should analyze your data usage patterns and identify frequently run queries or other expensive operations that could be candidates for caching. You can relieve pressure from your database by offloading read requests to the cache tier. Data elements that are accessed on every page load, or with every request but do not change, are often prime candidates for caching. Even data that changes frequently can often benefit from being cached with very large request volumes.

Exam Essentials

Know how to use Amazon ElastiCache. Improve the performance of your application by deploying Amazon ElastiCache clusters as part of your application and offloading read requests for frequently accessed data. Use the cache-aside pattern in your application first to check the cache for your query results before checking the database.

Understand when to use a specific cache engine. Amazon ElastiCache gives you the choice of cache engine to suit your requirements. Use Memcached when you need a simple, in-memory object store that can be easily partitioned and scaled horizontally. Use Redis when you need to back up and restore your data, need many clones or read replicas, or are looking for advanced functionality like sort and rank or leaderboards that Redis natively supports.

Understand how to scale a Redis cluster horizontally. An Amazon ElastiCache cluster running Redis can be scaled horizontally first by creating a replication group, then by creating additional clusters and adding them to the replication group.

Understand how to scale a Memcached cluster horizontally. An Amazon ElastiCache cluster running Memcached can be scaled horizontally by adding or removing additional cache nodes to the cluster. The Amazon ElastiCache client library supports Auto Discovery and can discover new nodes added or removed from the cluster without having to hardcode the list of nodes.

Know how to back up your Amazon ElastiCache cluster. You can create a snapshot to back up your Amazon ElastiCache clusters running the Redis engine. Snapshots can be created automatically on a daily basis or manually on demand. Amazon ElastiCache clusters running Memcached do not support backup and restore natively.

Exercises

In this section, you will create a cache cluster using Amazon ElastiCache, expand the cluster with additional nodes, and finally create a replication group with an Amazon ElastiCache Redis cluster.

EXERCISE 10.1

Create an Amazon ElastiCache Cluster Running Memcached

In this exercise, you will create an Amazon ElastiCache cluster using the Memcached engine.

1. While signed into the AWS Management Console, open the Amazon ElastiCache service dashboard.
2. Begin the launch and configuration process to create a new Amazon ElastiCache cluster.
3. Select the Memcached cache engine, and configure the cluster name, number of nodes, and node type.
4. Optionally configure the security group and maintenance window as needed.
5. Review the cluster configuration, and begin provisioning the cluster.
6. Connect to the cluster with any Memcached client using the DNS name of the cluster.

You have now created your first Amazon ElastiCache cluster.

EXERCISE 10.2

Expand the Size of a Memcached Cluster

In this exercise, you will expand the size of an existing Amazon ElastiCache Memcached cluster.

1. Launch a Memcached cluster using the steps defined in Exercise 10.1.
2. Go to the Amazon ElastiCache dashboard, and view the details of your existing cluster.
3. View the list of nodes currently provisioned, and then add one additional node by increasing the number of nodes.
4. Apply the configuration change, and wait for the new node to finish the provisioning process.
5. Verify that the new node has been created, and connect to the node using a Memcached client.

In this exercise, you have horizontally scaled an existing Amazon ElastiCache cluster by adding a cache node.

EXERCISE 10.3

Create an Amazon ElastiCache Cluster and Redis Replication Group

In this exercise, you will create an Amazon ElastiCache cluster using Redis nodes, create a replication group, and set up a read replica.

1. Sign in to the AWS Management Console, and navigate to the Amazon ElastiCache service dashboard.
2. Begin the configuration and launch process for a new Amazon ElastiCache cluster.
3. Select the Redis cache engine, and then configure a replication group and the node type.
4. Configure a read replica by setting the number of read replicas to 1, and verify that Enable Replication and Multi-AZ are selected.
5. Adjust the Availability Zones for the primary and read replica clusters, security groups, and maintenance window, as needed.
6. Review the cluster configuration, and begin provisioning the cluster.
7. Connect to the primary node and the read replica node with a Redis client library. Perform a simple `set` operation on the primary node, and then perform a `get` operation with the same key on the replica.

You have now created an Amazon ElastiCache cluster using the Redis engine and configured a read replica.

Review Questions

1. Which of the following objects are good candidates to store in a cache? (Choose 3 answers)
    A. Session state
    B. Shopping cart
    C. Product catalog
    D. Bank account balance
2. Which of the following cache engines are supported by Amazon ElastiCache? (Choose 2 answers)
    A. MySQL
    B. Memcached
    C. Redis
    D. Couchbase
3. How many nodes can you add to an Amazon ElastiCache cluster running Memcached?
    A. 1
    B. 5
    C. 20
    D. 100
4. How many nodes can you add to an Amazon ElastiCache cluster running Redis?
    A. 1
    B. 5
    C. 20
    D. 100
5. An application currently uses Memcached to cache frequently used database queries. Which steps are required to migrate the application to use Amazon ElastiCache with minimal changes? (Choose 2 answers)
    A. Recompile the application to use the Amazon ElastiCache libraries.
    B. Update the configuration file with the endpoint for the Amazon ElastiCache cluster.
    C. Configure a security group to allow access from the application servers.
    D. Connect to the Amazon ElastiCache nodes using Secure Shell (SSH) and install the latest version of Memcached.
6. How can you back up data stored in Amazon ElastiCache running Redis? (Choose 2 answers)

A. Create an image of the Amazon Elastic Compute Cloud (Amazon EC2) instance.

B. Configure automatic snapshots to back up the cache environment every night.

C. Create a snapshot manually.

D. Redis clusters cannot be backed up.

7. How can you secure an Amazon ElastiCache cluster? (Choose 3 answers)

   A. Change the Memcached root password.

   B. Restrict Application Programming Interface (API) actions using AWS Identity and Access Management (IAM) policies.

   C. Restrict network access using security groups.

   D. Restrict network access using a network Access Control List (ACL).

8. You are working on a mobile gaming application and are building the leaderboard feature to track the top scores across millions of users. Which AWS services are best suited for this use case?

   A. Amazon Redshift

   B. Amazon ElastiCache using Memcached

   C. Amazon ElastiCache using Redis

   D. Amazon Simple Storage Service (S3)

9. You have built a large web application that uses Amazon ElastiCache using Memcached to store frequent query results. You plan to expand both the web fleet and the cache fleet multiple times over the next year to accommodate increased user traffic. How do you minimize the amount of changes required when a scaling event occurs?

   A. Configure AutoDiscovery on the client side

   B. Configure AutoDiscovery on the server side

   C. Update the configuration file each time a new cluster

   D. Use an Elastic Load Balancer to proxy the requests

10. Which cache engines does Amazon ElastiCache support? (Choose 2 answers)

    A. Memcached

    B. Redis

    C. Membase

    D. Couchbase