Microsoft

AZ-204T00A

Learning Path 12: Implement caching for solutions

Agenda

- Develop for Azure Cache for Redis
- Develop for storage on CDNs

Module 1: Develop for Azure Cache for Redis

Learning objectives

- Explain the key scenarios Azure Cache for Redis covers and its service tiers.
- Identify the key parameters for creating an Azure Cache for Redis instance and interact with the cache.
- Connect an app to Azure Cache for Redis by using .NET Core.

Introduction

- Caching is a common technique that aims to improve the performance and scalability of a system.
- It does this by temporarily copying frequently accessed data to a location close to the application.

Explore Azure Cache for Redis (1 of 2)

- Azure Cache for Redis provides an in-memory data store based on the Redis software.
- Redis improves the performance and scalability of applications that use backend data stores.
- Processes large volumes of application requests by keeping frequently accessed data in the server memory.
- Brings a critical low-latency and high-throughput data storage solution to modern applications.

Explore Azure Cache for Redis (2 of 2)

Key scenarios

Azure Cache for Redis improves application performance by supporting common application architecture patterns.

- Data cache
- Content cache
- Session store
- Job and message queuing
- Distributed transactions

Service tiers

Azure Cache for Redis is available in these tiers:

- Basic
- Standard
- Premium
- Enterprise
- Enterprise Flash

Configure Azure Cache for Redis (1 of 4)

Create and configure an Azure Cache for Redis instance

There are several parameters you will need to decide in order to configure the cache properly for your purposes.

Configure Azure Cache for Redis (2 of 4)

Accessing the Redis instance

- Redis has a command-line tool for interacting with an Azure Cache for Redis as a client.
- Redis supports a set of known commands.

Command	Description	Command	Description
`ping`	Ping the server. Returns "PONG".	`get [key]`	Gets a value from the cache.
`set [key] [value]`	Sets a key/value in the cache. Returns "OK" on success.	`exists [key]`	Returns '1' if the **key** exists in the cache, '0' if it doesn't.
`incr [key]`	Increment the given value associated with **key** by '1'. The value must be an integer or double value. This returns the new value.	`incrby [key] [amount]`	Increment the given value associated with **key** by the specified amount. The value must be an integer or double value. This returns the new value.
`type [key]`	Returns the type associated to the value for the given **key**.	`del [key]`	Deletes the value associated with the **key**.
`flushdb`	Delete all keys and values in the database.		

Configure Azure Cache for Redis (3 of 4)

Adding an expiration time to values

- In Redis we expire values when they are stale by applying a time to live (TTL) to a key.
- When the TTL elapses, the key is automatically deleted, exactly as if the DEL command were issued.
  - Expirations can be set using seconds or milliseconds precision.
  - The expire time resolution is always 1 millisecond.

```
> set counter 100
OK
> expire counter 5
(integer) 1
> get counter
100
... wait ...
> get counter
(nil)
```

Configure Azure Cache for Redis (4 of 4)

Accessing a Redis cache from a client

To connect to an Azure Cache for Redis instance, you'll need the host name, port, and an access key for the cache. You can retrieve this information in the Azure portal.

- The host name is the public Internet address of your cache, which was created using the name of the cache. For example, sportsresults.redis.cache.windows.net.
- The access key acts as a password for your cache. There are two keys created: primary and secondary.

Interact with Azure Cache for Redis by using .NET (1 of 3)

Executing commands on the Redis cache

A popular high-performance Redis client for the .NET language is StackExchange.Redis. The package is available through NuGet and can be added to your .NET code using the command line or IDE.

```
//Creating a connection


using StackExchange.Redis;
...
var connectionString = "[cache-name].redis.cache.windows.net:6380,password=[password-here],ssl=True,abortConnect=False";
var redisConnection = ConnectionMultiplexer.Connect(connectionString);
```

Interact with Azure Cache for Redis by using .NET (2 of 3)

Once you have a `ConnectionMultiplexer`, there are 3 primary things you might want to do:

- Access a Redis Database (example below)
- Make use of the publisher/subscript features of Redis. (Outside the scope of this module.)
- Access an individual server for maintenance or monitoring purposes.

```
// Accessing a database
IDatabase db = redisConnection.GetDatabase();

// Example of storing a key/value in the cache
bool wasSet = db.StringSet("favorite:flavor", "i-love-rocky-road");

// Retrieving the value
string value = db.StringGet("favorite:flavor");
Console.WriteLine(value); // displays: ""i-love-rocky-road""
```

Interact with Azure Cache for Redis by using .NET (3 of 3)

Other common operations

Method	Description
`CreateBatch`	Creates a group of operations that will be sent to the server as a single unit, but not necessarily processed as a unit.
`CreateTransaction`	Creates a group of operations that will be sent to the server as a single unit and processed on the server as a single unit.
`KeyDelete`	Delete the key/value.
`KeyExists`	Returns whether the given key exists in cache.
`KeyExpire`	Sets a time-to-live (TTL) expiration on a key.
`KeyRename`	Renames a key.
`KeyTimeToLive`	Returns the TTL for a key.
`KeyType`	Returns the string representation of the type of the value stored at key. The different types that can be returned are: string, list, set, zset and hash.

Exercise: Connect an app to Azure Cache for Redis by using .NET Core

In this exercise you learn how to create a new Redis Cache instance by using Azure CLI commands and create a .NET Core console app to add and retrieve values from the cache.

Objectives

- Create Azure resources
- Create the console application
- Clean up resources

Summary and knowledge check

In this module, you learned how to:

- Explain the key scenarios Azure Cache for Redis covers and its service tiers
- Identify the key parameters for creating an Azure Cache for Redis instance and interact with the cache
- Connect an app to Azure Cache for Redis by using .NET Core

1. What is the lowest service tier of Azure Cache for Redis recommended for use in production scenarios?

2. What is the expire time resolution when applying a time to live (TTL) to a key in Redis?

Module 2: Develop for storage on CDNs

Learning objectives

- Explain how the Azure Content Delivery Network works and how it can improve the user experience.
- Control caching behavior and purge content.
- Perform actions on Azure CDN by using the Azure CDN Library for .NET.

Introduction

- A content delivery network (CDN) is a distributed network of servers that can efficiently deliver web content to users.
- CDNs store cached content on edge servers in point-of-presence (POP) locations that are close to end users, to minimize latency.

Explore Azure Content Delivery Networks (1 of 4)

Overview

- Azure Content Delivery Network (CDN) delivers high-bandwidth content to users by caching content at strategically placed physical nodes across the world.
- Azure CDN can also accelerate dynamic content, which cannot be cached, by leveraging various network optimizations using CDN POPs.

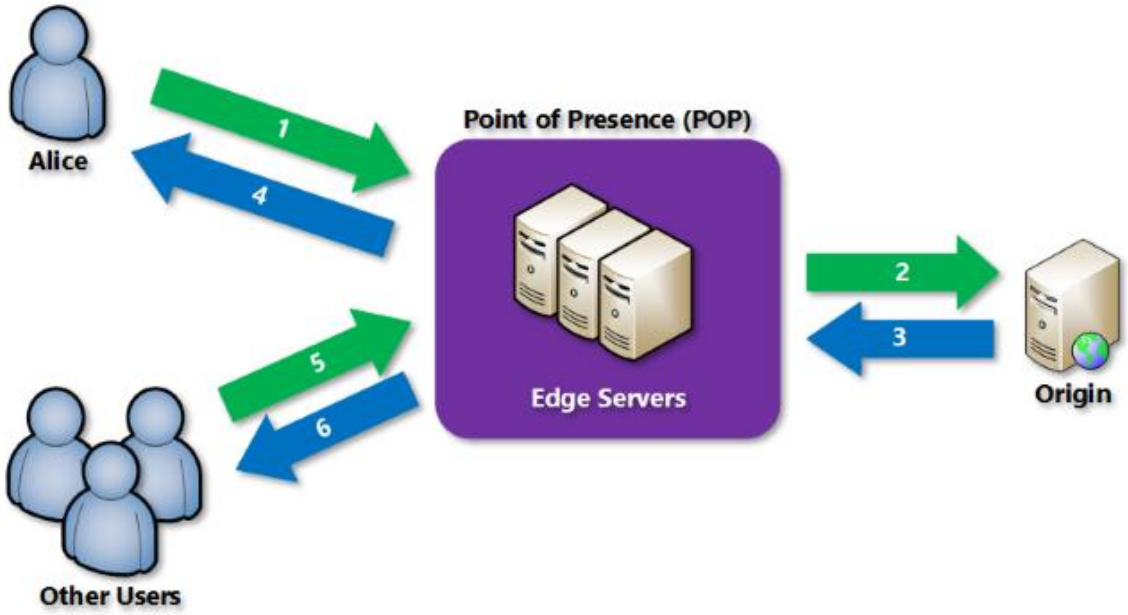
Benefits

- Better performance and improved user experience for end users.
- Large scaling to better handle instantaneous high loads.
- Distribution of user requests and serving of content directly from edge servers so that less traffic is sent to the origin server.

Explore Azure Content Delivery Networks (2 of 4)

How Azure Content Delivery Network works

1. Alice requests a file by using a URL with a special domain name, such as `<endpoint name>.azureedge.net`.
2. If no edge servers in the POP have the file in their cache, the POP requests the file from the origin server.
3. The origin server returns the file to an edge server in the POP.
4. An edge server in the POP caches the file and returns the file to the original requestor (Alice).
5. Additional users can then request the same file by using the same URL that Alice used.
6. If the TTL for the file hasn't expired, the POP edge server returns the file directly from the cache.



Explore Azure Content Delivery Networks (3 of 4)

Requirements

- You need to create at least one CDN profile, which is a collection of CDN endpoints.
- Every CDN endpoint represents a specific configuration of content deliver behavior and access.
- To organize your CDN endpoints by internet domain, web application, or some other criteria, you can use multiple profiles.

Limitations

Each Azure subscription has default limits for the following resources:

- The number of CDN profiles that can be created.
- The number of endpoints that can be created in a CDN profile.
- The number of custom domains that can be mapped to an endpoint.

Explore Azure Content Delivery Networks (4 of 4)

Azure Content Delivery Network (CDN) includes three products:

- Azure CDN Standard from Microsoft
- Azure CDN Standard from Edgio (formerly Verizon)
- Azure CDN Premium from Edgio (formerly Verizon)

Control cache behavior on Azure Content Delivery Networks (1 of 4)

Overview

- Because a cached resource can potentially be out-of-date, it is important for any caching mechanism to control when content is refreshed.
- To save time and bandwidth consumption, a cached resource is not compared to the version on the origin server every time it is accessed.
- If a cached resource is considered to be fresh, it is assumed to be the most current version and is sent directly to the client.
- A cached resource is considered to be fresh when its age is less than the age or period defined by a cache setting.

Control cache behavior on Azure Content Delivery Networks (2 of 4)

Controlling caching behavior

- Azure CDNs provide two mechanisms for caching files. However, these configuration settings depend on the tier you've selected.
- Caching rules in Azure CDN Standard for Microsoft are set at the endpoint level and provide three configuration options. Other tiers provide additional configuration options, which include:
  - Caching rules.
  - Query string caching.
- With the Azure CDN Standard for Microsoft Tier, caching rules are as simple as the following three options:
  - Ignore query strings.
  - Bypass caching for query strings.
  - Cache every unique URL.

Control cache behavior on Azure Content Delivery Networks (3 of 4)

Caching and time to live

- If you publish a website through Azure CDN, the files on that site are cached until their TTL expires. The `Cache-Control` header contained in the HTTP response from origin server determines the TTL duration.
- If you don't set a TTL on a file, Azure CDN sets a default value. However, this default may be overridden if you have set up caching rules in Azure. Default TTL values are as follows:
  - Generalized web delivery optimizations: seven days
  - Large file optimizations: one day
  - Media streaming optimizations: one year

Control cache behavior on Azure Content Delivery Networks (4 of 4)

Content updating

- In normal operation:
  - An Azure CDN edge node will serve an asset until its TTL expires.
  - The edge node reconnects to the origin server when the TTL expires and a client makes a request to the same asset.
  - The node will fetch another copy of the asset, resetting the TTL in the process.
- To ensure that users always receive the latest version of an asset, consider including a version string in the asset URL. This approach causes the CDN to retrieve the new asset immediately.
- You can purge cached content from the edge nodes, which refreshes the content on the next client request.

Interact with Azure Content Delivery Networks by using .NET (1 of 3)

You can use the Azure CDN Library for .NET to automate creation and management of CDN profiles and endpoints. Install the `Microsoft.Azure.Management.Cdn` directly from the Visual Studio Package Manager console or with the .NET Core CLI.

Common actions

- Create a CDN client
- List CDN profiles and endpoints
- Create CDN profiles and endpoints
- Purge an endpoint

Interact with Azure Content Delivery Networks by using .NET (2 of 3)

Create a CDN client

```
static void Main(string[] args)
{
    // Create CDN client
    CdnManagementClient cdn = new CdnManagementClient(new TokenCredentials(authResult.AccessToken))
        { SubscriptionId = subscriptionId };
}
```

Interact with Azure Content Delivery Networks by using .NET (3 of 3)

List CDN profiles and endpoints

```
private static void ListProfilesAndEndpoints(CdnManagementClient cdn)
{
    // List all the CDN profiles in this resource group
    var profileList = cdn.Profiles.ListByResourceGroup(resourceGroupName);
    foreach (Profile p in profileList)
    {
        Console.WriteLine("CDN profile {0}", p.Name);

        //List all the CDN endpoints on this CDN profile
        Console.WriteLine("Endpoints:");
        var endpointList = cdn.Endpoints.ListByProfile(p.Name, resourceGroupName);
        foreach (Endpoint e in endpointList)
        {
            Console.WriteLine("-{0} ({1})", e.Name, e.HostName);
        }
        Console.WriteLine();
    }
}
```

Summary and knowledge check

In this module, you learned how to:

- Explain how the Azure Content Delivery Network works and how it can improve the user experience.
- Control caching behavior and purge content.
- Perform actions on Azure CDN by using the Azure CDN Library for .NET.

1 When publishing a website through Azure CDN, the files on that site are cached until their time-to-live (TTL) expires. What is the default TTL for large file optimizations?

Discussion and lab

Group discussion questions

- You are building a cloud-native application using Azure that users in three regions will use: North America, Asia, and West Europe. You decide to implement a caching strategy with Redis. Describe what would be a good way to determine in which region the Redis cache should be physically located and why.
- Describe how an Azure CDN works.
- Can you describe a scenario where you would recommend Azure CDN over Azure Cache for Redis?

Lab 12: Enhance a web application by using the Azure Content Delivery Network

In this lab, you will implement the Azure Content Delivery Network capabilities to provide a caching solution based on customer locations. The lab configures a storage account for image and video files, which are impacted the most by the latency issues. You will use the Azure Content Delivery Network to implement the caching solution to aid in reducing latency for these image and video files.

http://aka.ms/az204labs

- Exercise 1: Create Azure resources
- Exercise 2: Configure Content Delivery Network and endpoints
- Exercise 3: Upload and configure static web content
- Exercise 4: Use Content Delivery Network endpoints

End of presentation