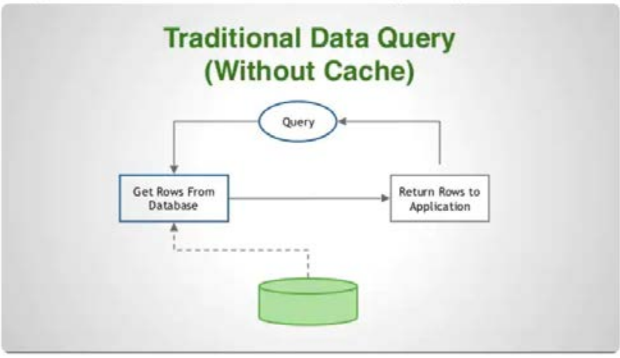
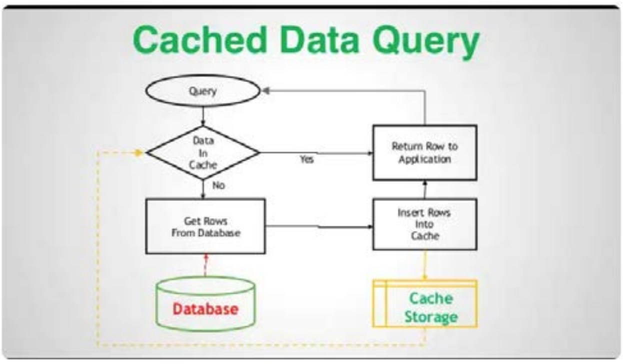
Using .NET(Csharp) Implemented Data Storing & Management Azure Databases (Redis Cache)

Problem Statement :

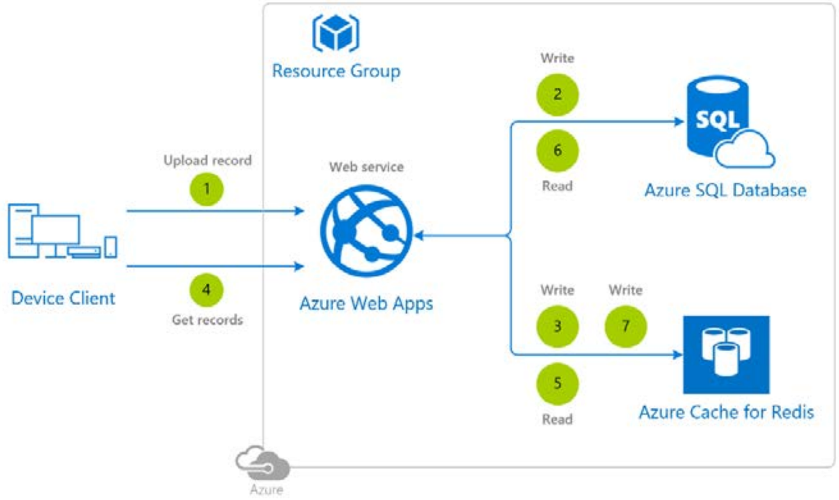
- Our Company firstly use SQL State management to store the Server side data.
- In terms of performance, a SQL Server-based session store is possibly the slowest option. Because your session variables are stored in a physical database, it takes more time to get them in and out of the database. This affects the performance of your Web site.
- Because you are storing your data in a SQL Server database, you need to have a SQL Server license. This can add to overall cost of your Web site.
- This method requires that all the data stored in session variables must be serializable. This may force you to mark your own classes as [Serializable] if you want to store them in a session.

Solution Architecture:





Azure Solutions:



Modern applications mostly work with a large amount of data. In this scenario, when you retrieve data from a database, it typically finds the table and gets the results that it sends back to the user. The performance, in such case, goes down due to multiple requests. So, to reduce some number of requests, you can use cache data that does not change frequently.

Redis Cache is an open source, in-memory database that is used for improving the performance of an application by retrieving and storing the data in the cache memory using a Key-value format. Azure Redis Cache is a feature-rich functionality that gives you access to secure, low-latency, high-performance throughput

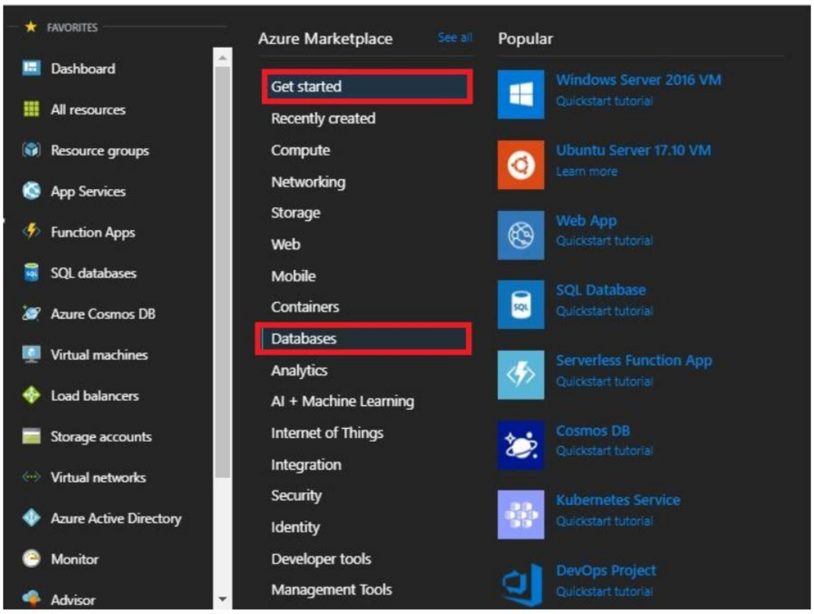
Technical Implementation:

Configure Azure Cache for Redis settings

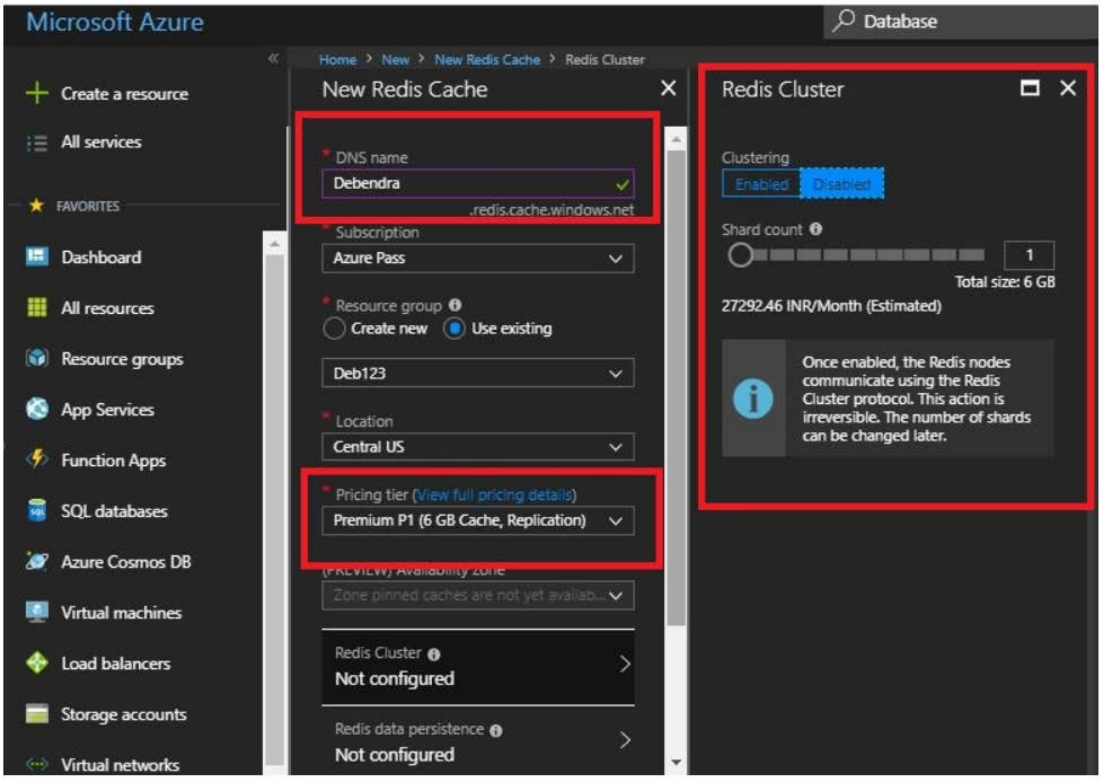
1. To create a cache, sign in to the [Azure portal]() and select **Create a resource**.

2. On the **New** page, select **Databases** and then select **Azure Cache for Redis**.

Redis Configuration :



Steps To create Database :

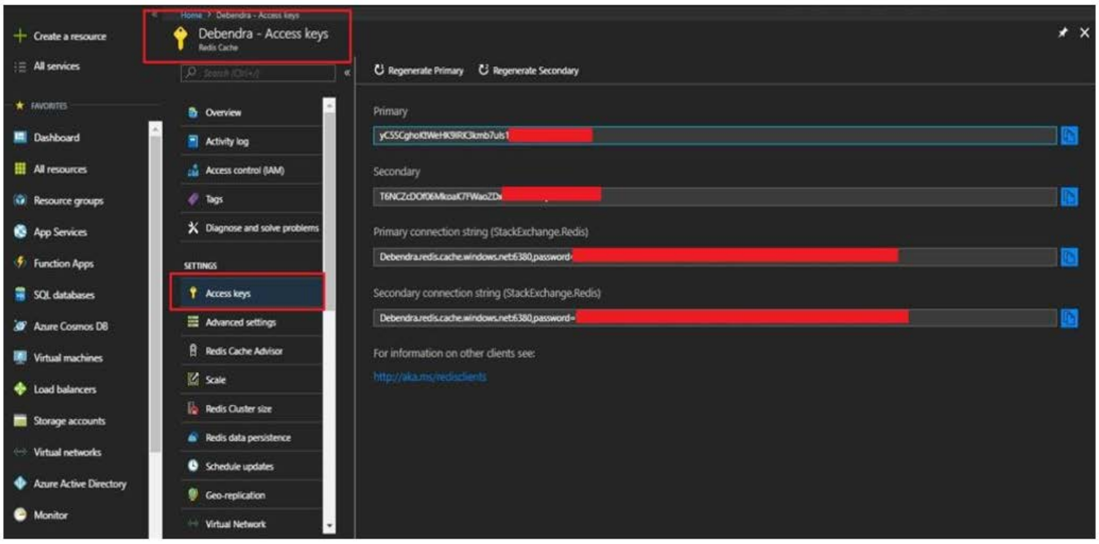


On the **New Redis Cache** page, configure the settings for your new cache.

Setting	Choose a value	Description
Subscription	Drop down	The subscription under which to

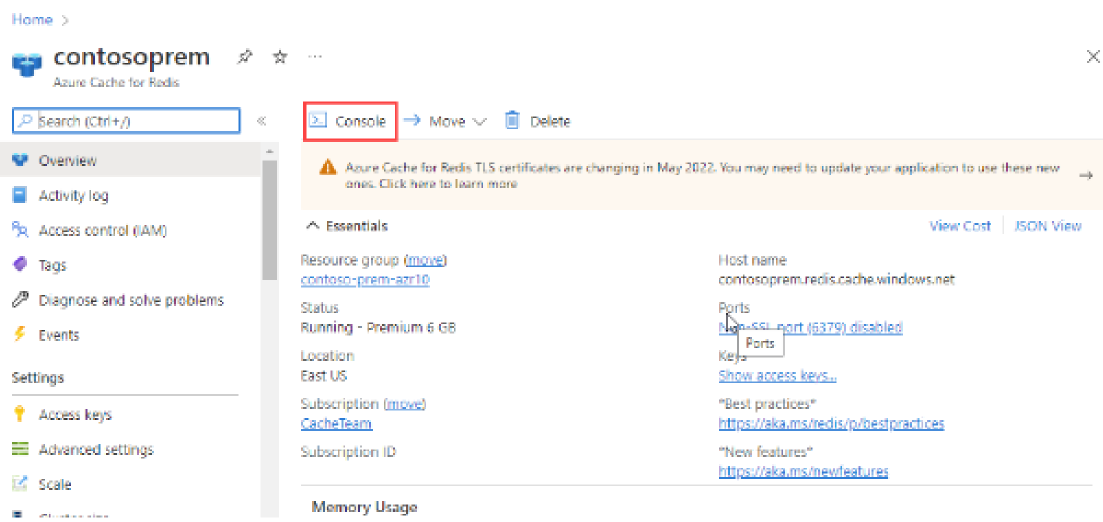
Setting	Choose a value	Description
	and select your subscription.	create this new Azure Cache for Redis instance.
Resource group	Drop down and select a resource group, or select **Create new** and enter a new resource group name.	Name for the resource group in which to create your cache and other resources. By putting all your app resources in one resource group, you can easily manage or delete them together.
DNS name	Enter a unique name.	The cache name must be a string between 1 and 63 characters that contain only numbers, letters, or hyphens. The name must start and end with a number or letter, and can't contain consecutive hyphens. Your cache instance's *host name* will be *<DNS name>.redis.cache.windows.net*.
Location	Drop down and select a location.	Select a region near other services that will use your cache.
Cache type	Drop down and select a tier.	The tier determines the size, performance, and features that are available for the cache. For more information, see Azure Cache for Redis Overview.

Azure Portal and check the access key for the Redis Cache :



The Redis Console does not work with **VNet**. When your cache is part of a VNet, only clients in the VNet can access the cache. Because Redis Console runs in your local browser, which is outside the VNet, it can't connect to your cache.

To access the Redis Console, select **Console** tab in the working pane of Resource menu.



To issue commands against your cache instance, type the command you want into the console.

Driver Code :

```
public HttpResponseMessage GetAllInfoClear([FromUri] string
SubscriptionName, string Type)  //--- To Clear All Data From
HashTable
    {
        try
        {

            FillHashTable.ClearServerHash(SubscriptionName, Type);
            return this.Request.CreateResponse(
```

```
                                       HttpStatusCode.OK,
                                       new ReturnObject()
                                       {
                                           Code = 0,
                                           Message = "All hash tables
clear"
                                       });
        }
        catch (Exception ex)
        {
            ex.Data.Add("SubscriptionName", SubscriptionName);
            ExceptionUtility.LogException(ex);
            return this.Request.CreateResponse(
                HttpStatusCode.InternalServerError, new
ReturnObject()
                {
                    Code = 0,
                    Message = "Error"
                });
        }

    }

    [HttpGet]
    public HttpResponseMessage GetRedis(string SubscriptionName)  //-
-- To Clear All Data From HashTable
    {
        try
        {
            var client = new
RestClient("http://localhost:50301............");
            client.Timeout = -1;
            var request1 = new RestRequest(Method.GET);
            IRestResponse response1 = client.Execute(request1);
            string op = response1.Content;
            op = op.Replace("\"", string.Empty).Trim();

            //WebRequest request =
WebRequest.Create("http://localhost:50301...........");

            //using (WebResponse response =
(HttpWebResponse)request.GetResponse())
            //{
            //    using (var reader = new
StreamReader(response.GetResponseStream()))
            //    {
            //        JObject parseJson =
JObject.Parse(reader.ReadToEnd());
            //        var getJResult = parseJson["status"];
            //        if (getJResult.ToString() == "OK")
            //        {
            //            var getJsonres = parseJson["results"][0];
            //        }
            //    }
            //}
```

```
                        return this.Request.CreateResponse(
                       HttpStatusCode.OK,
                       new ReturnObject()
                       {
                           Code = 0,
                           Message = op
                       });
        }
        catch (Exception ex)
        {
            ex.Data.Add("SubscriptionName", SubscriptionName);
            ExceptionUtility.LogException(ex);
            return this.Request.CreateResponse(
                HttpStatusCode.InternalServerError, new
ReturnObject()
                {
                    Code = 0,
                    Message = "Error"
                });
        }

    }



  string FlagForDataStorage = string.Empty;
    public static ConcurrentDictionary<string, string>
dictSubscriptionData;
    private static Lazy<ConnectionMultiplexer> lazyConnection = new
Lazy<ConnectionMultiplexer>(() =>
    {
        string cacheConnection =
ConfigurationManager.AppSettings["CacheConnection"].ToString();
        return ConnectionMultiplexer.Connect(cacheConnection);
    });

    public static ConnectionMultiplexer Connection
    {
        get { return lazyConnection.Value; }
    }
    public string MakeItemKey(params string[] keys)
    {
        return String.Join("_", keys);
    }

    public dynamic GetSetFromRedisGeneralConfiguration(string
SubscriptionName, string ItemKey, Int64 expire = 60)
    {
        try
        {
            if
(String.IsNullOrEmpty(ConfigurationManager.AppSettings["CacheConnecti
on"].ToString()))
            {
                return null;
            }
            IDatabase cache = Connection.GetDatabase();
            string serializedData =
cache.StringGet(String.Concat(SubscriptionName, "-", ItemKey));
            if (String.IsNullOrEmpty(serializedData))
            {
```

```
                dynamic objParam = new ExpandoObject();
                objParam.SubscriptionName = SubscriptionName;
                DataTable dtSubscriptionGeneralInfo = new
DataAccess().GetDataTable("CommonSubscriptionGeneralInfo", objParam);
                foreach (DataColumn dc in
dtSubscriptionGeneralInfo.Columns)
                {
                    if
(String.IsNullOrWhiteSpace(Convert.ToString(dtSubscriptionGeneralInfo
.Rows[0][dc])))
                    {
                        serializedData = "";
                    }
                    else
                    {

cache.StringSet(String.Concat(SubscriptionName, "-", dc.ToString()),
Convert.ToString(dtSubscriptionGeneralInfo.Rows[0][dc]),
TimeSpan.FromMinutes(expire));
                    }
                }
                serializedData =
cache.StringGet(String.Concat(SubscriptionName, "-", ItemKey));
                return serializedData;
            }
            else
            {
                return serializedData.ToString();
            }
        }
        catch (Exception ex)
        {
            return null;
        }
    }

    public dynamic GetSetFromRedis(string SubscriptionName, string
ItemKey, Int64 expire = 60)
    {
        try
        {
            if
(String.IsNullOrEmpty(ConfigurationManager.AppSettings["CacheConnecti
on"].ToString()))
            {
                return null;
            }
            IDatabase cache = Connection.GetDatabase();
            string serializedData =
cache.StringGet(String.Concat(SubscriptionName, "-", ItemKey));
            if (!String.IsNullOrEmpty(serializedData))
            {
                return serializedData.ToString();
            }
            else
            {
                return null;
            }
        }
        catch (Exception ex)
        {
```

```
            return null;
        }
    }
```

```
public HttpResponseMessage GetOptionalHoliday(dynamic obj)
{
    RedisCacheProvider _cacheProvider = new RedisCacheProvider();
    try
    {
        if (HttpContext.Current.Session["BrokenAccess"] == null || Convert.ToBoolean(HttpContext.Current.Session["BrokenAccess"]) == false)
        {
            objEmployeeGeneralInfo = new EmployeeGeneralInfo();
            if (objEmployeeGeneralInfo.CheckTokenIsAliveOrNot(ref obj) == true)
            {

                ReturnJsonObject JsonObjAttribute = null;
                string SubscriptionName = Convert.ToString(obj.TokenValue);
                SubscriptionName = SubscriptionName.ToUpper();

                string ModuleCode = "LMS";
                string SubModuleCode = "OptionalHoliday";
                bool blnAttributeBased = false;

                string Response = string.Empty;
                obj.ModuleCode = ModuleCode;
                obj.SubModuleCode = SubModuleCode;
                //SignUpGeneralInfo[] ArrSignUPInfoValue = (SignUpGeneralInfo[])objCommon.GetSignUpGeneralnfo("NEWCLIENT", SubscriptionName.ToUppe
                //if (ArrSignUPInfoValue.Length > 0)
                //{
                obj.SignUpID = Convert.ToInt64(_cacheProvider.GetSubscriptionInfoRedis(SubscriptionName + "-SignUpID", SubscriptionName));
```

CHALLENGE FACED :

1> Need To Clear Redis cache after hrs.
2> Facing Issues while migrating to Redis session state.

BUSINESS BENEFIT :

- Redis supports pipelining, so it is possible to send multiple commands at once, a feature often exploited by real world applications. **Redis pipelining is able to dramatically improve the number of operations per second a server is able do deliver**. Using pipelining results in a significant increase in performance.
- Reducing Server Costs.
- Minimal Latency for great Customer Experience
- Less Management than Traditional

References:

https://learn.microsoft.com/en-us/azure/azure-cache-for-redis/cache-configure
https://github.com/Huachao/azure-content/blob/master/articles/redis-cache/cache-configure.md
https://azure.microsoft.com/en-in/products/cache/
https://www.c-sharpcorner.com/article/azure-redis-cache-with-a/

- by Amol Dingankar
Technical Lead (ZingHR)