Microsoft

Optimize Costs on Azure

Azure Data Services – Cost Optimization Reference Guide

Prepared by

Data SQL Ninja Engineering Team (datasqlninja@microsoft.com)

Disclaimer

The High-Level Architecture, Migration Dispositions and guidelines in this document is developed in consultation and collaboration with Microsoft Corporation technical architects. Because Microsoft must respond to changing market conditions, this document should not be interpreted as an invitation to contract or a commitment on the part of Microsoft.

Microsoft has provided generic high-level guidance in this document with the understanding that MICROSOFT MAKES NO WARRANTIES, EXPRESS OR IMPLIED, WITH RESPECT TO THE INFORMATION CONTAINED HEREIN.

This document is provided "as-is". Information and views expressed in this document, including URL and other Internet Web site references, may change without notice.

Some examples depicted herein are provided for illustration only and are fictitious. No real association or connection is intended or should be inferred.

This document does not provide you with any legal rights to any intellectual property in any Microsoft product. You may copy and use this document for your internal, reference purposes.

© 2019 Microsoft. All rights reserved.

Note: The detail provided in this document has been harvested as part of a customer engagement sponsored through the Data SQL Ninja Engineering.

Table of Contents

1, Introduction

This whitepaper outlines the best practice for cost optimization in Microsoft Azure Platform. It provides you with a practical approach to cost management and highlights the tools available to you as you address your organization's cost challenges. Azure makes it easy to build and deploy cloud solutions. However, it's important that those solutions are optimized to minimize the cost to your organization. Following the principles outlined in this document and using our tools will help to make sure your organization is prepared for success.

2, Cost management best practice in Azure

2.1 Choosing the right price plan working for you best

Several of the most common billing models are identified below.

Free

- 12 months of popular free services
- $200 in credit to explore services for 30 days
- 25+ services are always free

Pay as you go

- No minimums or commitments
- Competitive Pricing
- Pay only for what you use
- Cancel anytime

Enterprise Agreement (EA)

- Options for up-front monetary commitments
- Access to reduced Azure pricing

The EA model, or say the Pre-pay option, usually offers the discount comparing Pay-As-You-Go (PAYG). Taking Databricks as an example, there are some options like one-year plan, 3-y plan and PAYG. Details are here,

https://azure.microsoft.com/en-us/pricing/details/databricks/

For the smallest commitment 25,000 DBCU in Databricks, 1-y plan can save 6% and 3-y plan can do 8% for your organization. The more committed, the more savings for you. If the commitments for DBCU (Databricks) commit unit is 2,000,000, then it saves you 33% and 37% respectively in 1-y plan and 3-y plan.

2.2 Choose the right Type/size for the VM and other products

This is the idea of using only what you need. For the organizations to build the whole set of development and production environments in Azure, one of most important decisions is to decide what is the proper size and resource type for the different environments. Assume there are development environment (Dev), QA environment (QA) and the formal end-user facing environment (Prod), usually you can assign the less CPU/memory/storage in lower environments like Dev and QA but more resources for Prod environment.

Taking Databricks as an example, for lower environments (DEV/QA/Stage), standard tier (instead of premium tier) perhaps is enough for testing purpose.

2.3, Use auto-scaling/auto-termination

Use autoscaling and auto-termination to reduce costs during off hours or when nodes are idle is another great feature to reduce the cost.
For Azure Databricks, it does provide this option. There is an option you can set the auto-terminate after 1 hour or maybe 30 mins. For this typical data processing engine, in the lower environments, basically you have deployment phase, data processing phase, user testing phase and result verification phase. The peek time using, let's say 100 nodes should only appear at the phase of data processing. After that, Clusters could be scale down for testing purpose and result verification phases.

More details, please refer to this document,

https://docs.microsoft.com/en-us/azure/architecture/best-practices/auto-scaling

2.4, Automation deployment/testing

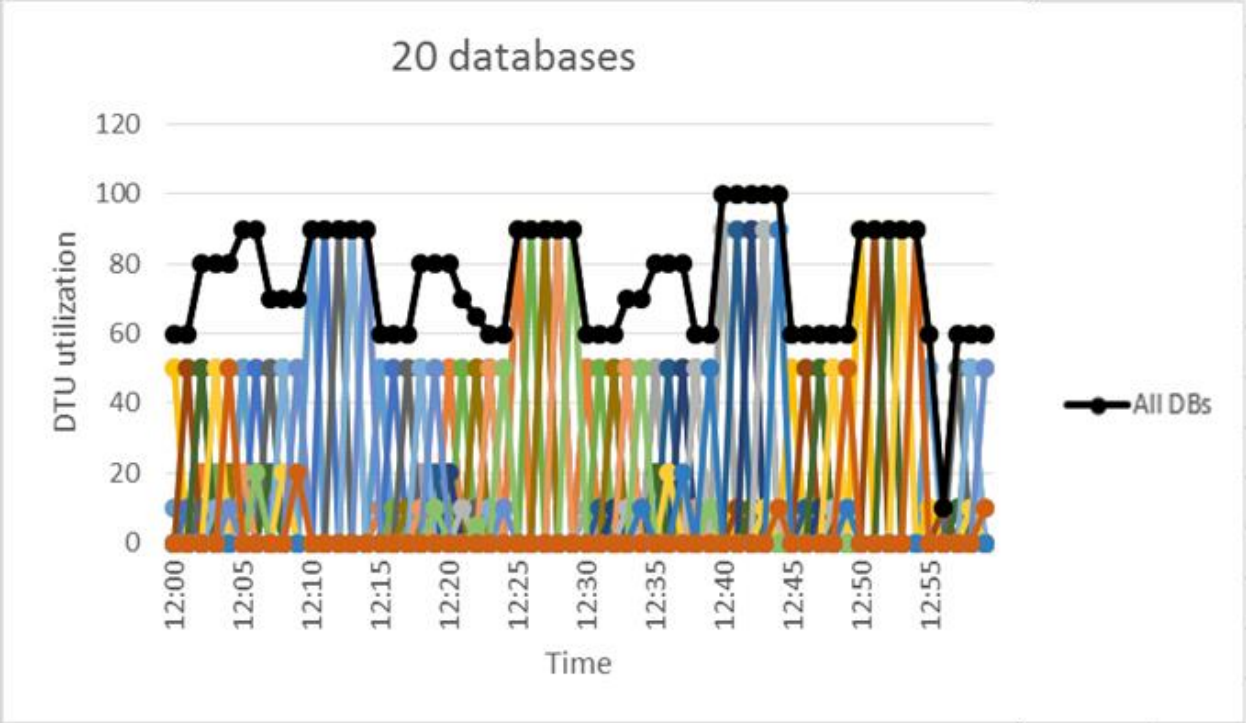
Manual deployment doesn't only generate more deployment errors but also make the process lengthy. Typically, it could take 4~8 hours for the release engineers to deploy application and database changes. On the contrast, automatic deployment/test can dramatically bring down the up-running time of clusters/azure products.
One of helping tools you can check is Azure DevOps. For more details, https://dev.azure.com/

2.5, More PaaS less IaaS if possible

Journey starts with IaaS (Infrastructure as a Service) like setting up VMs and hosting apps and SQL Servers. But considering using more PaaS (Platform as a Service) features of Azure as you can, will not only reduce your maintenance (like upgrading software) effort but also cut the bill of the product license fee if there is no feature limitations.

2.6, Using the concept of “Pool”

In many Azure data services, “Pool” is a great feature to save the cost. SQL Elastic Pools work best for hosting multiple DBs with diff peek time. Likewise, in Databricks, Pools can be attached to multiple clusters. Below illustration shows how all 20 databases share to use 100 DTUs (Black line).



This example is ideal for the following reasons:

- There are large differences between peak utilization and average utilization per database.
- The peak utilization for each database occurs at different points in time.
- eDTUs are shared between many databases.

2.7 Choose the right Blob storage

Azure storage offers different access tiers, which allow you to store blob object data in the most cost-effective manner. The available access tiers include:

- Hot - Optimized for storing data that is accessed frequently.
- Cool - Optimized for storing data that is infrequently accessed and stored for at least 30 days.
- Archive - Optimized for storing data that is rarely accessed and stored for at least 180 days with flexible latency requirements (on the order of hours).

The following considerations apply to the different access tiers:

- Only the hot and cool access tiers can be set at the account level. The archive access tier isn't available at the account level.
- Hot, cool, and archive tiers can be set at the blob level.

- Data in the cool access tier can tolerate slightly lower availability, but still requires high durability, retrieval latency, and throughput characteristics similar to hot data. For cool data, a slightly lower availability service-level agreement (SLA) and higher access costs compared to hot data are acceptable trade-offs for lower storage costs.
- Archive storage stores data offline and offers the lowest storage costs but also the highest data rehydrate and access costs.

Data stored in the cloud grows at an exponential pace. To manage costs for your expanding storage needs, it's helpful to organize your data based on attributes like frequency-of-access and planned retention period to optimize costs. Data stored in the cloud can be different based on how it's generated, processed, and accessed over its lifetime. Some data is actively accessed and modified throughout its lifetime. Some data is accessed frequently early in its lifetime, with access dropping drastically as the data ages. Some data remains idle in the cloud and is rarely, if ever, accessed after it's stored.

Data storage price table shows below, (All prices are per GB, per month.)

	Hot	Cool	Archive
First 50 terabyte (TB) / month	$0.0208 per GB	$0.0152 per GB	$0.00099 per GB
Next 450 TB / Month	$0.0200 per GB	$0.0152 per GB	$0.00099 per GB
Over 500 TB / Month	$0.0192 per GB	$0.0152 per GB	$0.00099 per GB

2.8, Shift workloads to serverless

Serverless computing automatically scales compute based on workload demand and bills for the amount of compute used. Azure Functions is a serverless computing solution that lets you run code as functions without the need to provision or manage any servers. Likewise, Azure SQL Database serverless is a compute tier for single databases that automatically scales compute based on workload demand and bills for the amount of compute used per second. The serverless compute tier also automatically pauses databases during inactive periods when only storage is billed and automatically resumes databases when activity returns.

https://docs.microsoft.com/en-us/azure/sql-database/sql-database-serverless

2.9, Transfer Workload to Container

Transfer Workload to Containers (Azure Kubernetes Service (AKS)) - VMs are Azure's most popular computing option, but not the only one. Containers are lighter than VMs and are less expensive due to their lower footprint and faster operability. This factor attracts a lot of organizations to move from VMs to containers.

2.10, Improve the app/DB performance if possible

Improving application and database performance is another example to save the cost by bringing down the service up-running time. The real case I saw, by changing the Spark code, running time on Databricks from 8 hours to 30 mins.

2.11, Cost management in Azure

Break down the usage/cost and keep monitoring/reviewing with the core team. Scale it up/down based on it.
https://ms.portal.azure.com/#blade/Microsoft_Azure_Billing/ModernBillingMenuBlade/Overview

2.12, Azure Advisor

There is one pillar for cost advisor giving you the recommendations to save $$$
https://ms.portal.azure.com/#blade/Microsoft_Azure_Expert/AdvisorMenuBlade/Cost

3, Reference

Reduce service costs using Azure Advisor: https://docs.microsoft.com/en-us/azure/advisor/advisor-cost-recommendations#buy-reserved-virtual-machine-instances-to-save-money-over-pay-as-you-go-costs

SQL database elastic pool: https://docs.microsoft.com/en-us/azure/sql-database/sql-database-elastic-pool

4, Feedback and suggestions

If you have feedback or suggestions for improving this data migration asset, please contact the Data SQL Ninja Team (datasqlninja@microsoft.com). Thanks for your support!

Note: For additional information about migrating various source databases to Azure, see the Azure Database Migration Guide.