

Elastic pool scale stuck due to long recovery database

Last updated by | Vitor Tomaz | Feb 24, 2023 at 3:24 AM PST

Contents

- [Issue](#)
- [Investigation/Analysis](#)
- [Mitigation](#)
- [RCA Template](#)
- [Internal Reference](#)
- [Root Cause Classification](#)

Issue

Standard elastic pool scale operation was stuck.

Investigation/Analysis

1. Confirm Kusto that the elastic pool scale operation was stuck at "MovingDatabaseToNewContainer" state for long time(over 24 hours will timeout).

```
MonManagementOperations
| where TIMESTAMP between (datetime(starttime)..datetime(endtime))
| where request_id =~ '393fd5d8-fc52-4118-93cc-2d367f666fc3'
//| where operation_type contains "UpdateLogicalElasticPool"
| project TIMESTAMP, event, request_id , operation_type, operation_parameters, operation_result, error_message
```



Moreover, note down the target SLO for the elastic pool from input parameters.

```

<?xml version="1.0"?>
<InputParameters xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-inst
  <SubscriptionId>d1096dc0-7e08-49ab-a280-16a3c47e0886</SubscriptionId>
  <ResourceGroupName>resource_group</ResourceGroupName>
  <LogicalServerName>server_name</LogicalServerName>
  <LogicalResourcePoolId>resource pool id</LogicalResourcePoolId>  <= This is the pool id
  <LogicalResourcePoolName>resource pool name</LogicalResourcePoolName>  <= This is the pool name
  <RequestedDtuGuarantee>300</RequestedDtuGuarantee>
  <RequestedDatabaseDtuGuarantee>0</RequestedDatabaseDtuGuarantee>
  <RequestedDatabaseDtuCap>200</RequestedDatabaseDtuCap>
  <RequestedStorageLimitInMB>512000</RequestedStorageLimitInMB>
  <Flags>None</Flags>
  <RequestedEdition>Standard</RequestedEdition>
  <ZoneResilientPool>false</ZoneResilientPool>
  <RequestedMinVCoresPerDatabase xsi:nil="true" />
  <RequestedMaxVCoresPerDatabase xsi:nil="true" />
  <RequestedSafetyChecks xsi:nil="true" />
  <LongTransactionLogInBytesThreshold xsi:nil="true" />
  <MaxCpuPercentageThreshold xsi:nil="true" />
  <MaxLogWriteRatePercentageThreshold xsi:nil="true" />
  <SkipBusinessHoursCheck xsi:nil="true" />
  <SkipResourcePoolSpaceUtilizationCheck xsi:nil="true" />
  <SkipEffectivePlacementFeatureCheck xsi:nil="true" />
  <RequestedLogStorageLimitInMB xsi:nil="true" />
  <MaxDbCopiesAllowedInParallel xsi:nil="true" />
  <RequestedReadScaleUnits xsi:nil="true" />
  <RequestedMinCapacity xsi:nil="true" />
  <RequestedXlogServiceLevelObjectiveId xsi:nil="true" />
</InputParameters>

```

3. Open XTS "Sterling\elastic pool.xts" and check "Resource Pool Utilization" for resource utilization. You will see avg_data_io_percent always has spike.
4. Further verify from kusto for long term(30 days) io performance of elastic pool

Execute: [Web] [Desktop] [Web (Lens)] [Desktop (SAW)] <https://sqlazureeus12.kustomfa.windows.net/sqlazure1>

```
MonResourcePoolStats
| where originalEventTimestamp >= ago(30d)
| where LogicalServerName == "server_name" and resource_pool_name == "resource_pool_name"
| summarize min(avg_data_io_percent), max(avg_data_io_percent), avg(avg_data_io_percent) by bin(start_time, 1h)
| render timechart
```

From the result, you can see periodic peaks of heavy IOPS. 5. Open XTS "Sterling\elastic pool size analyzer.xts" to check "Physical databases with remote replica under server" tab. You can get information whether the elastic pool is using remote storage, storage_type and blob tier.

6. Search from ICM portal to see if there is any database recovery or availability incident around same issue time. If not, you can pick the database name inside the elastic pool to check from Kusto.

```
// Find login errors in a given timeframe for a given Logical Server-DB name

MonLogin
| where TIMESTAMP between (datetime(starttime)..datetime(endtime))
| where logical_server_name =~ "servername" and database_name =~ "databasename"
| where error <> 0 and isempty(error)==0
//| project TIMESTAMP, error, state, NodeName, MachineName, package, message
| summarize count() by bin(TIMESTAMP, 5m), AppName, AppTypeName, error, state, package, NodeName
```

Usually there should be error 40613, state 126 during updateSlo and recovery.

```
//Find recovery information from MonRecoveryTrace
MonRecoveryTrace
| where TIMESTAMP > datetime({StartTime}) and TIMESTAMP < datetime({EndTime})
| where LogicalServerName == "{LogicalServerName}" and AppName == "{AppName}" //and database_name == "{DatabaseName}"
| where database_name !in("msdb","master","model","tempdb", "mssqlsystemresource","model_userdb", "model_mast")
| order by originalEventTimestamp asc
| serialize
| extend PrevTimestamp=prev(originalEventTimestamp, 1)
| extend elapse_time_sec = (originalEventTimestamp - PrevTimestamp) / 1s
| project elapse_time_sec, originalEventTimestamp, database_name, trace_message
| order by originalEventTimestamp asc nulls last
//| order by elapse_time_sec asc
```

It's not easy to track the information since the long recovery of database is impacting the update SLO operation of elastic pool, especially when there are many databases in the pool. One suggestion is to start investigation for those large and busy databases which are highly possibly to encounter long recovery issue.

Mitigation

For stuck workflow, please raise an ICM to PG, so they can help unblock the customer by fixing the broken state.

RCA Template

When a Standard elastic pool scale starts, what we do is create a new SQL instance of the new compute size on a separate compute node. This target pool instance is created up front and brought fully online before we start

moving over the databases which use remote storage to the new compute node.

For each individual database in the elastic pool, we go to the source instance and kill all outstanding transactions for this database, then wait for these transactions to all kill off.



The reason we kill all the transactions is to avoid long recovery times when flipping over to the target instance. When all the transactions are killed off, we quickly block session access to source instance, fail over the remote storage blobs to the target instance and attach the databases and bring them online. What went wrong in your case is there was a very long database recovery in progress which caused the pool scale to get stuck unable to proceed. Because of the unavailable database in long recovery, the pool scale could not move forward or roll back.

Note disk flush failed due to underlying storage issue where we saturated the Azure storage account with flush (storage was reporting server busy errors).

So this triggered a long recovery on the target database once we flipped over, which caused the inability to access the database for a long period of time.

Your pool is **<elastic pool service tier>** which allows a certain level of IOPS as well as you have **<database number in the pool>** concurrent databases in this pool and I see you are approaching 80-100% IOPS saturation frequently (daily). So with **<elastic pool service tier>** level of IOPS you experienced a longer than usual recovery as there are limited IOPS allowed to perform IOPS to each database in the pool.

Internal Reference

- [Incident 295717938](#) 
- Correlated availability ICM [Incident 294629666](#) 

Repair Item: [1740801](#) 

Root Cause Classification

Cases resolved by this TSG should be coded to the following root cause:

/Azure SQL v3/CRUD/Elastic Pool/Scale failure

How good have you found this content?



-