Canned_RCA_Scaling_Slow_or_Stuck

Last updated by | Charlene Wang | Jan 3, 2023 at 12:30 AM PST

Contents

Issue

This TSG covers pre-canned RCA's for Update SLO stuck issues **Do not copy past and send the content as it is to the customer, please make sure the RCA is reviewed and approved by PG ICM owner or your SME\TA.**

Issue Type

Slow scale due to overloaded database

On **<Start Time>** you attempted to upgrade database **<database name>** to **<Target SLO>**. Microsoft engineers looked at the operation logs in detail and found the following happened:

Note when the database upgrade starts, what we do is create a new SQL instance of the larger size on a separate compute node. This instance is created up front and brought fully online before we start copying database file data from source instance. The target instance does not yet attach the database files however. Next we go to the source instance and kill all outstanding transactions, wait for these to all kill off. The reason we kill all the transactions is to avoid a 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 disks to the target instance and attach the database and bring it online. What went wrong in your case is there were large outstanding transactions (3GB of log worth) in progress when we started killing the transactions. We did successfully kill all the transactions, but BEFORE we flipped over the source database went into recovery mode due to failures flushing the transaction log to disk. Disk flush failed due to underlying storage issue where we overloaded the storage account with flush (storage 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 time.

Stuck Scale Due to Stuck RefreshExternalSecretsBeforeActivatingTargetDatabase

When a Basic to Standard database upgrade starts, we create a new SQL instance of the new compute size on a separate compute node. This instance is created up front and brought fully online before we detach database files from source instance and re-attach to the target instance. In addition, before we flip over the database files we need to push down encryption certificates for the database to the target instance so it can decrypt and bring the database online on the target instance. In your case the workflow to push down the instance certificates got stuck due to a race condition and was unable to proceed without manual intervention by Microsoft engineers. I've examined the code that does this operation and I have created a repair to ensure we don't get stuck at this step. We also have new alerts to detect this special case of stuck workflow in place to alert engineers when this happens.

We are committed to keep pushing scale latency down and improving platform reliability and experience. We do appreciate you as a customer and we understand that you’ve tied success of your product to success of SQL Azure platform. This is not something we take lightly. Please continue scaling your database to manage cost and meet traffic demands.

Internal reference

- Icm# https://portal.microsofticm.com/imp/v3/incidents/details/268221233/home
- Repair: https://msdata.visualstudio.com/Database Systems/_workitems/edit/1503034

Long Recovery

Sample ICM

On **<Start Time>** you attempted to upgrade database **<database name>** to **<Target SLO>**. Microsoft engineers looked at the operation logs in detail and found the following happened:

Note when the database upgrade starts, what we do is create a new SQL instance of the larger vCore size on a separate compute node. This instance is created up front and brought fully online before we start failing over the source instance. The target instance does not yet attach the database files however. Next we go to the source instance and kill all outstanding transactions, wait for these to all kill off. The reason we kill all the transactions is to avoid a 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 disks to the target instance and attach the database and bring it online. What went wrong in your case is there were large outstanding transactions (3GB of log worth) in progress when we started killing the transactions. We did successfully kill all the transactions, but BEFORE we flipped over the source database went into recovery mode due to failures flushing the transaction log to disk. Disk flush failed due to underlying storage issue where we overloaded the storage account with flush (storage 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 time.

I don't see anything on your end that you did to cause this issue. You are likely driving the log rate fairly hard for a standard database but I don't see excessive LOG growth over time (looking back 1 week after the fact).

Using new feature called Accelerated Database Recovery that will greatly reduce the recovery time here, you can read about it here:

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

Thanks for using SQL Azure database and we apologize for this incident.

Stuck Scale Due to UpdateVCoreLogMaxSizeOnUpdate

When a General Purpose or Business Critical database upgrade starts, we create a new SQL instance of the new compute size on a separate compute node. This instance is created up front and brought fully online before we start copying or moving database files from source instance. During General Purpose or Business Critical scale when the maximum database size is modified we also re-size the log for the database to 30% of the maximum databases size during the scaling operation. In your case the log resize operation got stuck requiring manual intervention from Microsoft engineers to fix. We have since created alert to detect this stuck workflow issue and this alert now is catching and reporting these issues when we find them.

We are committed to keep pushing scale latency down and improving platform reliability and experience. We do appreciate you as a customer and we understand that you’ve tied success of your product to success of SQL Azure platform. This is not something we take lightly. Please continue scaling your database to manage cost and meet traffic demands.

Stuck Scale Due to NodeAgent Issues

When a database upgrade starts, we create a new SQL instance of the new compute size on a separate compute node. This instance is created up front and brought fully online before we start copying or moving database files from source instance. During the scale we also periodically talk to the new compute instance to setup the instance for the scale operation. In your case the new instance was placed on a particular compute node with a stuck agent, and this agent kept failing requests to talk to the new compute instance, causing the scale operation to get stuck forever. This required manual intervention from Microsoft engineers to fix. We tracked down the particular compute node having this issue and we repaired this node to fix the problem. I have also filed a repair to add a specific alert for this stuck scale situation (stuck compute node agent) so that in the future when we hit this, we will be alerted and fix the problematic node.

We are committed to keep pushing scale latency down and improving platform reliability and experience. We do appreciate you as a customer and we understand that you’ve tied success of your product to success of SQL Azure platform. This is not something we take lightly. Please continue scaling your database to manage cost and meet traffic demands.

Stuck Scale Due to Storage Issues

When an elastic pool scale starts, what we do is create a new SQL instance of the new compute size on a separate compute node. This instance is created up front and brought fully online before we start moving the databases from the source to the target instance. For Standard and General Purpose elastic pools, the underlying databases use remote storage, so during pool scale we detach each database from the source instance and re-attach and bring online on the target instance. This operation normally takes 5 seconds per database but one of your databases hit a storage issue during the re-attach phase. This triggered an availability alert and Microsoft engineers stepped in and fixed the issue however this caused the pool scale to remain stuck for a period of time until completely mitigated.

We are committed to keep pushing scale latency down and improving platform reliability and experience. We do appreciate you as a customer and we understand that you've tied success of your product to success of SQL Azure platform. This is not something we take lightly. Please continue scaling your database to manage cost and meet traffic demands.

When an database scale starts, what we do is create a new SQL instance of the new compute size on a separate compute node. This instance is created up front and brought fully online before we start moving the active database from the source to the target instance. For Standard and General Purpose databases, the underlying database uses remote storage, so during a database scale we detach the database from the source instance and re-attach and bring online on the target instance. This operation normally takes 5 seconds but your database hit a storage issue during the re-attach phase where the lease on the storage blob could not be updated. This triggered an availability alert and Microsoft engineers stepped in and fixed the issue however this caused the database scale to remain stuck for a period of time until completely mitigated.

We are committed to keep pushing scale latency down and improving platform reliability and experience. We do appreciate you as a customer and we understand that you've tied success of your product to success of SQL Azure platform. This is not something we take lightly. Please continue scaling your database to manage cost and meet traffic demands.

Slow Scale due to Changing Storage Format

During an elastic pool scale for server pt4ghlrwnt and elastic pool pt4ghlrwnt-ElasticPool-61-iafy on 5/9 internally we converted your internal elastic pool storage configuration to a new more optimized storage configuration model. This triggered a longer than normal scale operation for the elastic pool (3 hours versus normal 5 minutes). This was a one time conversion and subsequent scale operations will not trigger this conversion. During the one time conversion we needed to do a streaming copy from the source databases to the target database files to convert to the new storage format. Normally during an elastic pool scale for Standard and General Purpose we don't do this streaming file copy or conversion so the scale operation is much faster.

We are committed to keep pushing scale latency down and improving platform reliability and experience. We do appreciate you as a customer and we understand that you've tied success of your product to success of SQL Azure platform. This is not something we take lightly. Please continue scaling your database to manage cost and meet traffic demands.

Stuck Operations Due To HA Engine Bugs

- Repair item 13708328
- Repair item 13773542

Latest stuck drop bugs are 13708328 and 13773542 which still exists in production and are not yet fixed

Note when an elastic pool upgrade starts, what we do is create a new SQL instance of the new compute size on a separate compute node. This instance is created up front and brought fully online before we start moving databases from the source to the target instance. Once all the databases are moved to the target instance, we close down and clean up the source instance. In your case during the shutting down of the older source instance, we hit a stuck operation due to a known engine bug that left the instance in a stuck dropping state. Once Microsoft engineers saw the stuck operation they manually intervened and killed the stuck source instance to un-stick the workflow.

We are committed to keep pushing scale latency down and improving platform reliability and experience. We do appreciate you as a customer and we understand that you've tied success of your product to success of SQL Azure platform. This is not something we take lightly. Please continue scaling your database to manage cost and meet traffic demands.

Stuck Database or Pool Move which was triggered for Capacity Reasons

Microsoft sometimes needs to move customer pools and databases between Azure SQL clusters for maintenance. This happens due to different reasons: (1) Decommissioning of hardware heading to out of warranty status (2) Defragmentation of existing clusters. (3) Evacuating databases from faulty hardware Due to architectural limitations, move operations like these prevent our customers from being able to change the database/pool service level objectives and add databases to pools while the operation is going on. The customer's pool was one of such resources that had to be moved. A move operation for a pool requires moving all the databases in a pool from one cluster to another, followed by a failover once all the moves are completed. A bug in the workflow caused us to indefinitely postpone moving one of the DBs in the pool, resulting in a long running operation blocking the customer workflow. Our on-call engineer advanced the long running operation and mitigated the issue. We identified the bug and are currently upgrading our management entities worldwide to fix it. The current estimate is for the bug to be fixed everywhere by end of next week.

Stuck Pool Scale Due to Stuck Copy

You attempted to scale your elastic pool Seaside under server seasidetest and the scaling operation was stuck for a long time. I looked at the operation logs in detail and I found the following happened:

When an elastic pool upgrade starts, what we do is create a new SQL instance of the new compute size on a separate compute node. This instance is created up front and brought fully online before we start copying databases from the source to the target instance. We use geo-copy physical database copy operation to do this and this leaves the source pool fully functional while this copy is ongoing. Once all the databases copies are complete we fail over to the target instance and then we asynchronously clean up the source instance. In your case during the copy phase one of the databases hit a stuck copy operation due that left the database in a stuck copying state. Microsoft engineers had to step in an manually unstick the operation. We are currently investigating why these copies get stuck but don't have a clear root cause yet, they are very rare at this moment. In the meanwhile we are working on putting in place a BOT to catch these long running stuck copy operations and alert us to these stuck operations when these happen.

We are committed to keep pushing scale latency down and improving platform reliability and experience. We do appreciate you as a customer and we understand that you've tied success of your product to success of SQL Azure platform. This is not something we take lightly. Please continue scaling your database to manage cost and meet traffic demands.

TSG GEODR0040

Stuck Pool Scale Due to Premium File Share Issues

During an elastic pool scale internally we attempted to convert your internal elastic pool storage configuration to a new more optimized storage configuration model. This triggered a longer than normal scale operation that became stuck for the elastic pool due to the need to convert file formats. This is typically a one-time conversion and subsequent scale operations will not trigger this conversion. During the one time conversion we needed to do a streaming copy from the source databases to the target database files to convert to the new storage format. Normally during an elastic pool scale for Standard and General Purpose we don't do this streaming file copy or conversion so the scale operation is much faster. Microsoft engineers had to step in and fix the stuck workflow. We have now converted and monitored the storage conversion for you so this will not happen again.

We are committed to keep pushing scale latency down and improving platform reliability and experience. We do appreciate you as a customer and we understand that you’ve tied success of your product to success of SQL Azure platform. This is not something we take lightly. Please continue scaling your database to manage cost and meet traffic demands.

Long Period of Unavailability During Scale Operation Due To Slow Alias Update

When a database scale starts, we create a new SQL Server instance of the new compute size on a separate compute node. This SQL Server instance is created up front and brought fully online before we start transferring data from the database from the source to the target SQL Server instance. During the database transfer to the new compute instance, we use a streaming geo-copy of the database from source to target. While this copy is happening the source database is still fully functional and readable and writable. Once the database streaming copy operation is caught up and ready for failover, we block connections on the source database, kill all outstanding transactions, and then fail over to the target instance. Failing over also updates connectivity information for our connectivity gateways so that the incoming connections to the database are routed to the correct SQL Server instance. This failover phase is put into a critical workflow with high priority since this causes downtime. In your case, during the critical failover phase, an operation to update the connectivity information became stuck and did not update the connectivity records to point to the new database. Microsoft engineers had to step in an manually unstick the operation to restore connectivity based on alerts.

We are committed to keep pushing scale latency down and improving platform reliability and experience. We do appreciate you as a customer and we understand that you’ve tied success of your product to success of SQL Azure platform. This is not something we take lightly. Please continue scaling your database to manage cost and meet traffic demands.

Stuck Pool Scale Due to Bad Encryption Certificate

When an elastic pool upgrade starts, what we do is create a new SQL instance of the new compute size on a separate compute node. This instance is created up front and brought fully online before we start copying databases from the source to the target instance. We use geo-copy physical database copy operation to do this and this leaves the source pool fully functional while this copy is ongoing. Once all the databases copies are complete we fail over to the target instance and then we asynchronously clean up the source instance. In your case during the initial setup of the target instance, the instance did not correctly initialize the encryption certificate, and this caused the instance to keep failing to bootstrap and allow database copying to start. Microsoft engineers had to step in an manually unstick the operation. We are currently investigating why these copies get stuck but don't have a clear root cause yet, they are very rare at this moment. In the meanwhile we are working on putting in place a BOT to catch these long running stuck copy operations and alert us to these stuck operations when these happen.

We are committed to keep pushing scale latency down and improving platform reliability and experience. We do appreciate you as a customer and we understand that you've tied success of your product to success of SQL Azure platform. This is not something we take lightly. Please continue scaling your database to manage cost and meet traffic demands.

Also better one from Brian 😀

During the process of scaling an elastic pool, a new compute container must be created to host the elastic pool. This new compute container will be of the target size. Once that container is created, all of the databases will be moved from the old SQL instance container to the new one (either by copying data or detach/attaching from remote storage). Once this is complete, the old SQL container is destroyed. In the case of this pool scale operation, the creation of the new SQL container got stuck due to an underlying health condition on the cluster hosting it. It got stuck in such a way that the automated provisioning system couldn't recover/retry, and just ended-up waiting instead. This led to the whole pool scale operation being stuck. The mitigation was to manually cancel the pool scale operation and have it be re-issued, which resulted in success. Azure SQL engineers are investigating ways to detect and auto-remediate this condition such that the provisioning system won't get stuck when it sees this again.

From https://portal.microsofticm.com/imp/v3/incidents/details/229802622/home

Long Running Scale Due To Long Running Stuck Copy

When a Business Critical database upgrade starts, what we do is create a new SQL instance of the new compute size on a separate compute node. This instance is created up front and brought fully online before we start copying the database from the source to the target instance. We next use geo-copy physical database copy operation to copy the database from the source to target instance and this leaves the source database fully functional while this copy is ongoing. Once the database copy is fully complete we fail over to the target instance and then we asynchronously clean up the source instance. In your case during the copy phase the databases hit a stuck copy operation due that left the database in a stuck copying state. Microsoft engineers had to step in an manually unstick the operation. We are currently investigating why these copies get stuck but don't have a clear root cause yet, they are very rare at this moment. In the meanwhile we are working on putting in place a BOT to catch these long running stuck copy operations and alert us to these stuck operations when these happen.

Note that in general a Business Critical database typically copies at a rate of 1-2 GB per minute but the copy rate can vary. So scaling very large Business Critical databases is a size of data operation. Also note that we do a physical database copy, so if you have any additional unused space in your physical database files this can extend the time to perform the copy. So to reduce this you can run DBCC SHRINKDATABASE to clean up excessive disk space in the physical databases. I noticed in your case your database is using 508 GB space but the physical file is 824 GB so there is some amount of extra space in the physical files which can extend the time it takes to copy during scale.

We are committed to keep pushing scale latency down and improving platform reliability and experience. We do appreciate you as a customer and we understand that you've tied success of your product to success of SQL Azure platform. This is not something we take lightly. Please continue scaling your database to manage cost and meet traffic demands.

Long Running Scale Due To Inability To Create Space On Target (PLB)

When a Business Critical database upgrade starts, what we do is create new SQL replicas of the new compute size on separate compute nodes. A target instance is created up front and brought fully online before we start copying the database from the source to the target instance. We use geo-copy physical database copy operation to copy the database from the source to target instance and this leaves the source database fully functional while this copy is ongoing. Once the database copy is fully complete we fail over to the target instance and then we asynchronously clean up the source instance. In your case during the copy phase of the scale operation there were multiple databases copying into a single target instance which was requiring more local disk space on the nodes where the database replicas reside. A majority of the nodes had sufficient disk space to copy the incoming data but one node had a very large replica on the node taking up most of the disk space on the node. Normally during scale our disk placement sub-systems will load balance these replicas to free up disk space needed by the incoming replicas but in this case it took a very long time to move the very large replica. This caused out of disk space on one of the replicas on the node which impacted multiple databases.

Once the incoming replicas were stalled this caused additional issues with primary replica unable to persist log to secondary replica nodes which in turn caused unavailability and automatic failover.

We have a repair for this situation currently being tested in pre-production to avoid this problem it will stall the scale operations until available space is present on all the target nodes, plan is to push out this fix in the next month once we fully test it.

We are committed to keep pushing scale latency down and improving platform reliability and experience. We do appreciate you as a customer and we understand that you’ve tied success of your product to success of SQL Azure platform. This is not something we take lightly. Please continue scaling your database to manage cost and meet traffic demands.

From https://portal.microsofticm.com/imp/v3/incidents/details/295194719/home

Long Running Stuck Scale Due To Inability To Revert Hekaton Changes (MarkUpdateSloCompletedForHekatonDb)

During database scale we prepare and initialize the state of SQL Server In-Memory OLTP files during start of scale for each database as well as restore any file changes to SQL Server In-Memory OLTP at the end of the scale operation. This happens on both roll forward and rollback of scale operation. In your scale case the newly created database hit a SQL Server In-Memory OLTP recovery issue and this caused the scale operation to fail and rollback. During rollback the scale operation also got stuck attempting to restore SQL Server In-Memory OLTP files on the source database instance.

We have a repair for this situation currently rolling out to production to avoid the rollback problem, it will just skip any SQL Server In-Memory OLTP file manipulation on scale rollback since it is not needed.

We are committed to keep pushing scale latency down and improving platform reliability and experience. We do appreciate you as a customer and we understand that you’ve tied success of your product to success of SQL Azure platform. This is not something we take lightly. Please continue scaling your database to manage cost and meet traffic demands.

Internal ICM (Don't share with customer)

Long Running Pool Scale Due To ReserveSpaceIdList Regression

When a Business Critical pool upgrade starts, what we do is create new SQL instances of the new compute size on separate compute nodes. A target instance is created up front and brought fully online before we start a streaming copy of the pooled databases from the source to the target instance. We use geo-copy physical database copy operation to copy the databases from the source to target instance and this leaves the source databases fully functional while this copy is ongoing. Note prior to these streaming copies we also send signals to the target compute instance to pre-reserve disk space on the target nodes. This ensures that the target nodes on the new compute instances have sufficient disk space and also that our compute load balancing has enough time on the target nodes to ensure there is sufficient disk space available. Once the database copies are fully complete we fail over to the target instance and then we asynchronously clean up the source instance. In your case during the reservation phase of the scale operation we hit a new regression where the reservation code hit a limit which did not allow more that 95 databases at a time to reserve space in parallel. This caused the 96th streaming copy to fail to start and caused the pool scale to hang requiring manual intervention from Microsoft engineers to fix.

We have expedited a feature switch to turn off this new code and this feature switch is rolling out in safe deployment order to production now. Until this rolls out worldwide we will need to monitor your pools and ensure we mitigate this problem it will stall the pool scale operations without manual intervention. You can avoid this issue as well by ensuring no more than 95 database per pool. We will notify you once the feature switch is pushed out worldwide and you can resume normal scaling.

We are committed to keep pushing scale latency down and improving platform reliability and experience. We do appreciate you as a customer and we understand that you've tied success of your product to success of SQL Azure platform. This is not something we take lightly. Please continue scaling your database to manage cost and meet traffic demands.

Long Running Database Scale Due To Long Running Copy (By Design)

When a Business Critical database upgrade starts, what we do is create a new SQL instance of the new compute size on a separate compute node. This instance is created up front and brought fully online before we start copying the database from the source to the target instance. We next use geo-copy physical database copy operation to copy the database from the source to target instance and this leaves the source database fully functional while this copy is ongoing. Once the database copy is fully complete we fail over to the target instance and then we asynchronously clean up the source instance. In your case the copy phase of the databases proceeded normally at the normal 1-2 GB per minute rate and there were no other unforeseen issues during the scale.

Note that in general a Business Critical database typically copies at a rate of 1-2 GB per minute but the copy rate can vary. So scaling very large Business Critical databases is a size of data operation. Also note that we do a physical database copy, so if you have any additional unused space in your physical database files this can extend the time to perform the copy. So to reduce this you can run DBCC SHRINKDATABASE to clean up excessive disk space in the physical databases. We have public documentation explaining the scale latency here Scale single database resources - Azure SQL Database | Microsoft Docs

We are committed to keep pushing scale latency down and improving platform reliability and experience. We do appreciate you as a customer and we understand that you've tied success of your product to success of SQL Azure platform. This is not something we take lightly. Please continue scaling your database to manage cost and meet traffic demands.

Long Running Pool Scale Due To Long Running Copy (By Design)

https://portal.microsofticm.com/imp/v3/incidents/details/312878125/home

When a Business Critical pool upgrade starts, what we do first is create a new SQL instance of the new compute size on a separate compute node. This instance is created up front and brought fully online before we start copying the databases inside the pool from the source to the target instance. We next use geo-copy physical database copy operation to copy the databases from the source to target instance and this leaves the source database fully functional while this copy is ongoing. For pool scale we typically run a max of 8 parallel database copies at once to reduce copy load on the source instance. Once the database copies are fully completed we fail over all the databases to the target instance and then we asynchronously clean up the source instance. In your case the copy phase of the databases proceeded normally at the normal 1-2 GB per minute rate and there were no other unforeseen issues during the scale.

Note that in general a Business Critical database typically copies at a rate of 1-2 GB per minute but the copy rate can vary. So scaling very large Business Critical databases is a size of data operation. Also note that we do a physical database copy, so if you have any additional unused space in your physical database files this can extend the time to perform the copy. So to reduce this you can run DBCC SHRINKDATABASE to clean up excessive disk space in the physical databases. We have public documentation explaining the scale latency here Scale single database resources - Azure SQL Database | Microsoft Docs

In this particular case, the "long pole" of the pool scale operation was the database act-pr-act-db which was 1984 GB in size during the start of the pool scale. When we copy, we compress the data over the wire, so overall copy was 616 GB bytes copied in 270 minutes so 2.27 GB/minute transfer rate which is typical copy rate. Note that depending on the time of day and networking conditions in the region, etc... the network copy rate can go up and down but typical rate I see is 2GB/minute. I have seen copies as fast as 60 GB/minute when conditions are good (like late at night when regional network is lightly loaded for example).

We are committed to keep pushing scale latency down and improving platform reliability and experience. We do appreciate you as a customer and we understand that you've tied success of your product to success of SQL Azure platform. This is not something we take lightly. Please continue scaling your database to manage cost and meet traffic demands.

How good have you found this content?