Azure SQL Database for Gaming Industry Workloads

Technical Whitepaper

Author: *Pankaj Arora, Senior Software Engineer, Microsoft*

Contents

1 Introduction

Resource demand from the gaming industry is one of the highest growing markets for cloud providers and it is critical that Microsoft Azure and Azure SQL Database offer the fastest performance coupled with the lowest price-to-performance ratios. Gaming customers are choosing Azure SQL Database over virtual machine and in-house alternatives so they can run on the latest stable version of the database engine, get built-in high-availability, automated backups, and geo-replication while being able to easily scale up or out depending on new game launch requirements.

This document briefly details the technical aspect of *why* Azure SQL Database is the best place to run Azure SQL Database for online games. This document is divided into four main sections:

Proven platform with all the latest features. Azure integration, automated backups, high availability with tight integration with Azure Service Fabric.

Stability, including no Microsoft-planned downtime (or failover) and very rare unplanned downtime. Custom maintenance windows are enabled by having isolated tenant rings by customer (separate deployment units).

Best performance and advanced monitoring to detect anomalies. Top support and 24x7 availability from the Azure SQL Database customer support and product team.

Automatic, ongoing future performance improvements to progressively advance hardware architectures, software capabilities and platforms.

2 Proven Platform

Azure SQL Database is an offering that is part of the broader Microsoft Azure eco-system. The Azure Compute, Network, Storage, and SQL Database teams all work very closely together to maximize performance from every module and component interaction.

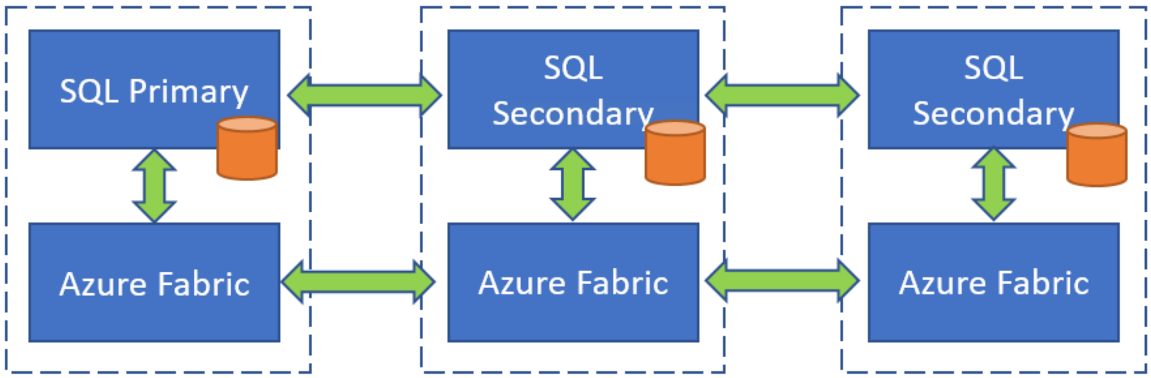
Additionally, the SQL engineering team itself is responsible for operationalizing Azure SQL Database. Azure SQL Database engineers use a 100% DevOps model, owning all functionality and also leading the day-to-day running and management of the service.

2.1 Azure SQL Database as proven platform

Azure SQL Database is a true Platform as a Service (PaaS) offering with many native advantages over the competition.

2.1.1 Azure Service Fabric integration

The Azure SQL Database platform takes responsibility for maintaining High Availability (HA) by tightly integrating with Azure Service Fabric (ASF). ASF allows persistence of data and log files using N-way replication on local SSD disks. We can take advantage of having data and log files on local SSD for maximum possible performance - *without risking a single point of failure*.



Implicit advantages of using ASF:

- Built-in hardware, software, and network failure detection along with automatic leader election.
- Intelligent blocking of operations that could lead to data loss.
- Upgrade coordination, health monitoring and automatic rollback of faulty builds.

2.1.2 Automated Backup and PITR

Azure SQL Database architecture automatically takes care of backups for customer databases. Full backups are taken every week, differential backups every 12 hours, and log backups are performed every 5 minutes. Users can also determine how long these backups are retained. Point-in-time restore is also available as a self-service capability, allowing customers to restore a copy of the database from these backups to any point within the retention period. This is useful both for an unexpected "oops" recovery (e.g. accidental deletion of important data during game upgrade) and for gamer customer support – allowing restores of game play from a particular point in time.

3 Stability using separate deployment units and custom maintenance windows

Databases for high-performance tier customers are maintained separately in their own set of virtual machines. This group of VMs are called a "tenant ring", or "TR" internally. Each TR is managed by Azure Service Fabric and capacity can then be grown or reduced on demand.

Custom Maintenance Window (CMW) takes care of all Microsoft Azure initiated impactful upgrades and related failovers. Our customers, field engineers (customer facing), and the Microsoft Azure (including the Azure SQL Database team) agree on a ~2-hour monthly timeframe for maintenance updates. Azure SQL Database engineers then deploy the latest OS, SQL binaries with new features and bug fixes within this agreed-upon maintenance window. The service is available during these windows but will experience short durations (<5s) where queries need to retry to complete.

Microsoft and the Azure SQL Database engineering team fully owns launching new deployments, the monitoring of in-flight deployments, and the adjusting any automatic rollback due to unhealthy signals. A few guidelines and best practices we follow:

- Azure SQL Database, Service Fabric, and Guest OS updates are batched together whenever possible.

- Host updates are separate (although they can sometimes run in parallel with other updates).
- Each TR is divided into five update domains (or UDs).
- Per UD update time is typically 15 minutes, 50% of the time, within 40 minutes 99% of the time.
- Top-of-rack (TOR) upgrades, including networking related upgrades for Azure VMs, may also occur infrequently (usually once-per-year).
- Customers may experience failovers during the monthly 2-hour maintenance window, thus scheduling the impactful updates during the least impactful (lowest traffic) window will be ideal.

Running on isolated TRs also gives some operational flexibility in terms of optimizing resources. Azure SQL Database has a service running on each node (named *Rgmgr*) which is responsible for managing CPU, memory and disk resources on a node and also governing individual database service resources based on their Service Level Objective (SLO).

3.1 Managing skew across cores and schedulers

There is a known issue that when a process is affinitized to cores in multiple processor groups Windows threads will by default prefer the processor group and NUMA node where the process was initially created (unless the process itself manages affinity of its threads, as is the case for P15s and higher performance tiers). This default is called the “ideal node”. Given this effect, Windows might schedule all SQL database worker threads to cores in one processor group, while cores in another processor group remain idle.

To avoid the load skew across cores and schedulers, *Rgmgr* makes sure a newly placed database is *never* affinitized across different processor groups if the core count of the process (SLO size) is less than the size of processor group.

3.2 Disabling unintended timers

Occasionally there are conditions when the rebalancing timer inside *Rgmgr* has to change the affinity mask of the SQL database process (or move the process to a different set of cores). During this re-affinitization, the core count of the process remains the same, and just the physical cores affinitized to this process are changed. Because CMW restricts changes outside of a specific time frame, we can disable such timers to avoid any impact because of cross NUMA-moves.

4 Best Performance and Monitoring

The gaming industry is very sensitive to transaction throughput and individual transaction commit-latency. Currently there is ~1ms latency from client VM to SQL transaction processing and back. This section covers what Azure SQL Database does to ensure low latency and transactional resilience. *The following sections describe specialized native capabilities and settings which we apply to high-performance gaming customer environments.*

4.1 Best engineers, 24 x 7

Gaming customers running on high-performance tiers receive top priority treatment and 24x7 support directly from the Azure SQL Database support and engineering team. The Azure SQL Database engineering team is automatically notified via alerts and

4.2 Login governance

We keep a separate thread pool (including CPU and memory resources) specifically for processing logins. This makes sure availability is never affected and there are always reserved resources to process incoming logins. Azure SQL Database uses a weighted round-robin algorithm based on schedulers in the NUMA node.

4.3 Affinity type

As described in the previous section, if the processor can always fit in a single processor group, *RgMgr* guarantees it will allocate it accordingly. For the P15 service tier, as well as databases with 20 vCores and higher, where it is not possible to fit all processors on a single processor group, we made Azure SQL Database run with a hard affinity so that SQL intentionally configures affinities of worker threads to use CPU's across *both* processor groups. "Hard affinity" is the mode where SQL database enumerates all the cores on a node and creates a SOS scheduler for each core. The schedulers/cores which are not in SQL's affinity mask are marked offline.

For small SKU's like P6, this can result in offline NUMA nodes, which can cause further issues, so we avoid hard affinity configurations for SKU's which can fit in a single processor group. These SKU's run with an agnostic affinity configuration. This allows Windows to migrate our threads across cores based on actual consumption if needed. *This configuration has produced more predictable latencies based on benchmark testing and customer feedback*. This affinity policy provides better workload performance and predictability.

4.4 Faster transaction commits and high commit throughput

Transaction log commits are not throttled or governed. This allows higher throughput for gaming workloads. We also disable automatic log truncation in order to avoid contention for active user transactions. Again, this is just to help customer workload, and we still shrink logs as part of recurring transaction log back-up and checkpoint processes.

4.5 Backup improvements

We govern CPU resources available for database backups to ensure that customer workloads always run at the top priority and get as many resources as they can. We also can customize differential backup schedules based on customer workload reduced-traffic periods.

4.6 Elastic IO Resource Governance

For Premium-tier and Business-critical-tier gaming databases, we relax IO-related resource governance for both reads and write activity. This allows an Azure SQL Database workload to reach node-level SSD limits during high peak times.

4.7 No idle session timeout

We let idle sessions persist as long as they have to (*we do not force disconnections*).

4.8 High priority monitoring

We prioritize login failure and any customer-configured alerts to ensure surfacing in less than 5 minutes of event-firing.

monitoring described earlier. Customers can also reach us directly for any guidance needed regarding workload throughput or latency. [1]

5 Future-proofing via the Azure SQL Database roadmap

As of this writing, here are a few of the improvements planned for Azure SQL Database that will be applicable to gaming workloads and characteristics:

- Full SQL Server compatibility via the Azure SQL Database Managed Instance offering
- Continuously Improving PaaS service:
  - Azure SQL Database Hyperscale (flexible storage with capability to grow data up to 100TB).
  - Usage based Billing (via an upcoming “Serverless” offering)
  - Today CMW duration is programmed into system by an Azure SQL Database engineer. In the future we will support automated configuration and monitoring.
- We are exploring global CPU governance offered by Windows job objects for databases on high performance TRs. This enables more parallelism without the need for affinity management. Early testing of this change shows a 20-30% gain in performance numbers.
- Introducing Gen 6 hardware with NVMe direct and RDMA support, 4M IOPS and 500 us round-trip latencies. The below table specifies the anticipated latencies:

Transaction Commit (time component)	***Gen5 with Accelerated Networking***	***Gen6 + Direct NVMe + RDMA +*** *Proximity zones*
Client VM to/from SQL	300us	75us
SQL primary update (CPU + Disk)	300us	300us
Primary to/from Secondaries	300us	15us
Secondary log commit (disk)	100us	100us
Total time	**1000us**	**490us**

[1] Most of the settings, monitoring and alerts described above are now applicable to other expanded offerings from Azure SQL Database (particularly on high offers like P15 or 40+ Vcore). In general, our principle has been to safely apply all the learnings from maintaining Azure SQL Database for gaming industry workloads to other offerings across the Azure SQL Database where-ever possible.