SQLSATURDAY

South Florida 2023 #1053

# **Optimize Your SQL Database**
# Tips for Peak Performance

**Darius Liktorius**

Principal Architect & Engineering Fellow

Cognizant

#SQLSatSoFla

# Darius Liktorius

**Principal Architect**

Cognizant

**Liktorius.com**

**@DLiktorius**

**linkedin.com/in/DariusLiktorius**

Over 25 years of experience with SQL Server:

- Architect at Cognizant
- Specializing in scalability, availability and performance
- Co-Founder of Microsoft Cloud South Florida User Group

SQLSATURDAY
South Florida 2023 #1053

## Agenda

- Overview
- Application Usage Patterns
- Infrastructure & Storage
- Partitioning, Indexes & Statistics
- Replicas & Sharding
- Q & A

## Overview

## Overview – What's Covered

- We are covering:
  - Critically Important Design & Deployment Decisions
  - Essential Tools & Maintenance Operations
- We are <u>not</u> covering:
  - Troubleshooting & Optimizing Queries – enable Query Store!
  - Usage of Diagnostics Tools

## Overview – Stereotypes

- Application Developers:
  - Are not bad people!
  - Leverage effort-reducing libraries
  - Do not appreciate impacts against DB
- DBAs, DevOps & Architects:
  - Rightfully question application code
  - *Sometimes* make critically important design oversights

## Tools to Consider

- Database Engine Tuning Advisor
- Azure SQL Database – Automatic Tuning
- Third-Party:
  - SQL Sentry by SentryOne (Solarwinds)
  - Quest Foglight on SQL
  - Idera SQL Diagnostic Manager
  - SQL Grease

## Application Usage Patterns

## Application Usage Patterns

- Over-normalization
- Object-Relational Mapping (ORM) Libraries
    - Lazy-loading, Loops, Unnecessary Joins
    - Evaluate actual queries
    - Use DTOs
- Connection Pooling & Disposal

## Application Usage Patterns – cont’d.

- Areas of contention:
  - Active Tables
  - Table “Hot” Spots
- Cache, Cache and Cache some more!
  - Redis
  - Memcached

## Azure SQL DB – Transient Faults (EF Core)

```
// Startup.cs from any ASP.NET Core Web API
public class Startup
{
    // Other code ...
    public IServiceProvider ConfigureServices(IServiceCollection services)
    {
        // ...
        services.AddDbContext<CatalogContext>(options =>
        {
            options.UseSqlServer(Configuration["ConnectionString"],
            sqlServerOptionsAction: sqlOptions =>
            {
                sqlOptions.EnableRetryOnFailure(
                maxRetryCount: 10,
                maxRetryDelay: TimeSpan.FromSeconds(30),
                errorNumbersToAdd: null);
            });
        });
    }
//...
}
```

## Infrastructure

## Infrastructure – Self Managed

- Virtual Machine / Bare Metal:
  - Sufficient CPU allocation for the load – T.B.D.
  - Memory (RAM) to host frequently used tables, active partitions and indexes
  - Storage RAID Level:
    - 1 (SSD) or 10 (HDD) for Transaction Log and Tempdb
    - 5, 6, 50, 60, or 10 for Database

## Infrastructure – Cloud Hosted

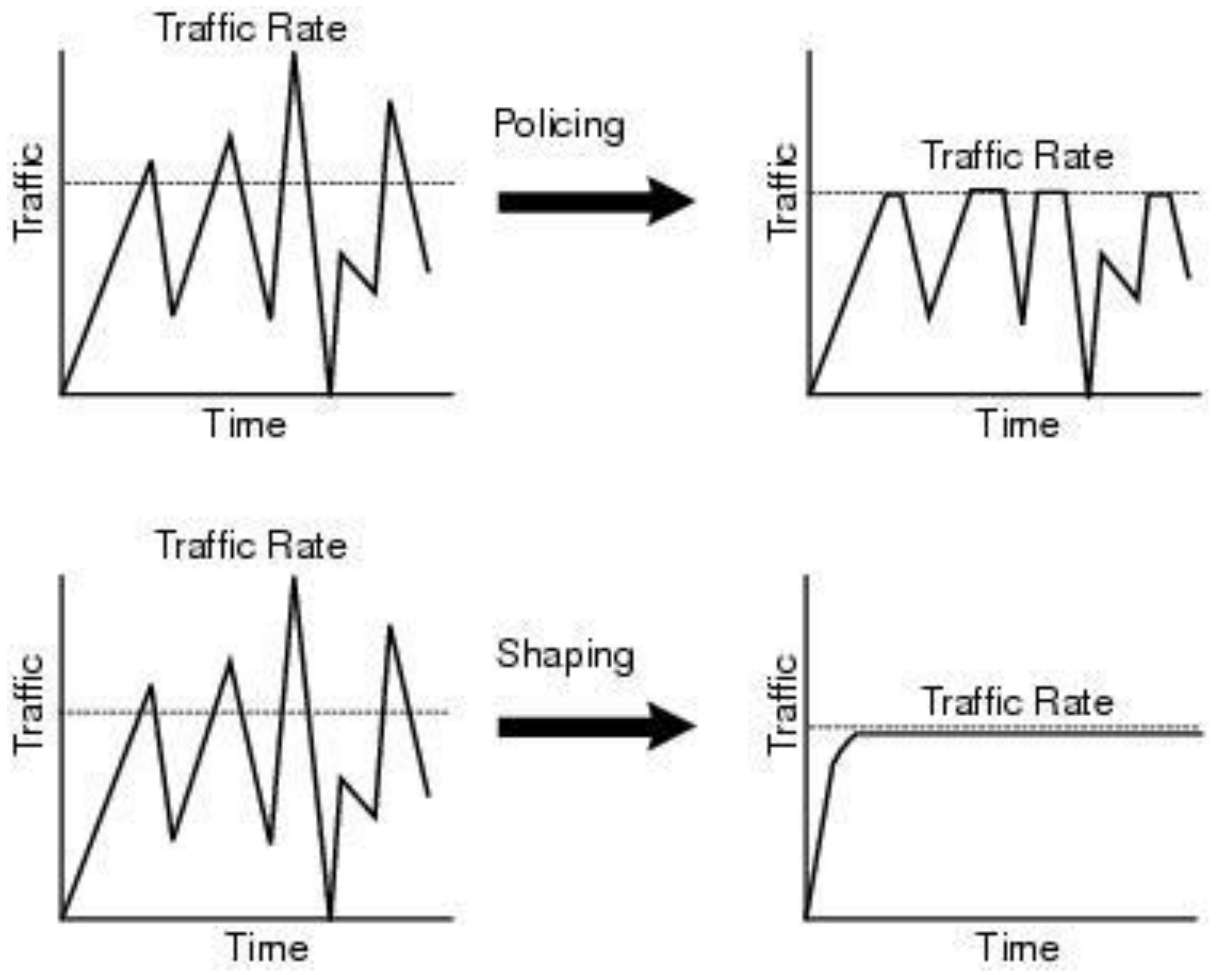
- Tiers, SKUs:
  - Affect CPU, Memory (e.g., E-Series), Disk (e.g., ephemeral)
  - SSD vs HDD & Throughput
  - Block vs Blob storage in Cloud (more later)
- Networking:
  - VNET integration, Service Endpoints, Private Link, etc.

## Storage

## Storage for SQL Server

- Why should I care?
- SQL Server is sensitive to disk latency
  - Optimal latency for database: **<= 10ms**
  - Optimal latency for transaction log: **<= 2ms**

# Network Throttling - Policing vs Shaping



# Azure Storage Architecture

## Disks
Persistent disks for Azure IaaS VMs

Premium Storage Disks option:
SSD based, high IOPS, low latency

## Files
Fully Managed File Shares in the Cloud

SMB and REST access

"Lift and shift" legacy apps

## Blobs
Highly scalable, REST based cloud object store

Block Blobs: Sequential file I/O

Page Blobs: Random-write pattern data

## Hybrid
Azure File Sync

StorSimple

## Built on a unified Distributed Storage System
Durability, Encryption at Rest, Consistent Replication, Fault Tolerance, Load-Balancing

# Storage Comparison

## Azure

- **Shared** Infrastructure
- Throttling – **choppy** (*Network Policing*)
- Ethernet Storage (iSCSI)
- SQL Database & M.I. in **Standard/GP Tiers –** ***overcome with BC & HS***
- **Multiple HA Options**
- VMs: ***Use Storage Pools***

## AWS

- **Dedicated** Infrastructure
- Throttling – **smooth** (*Traffic Shaping*)
- True **Block Storage**
- Also used by Amazon RDS
- ***Limited HA*** – Local AZ only – *Like Azure LRS*

## GCP

- **Dedicated** Infrastructure
- Throttling – **smooth** (*Traffic Shaping*)
- True **Block Storage**
- Also used by Cloud SQL
- **Multiple HA** – Local AZ, Multi-AZ, Cross-Region

# Extreme Performance Storage Comparison

## Azure

### Ultra Disk

- **<u>Dedicated</u>** Infrastructure
- **<u>Block</u>** Storage (for VMs)
- **Fast** – Up to 160k IOPS or 4,000 MB/sec
- Throttling – VM and Disk but **<u>smooth</u>** (Shaping)
- Redundant Storage (LRS and ZRS) – Varies by Region

## AWS

### io2 Block Express

- Dedicated Infrastructure
- Block Storage
- **Fastest** – Up to 256k IOPS or 7,500 MB/sec
- Throttling – VM and Disk Smooth (Shaping)
- **Local-Zone Redundancy** only

## GCP

### Extreme Persistent Disks

- Dedicated Infrastructure
- Block Storage
- **Slowest** – Up to 120k IOPS or 2,200 MB/sec
- Throttling – Smooth
- **Local-Zone Redundancy** only

## Local SSD Storage

- **Ephemeral** (Transitory) – Not persistent
- Azure, AWS and GCP **all have Local SSD options**
- **USE THEM!**

## File Placement – for VM / On-Premises

- Separate Log & Data File Locations
- Utilize File Groups (FG’s)
- Split Tables and Non-Clustered Indexes into separate FG’s
- Consider dedicated FG for very large tables

## Partitioning, Indexes & Statistics

# Partitioning

- **Physically separates data** based on criteria (e.g., date ranges)
- **Reduces or eliminates** cross-query data page **locking**
- Allows for **efficient** management & **deprecation of data**

## Indexes

- **Clustering approach:** consider de-coupling Primary Key from Clustered Column(s)
- **Fill Factors:** Unless contiguously inc/dec-rementing values (e.g., Identity Columns), *always* specify a Fill Factor < 100
- **Maintenance:** Ensure you are regularly (nightly, intra-day) reorganizing and/or rebuilding your indexes!
  Don’t forget about statistics!

# Table Statistics “STATS”

- **Critically Important** – has direct impact on index selectivity
- **Rate of Change** – Will not update unless >=30% of delta
- **Best Practices** –
  - Keep **auto-update enabled**, <u>but run nightly</u>
  - Consider **specific tables** for one-off updates
  - Utilize **async** update (e.g., large tables w/ frequent, big updates)

## Replicas & Sharding

## Replicas

- Provide **High Availability & Scalability**
- Enabled via Availability Groups & DAGs
- Azure SQL Database Hyperscale – adds Named Replicas
- **Synchronous vs Asynchronous**
- **Readable** – connection string: “applicationIntent=readonly”

## Sharding

- Divide a data store into a set of horizontal partitions or shards. This can improve scalability when storing and accessing large volumes of data.
- Azure SQL Database - Elastic Database Client Library

## Q & A

## Thank you

### Presentation Landing Page & Resources:

**Liktorius.com/go/SQLSAT1053**

**Darius Liktorius**

@DLiktorius

linkedin.com/in/DariusLiktorius

#SQLSatSoFla

SQLSATURDAY
South Florida 2023 #1053