

# SQL Server on Azure

## Best Practices für den DB Server in der Cloud

Andre Essing  
Technology Solutions Professional  
Data Platform





**Andre Essing**  
Technology Solutions Professional  
Microsoft Deutschland GmbH

Andre advises, in his role as Technology Solutions Professional, customers in topics all around the Microsoft Data Platform. He is specialized in mission critical systems, high-availability, security, operating and of course the cloud.

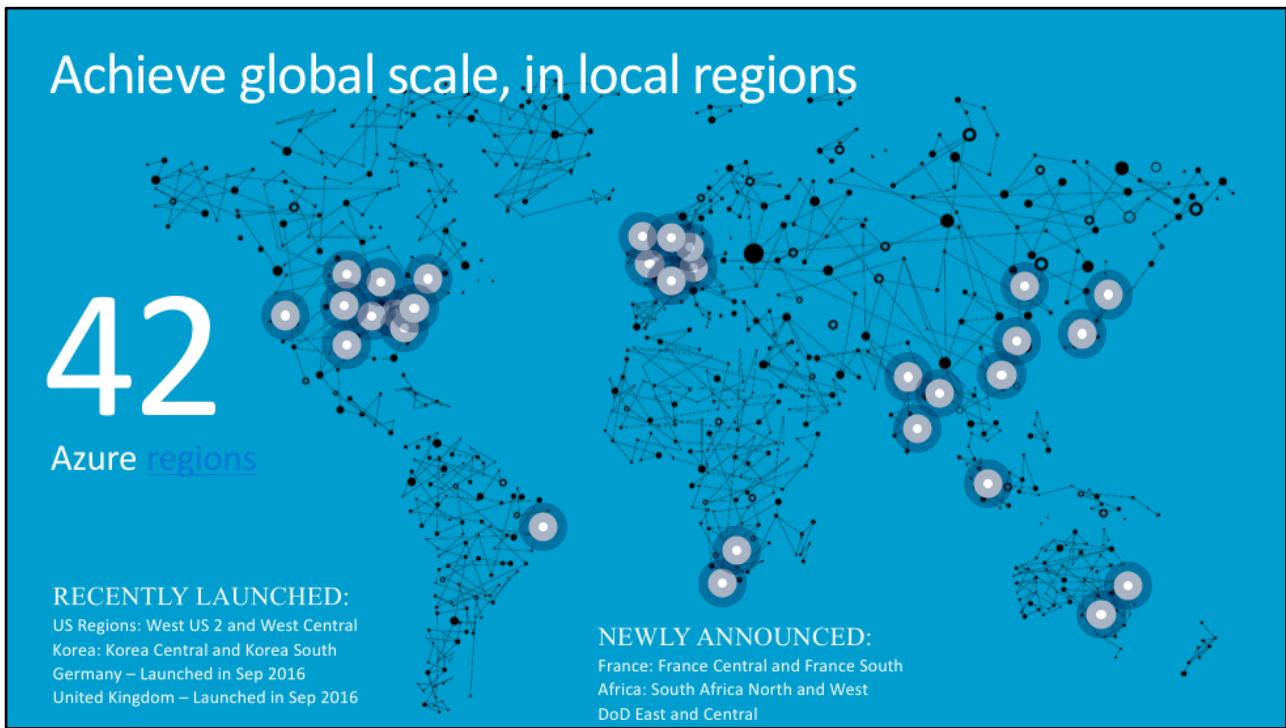
 [andre.essing@microsoft.com](mailto:andre.essing@microsoft.com)

 [Andre\\_Essing](#)

 [aessing](#)

 [@aessing](#)

 [aessing](#)



## How hyperscale cloud works



20.08.17 15:32



20.08.17 15:32



## SQL Server on Azure - Welche Möglichkeiten gibt es?

SQL Server in einer VM  
(IaaS)



Azure SQL Database  
(PaaS)



## SQL Server on Azure - Welche Möglichkeiten gibt es?

SQL Server in einer VM  
(IaaS)



Azure SQL Database  
(PaaS)



## Cloud Dienste – Verantwortlichkeiten



On-Premises	Infrastructure (as a Service)	Platform (as a Service)	Software (as a Service)	
Applikationen	Applikationen	Applikationen	Applikationen	Kundenumgebungen werden auf der Azure Infrastruktur voneinander isoliert
Daten	Daten	Daten	Daten	Gemeinsam genutzte physikalische Umgebung
Runtime	Runtime	Runtime	Runtime	Verwaltung durch:
Middleware	Middleware	Middleware	Middleware	Kunde
Betriebssystem	Betriebssystem	Betriebssystem	Betriebssystem	Anbieter
Virtualisierung	Virtualisierung	Virtualisierung	Virtualisierung	
Server	Server	Server	Server	
Storage	Storage	Storage	Storage	
Netzwerk	Netzwerk	Netzwerk	Netzwerk	

← Microsoft Azure →

## Agenda

- Storage
- Virtuelle Maschinen
- SQL Server
- Sicherheit



# Storage

## Storage Accounts und Disks

- Standard oder Premium Storage Accounts
- Nur Premium Locally Redundant Storage (P) oder Locally Redundant (L) verwenden
  - Georeplikation führt zu korrupten Datenbanken
- Disk Caching Policy anpassen
- 4TB ist aktuell die maximale Größe einer virtuellen Disk
- Managed Disks verwenden



### Standard oder Premium Storage Accounts

Azure bietet je nach Anforderung Standard oder Premium Storage Accounts an. Dabei basieren Standard Storage Accounts auf handelsüblichen drehenden magnetischen Datenträgern (HDD) und Premium Storage Accounts auf der neuesten Solid State Drive Technology (SSD).

Im Vergleich können Standard Storage Accounts maximal 500 IOPS pro Disk, zu maximal 7500 IOPS pro Disk im Premium Storage abbilden. Eine einzelne Virtuelle Maschine kann somit eine maximale Größe von 256TB Storage und bis zu 80.000 IOPS (2.000MB/s) bei geringen Latenzen erreichen. Standard Storage Accounts haben momentan 16.000 IOPS als Maximum.

### Nur Premium Locally Redundant Storage (P) oder Locally Redundant (L) oder verwenden

Da bei aktivierter Georeplikation, auf Grund der asynchronen Replikation, ein konsistentes schreibverhalten über mehrere virtuelle Disks nicht garantiert werden kann, muss Georeplikation (GRS) zwingend deaktiviert werden. Auf Grund der asynchronen Replikation kann Georeplikation ansonsten zu korrupten Datenbanken führen. Hochverfügbarkeitsszenarien zur Absicherung der Daten müssen deshalb mit SQL Server Boardmitteln (Availability Groups,

Mirroring, oder Transaction Log Shipping) realisiert werden. GRS darf nur genutzt werden wenn Daten- und Transactionlog-Dateien auf einer Disk liegen.

## Disk Caching Policy anpassen

Bei Disks innerhalb eines Premium Storage Accounts sollte die Disk Caching Policy auf „ReadOnly“ gesetzt werden. Dies gilt jedoch nur für Disks auf denen Datenbankdateien oder die TempDB abgelegt werden. Bei Disks auf denen die Transactionlogdateien liegen muss die Disk Caching Policy auf „None“ stehen. Hostet man die SQL Server Disks hingegen auf einem Standard Storage Account, so muss die Disk Caching Policy immer auf „None“ stehen.

Ausgenommen hiervon ist die Disk mit dem Betriebssystem, hier steht die Policy immer auf „ReadWrite“. Aus diesem Grund sollten niemals Datenbankdateien, auch nicht die der Systemdatenbanken, auf dem Betriebssystem-Datenträger abgelegt werden.

Zum anpassen der Policies kann man die PowerShell Commandlets Set-AzureDataDisk und Set-AzureOSDisk verwenden.

## 4TB ist aktuell die maximale Größe einer virtuellen Disk

Virtuelle Disks werden in Storage Accounts als Page-Blobs abgelegt. Das ermöglicht momentan ein Maximum von 4TB pro virtueller Disk. Somit können VMs maximal 256TB hosten.

Die aktuellen Limits der Azure Storage Accounts sind hier <https://azure.microsoft.com/de-de/documentation/articles/azure-subscription-service-limits/> dokumentiert.

## Demo

- Anmeldung an Azure mittels PowerShell Azure CLI
- Erzeugen einer Ressource Gruppe
- Erstellen einer Managed Disk
- Erstellen von Storage Accounts



## Temporärer Storage

- Laufwerk D: ist temporärer Storage
  - Wird nicht wie normaler Azure Blob Storage persistiert
  - Daten sind nach einem VM Neustart unwiderruflich gelöscht
  - Nicht für Datenbank- oder Transactionlogdateien geeignet
- Optimal für TempDB und Bufferpool Extensions
  - Nur bei VMs mit lokaler SSD (D, DS, GS)
  - Erfordert Scripting



### **Laufwerk D: ist temporärer Storage**

Jede virtuelle Maschine auf Azure besitzt ein Laufwerk D:, das auf lokalen Storage auf dem Azure VM Host zugreift. Ein ganz wichtiger Punkt hierbei, das Laufwerk ist ein temporäres Laufwerk auf dem die Daten nicht wie im Azure Blob Storage persistiert werden. Dies bedeutet, dass wenn die VM neu gestartet wird oder durch einen Ausfall des Hosts auf einen anderen Host verschoben wird, sind die zuvor auf dem temporären Laufwerk gespeicherten Daten unwiederbringlich verloren. Deshalb ist das Laufwerk zur Ablage von Datenbank- oder Transactionlogdateien nicht geeignet.

Microsoft legt hier zum Beispiel die Windows Auslagerungsdatei ab, da diese keine Daten enthält und bei jedem Systemneustart neu erzeugt wird.

### **Optimal für TempDB und Bufferpool Extensions**

Basiert das temporäre Laufwerk D: auf einer SSD, wie bei den VM Klassen D, DS und GS, dann kann es durchaus verwendet werden um die TempDB oder eine Bufferpool-Extension des SQL Servers abzulegen. Bei intensiver Nutzung der TempDB durch eine Applikation kann dies eine Performancesteigerung mit sich bringen. Von einer Verlagerung der TempDB oder der einer Anlage von Bufferpool-Extensions auf einem temporären Laufwerk ohne Solid State Drive

Technologie ist jedoch abzuraten, da die Performance des temporären Laufwerks nicht mit dem Azure Blob Storage mithalten kann.

Bei der Nutzung des Volumes für TempDB und Bufferpool-Extensions ist es erforderlich ein Skript bei Serverstart auszuführen, welches eine entsprechende Ordnerstruktur anlegt und im Anschluss daran die SQL Server Dienste startet. Siehe auch  
<http://blogs.technet.com/b/dataplatforminsider/archive/2014/09/25/using-ssds-in-azure-vms-to-store-sql-server-tempdb-and-buffer-pool-extensions.aspx>.

## Disks optimal einsetzen

- Datenbank- und Transactionlogdateien
  - auf unterschiedlichen Disks ablegen
  - nicht auf der Betriebssystemdisk ablegen
- Verwendung von Stripesets um IOPS zu erhöhen
  - 64KB Stripe Size für OLTP
  - 256KB Stripe Size für DWH
  - Column Count konfigurieren
- Disks mit NTFS und 64KB Cluster Size formatieren



### Datenbank- und Transactionlogdateien

Um eine optimale Performance und Datensicherheit zu erreichen, dürfen die Transactionlogdateien nicht zusammen mit den Datenbankdateien auf einer Disk abgelegt werden. Auch die Lagerung von MDF, NDF und LDF Dateien auf dem Betriebssystemdatenträger ist nicht zu empfehlen. Davon mal ab, dass eine Lagerung der Datenbankdateien zusammen mit dem Betriebssystem die Performance der Datenbank vermindert, verhindert auch das aktivierte Read und Write Caching auf der OS Disk einen optimalen Datenbankbetrieb. Dies gilt ebenso für die Dateien der Systemdatenbanken, wie auch für die Nutzerdatenbanken.

### Verwendung von Stripesets um IOPS zu erhöhen

Windows ermöglicht es, zum Beispiel in Windows Server 2012 R2 mit Storage Spaces, mehrere Disks zu einem Stripeset zusammen zu fügen. Sollte die Kapazität oder der Durchsatz (IOPS) einer einzelnen Disk nicht für die Workload des SQL Servers ausreichen, so kann man problemlos mehrere Disks zu einem Stripeset zusammenfügen. Die virtuellen Maschinen in Azure besitzen jeweils eine Grenze der maximal nutzbaren IOPS, die zu beachten ist.

Um bei der Erstellung eines Stripesets eine optimale Performance zu erreichen, sollte die Stripe Size beim erstellen des Stripesets auf 64KB für eine OLTP Workloads und auf 256KB für ein Datawarehouse konfiguriert werden. Hinzu kommt das der Column Count des Stripesets auf die Anzahl der im Stripeset enthalten Disks eingestellt werden sollte, um optimales Lastverhalten des Stripsets zu erzielen.

### **Disks mit NTFS und 64KB Cluster Size formatieren**

Was bei On-Premise Systemen gut ist, muss bei Cloud Systemen nicht schlecht sein. So sollten auch bei SQL Servern auf Azure die Daten- und Transactionlog Disks mit NTFS und einer Clustersize von 64KB formatiert werden um auch hier eine optimale Performance zu erreichen.

## Datenbanken direkt im Azure Blob Storage

- Keine Einschränkungen beim resizen von VMs
- Einfache Migration zwischen VMs
- Datenbank- und Transactionlogdateien werden als Page Blobs im Storage Account abgelegt
- Azure Blob Storage supported aktuell kein
  - In-Memory OLTP (Hekaton)
  - File Streaming
- AGs sind supported, mit Einschränkungen
- Nur mit Standard Storage Accounts nutzbar



Seit Version 2014 bietet der SQL Server die Möglichkeit Datenbank- und Transactionlogdateien direkt auf einem Azure Blob Storage Account abzulegen.

### Keine Einschränkungen beim resizen von VMs

Azure VMs haben neben Ihren Eigenschaften wie CPU und Memory auch eine Einschränkung der maximalen Disk-Anzahl. Dies hat den Nachteil, dass evtl. wenige Cores und wenig RAM für den SQL Server zwar völlig ausreichen, die Maximale Anzahl der Disks jedoch nicht. Auch ein Downsizing einer VM mit z.B. 8 Disks ist nur bis zu einem bestimmten VM Größe möglich. Benutzt man jedoch Azure Blob Storage als direkte Ablage der Datenbank-Dateien, so kann man ein System frei vergrößern und verkleinern und ist nicht von den Beschränkungen der Disks abhängig.

### Einfache Migration zwischen VMs

Eine Migration einer Datenbank von einem SQL Server zum Nächsten wird mit Storage Accounts ebenfalls vereinfacht. Man muss hier keine Dateien von einer VM in eine andere kopieren. Man kann die Datenbank auf dem alten

Server einfach abhängen, den neuen Server mit dem Storage Account verbinden und die Datenbank auf dem neuen Server einhängen. Dies erleichtert und beschleunigt die Migration ungemein.

### **Datenbank- und Transactionlogdateien werden als Page Blobs im Storage Account abgelegt**

Da Datenbankdateien auf einem Azure Blob Storage Account als Page Blobs gespeichert werden, liegt das Limit pro Datei bei 1TB. Dazu kommt, dass jede Datei einen Durchsatz von 60MB pro Sekunde oder bis zu 500 IOPS liefern kann.

### **Azure Blob Storage supported aktuell kein**

Momentan ist es nicht möglich einen Storage Account als Datenablage für FileStreaming oder InMemory OLTP zu verwenden. Hierzu müssen weiterhin Disks verwendet werden.

### **AGs sind supported, mit Einschränkungen**

Generell ist es möglich Availability Groups mit direkt im Storage Account abgelegten Datenbankdateien zu nutzen. Jedoch gibt es hier die folgenden Einschränkungen:

1. Das Hinzufügen von Datenbankdateien zu einer Datenbank die an einer Availability Group teilnimmt ist nicht möglich. Hierzu muss die Datenbank erst aus der Availability Group genommen werden, bevor das Hinzufügen einer Datei möglich ist.
2. Der Wizard zum erstellen einer Availability Group bzw. zum aufnehmen einer DB in die Availability Group kann nicht verwendet werden. Vielmehr muss das Backup und der Restore der Datenbank, zur Aufnahme in die Availability Group, manuell erfolgen.

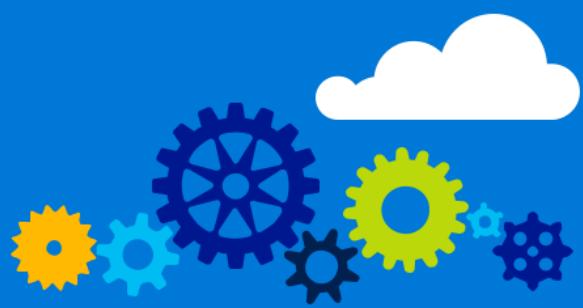
### **Nur mit Standard Storage Accounts nutzbar**

Premium Storage Accounts sind aktuell nur mit virtuellen Disks nutzbar. Eine direkte Ablage von Datenbankdateien ist somit nur in einem Standard Storage Account möglich. So ist es nicht möglich die Performancevorteile des Premium

Storage Accounts zusammen mit Datenbanken im Azure Blob Storage zu nutzen.

## Demo

- Erstellen einer DB auf einem Azure Storage Account



# Virtuelle Maschinen

## Sizing einer VM

- Empfehlung für SQL Server VMS
  - DS3 für SQL Server Enterprise Edition
  - DS2 für SQL Server Standard Edition
- VM Size nach Workload
  - Bestehende Systeme vorher monitoren
  - CPU, Arbeitsspeicher und IOPS
  - Up- and Downsizing (fast) immer mit Reboot möglich



### Empfehlung für SQL Server VMS

Eine Empfehlung zur optimalen Größe von VMs mit installiertem SQL Server.

- VM Klasse DS3 für SQL Server Enterprise Edition
- VM Klasse DS2 für SQL Server Standard Edition

Diese Empfehlung wird als Einstiegsgröße für einen performanten SQL Server empfohlen.

### VM Size nach Workload

Generell sollte die Größe der VMs nach der Workload gewählt werden, nicht nach der Edition. Bei bestehenden Systemen sollte man deshalb zuvor ein Performance Monitoring durchführen um die wirklichen Anforderungen an die VM zu ermitteln. Die wichtigsten Parameter hierzu sind CPU, Arbeitsspeicher und IOPS. Letzten Endes das gleiche Verhalten, bevor man physische Maschinen On-Premise virtualisiert.

Sollte man diese Möglichkeit nicht haben, da man zum Beispiel eine VM für

eine neue Workload sized und keine brauchbaren Vorgaben zur Verfügung stehen, sollte man im Hinterkopf behalten, dass ein Resizing einer VM auf Azure mit einem Reboot möglich ist.

## Demo

- Erstellen eine SQL Server VM auf Azure



## Eine performante SQL Server VM...

- ...sollte in einer Azure Region nahe der Workload erstellt werden
  - Hält Latenzen gering
- ...muss zusammen mit den Storage Accounts in der gleichen Region gehostet werden
  - Hält Latenzen gering
  - Vermeidet zusätzliche Kosten
  - Managed Disks verwenden
- ...darf nicht automatisch skalieren
  - VM Scale Sets



### **...sollte in einer Azure Region nahe der Workload erstellt werden**

Azure bietet eine Vielzahl von Regionen zur Auswahl um VMs zu erstellen. Um nun die Latenzen im Netzwerk so gering zu halten wie möglich, sollte man die Region für seine VM wählen, dass am nächsten zum Standort ist, an dem die Daten benötigt werden. Für Deutschland wäre das zum Beispiel West Europe (Amsterdam).

### **...muss zusammen mit ihren Storage Accounts in der gleichen Region gehostet werden**

Eine VM sollte in der gleichen Azure Region gehostet werden wie auch die Storage Accounts der VM. Dadurch werden die Latenzen der I/Os gering gehalten. Darüber hinaus werden zusätzliche Kosten vermieden, da kein ausgehender Traffic generiert wird.

### **...darf nicht automatisch skalieren**

Azure bietet die Möglichkeit Cloud Services automatisch zu skalieren. So wird, wenn die aktiven VMs überlastet sind, eine weitere VM hochgefahren und die

Last verteilt. Genauso wird die VM wieder heruntergefahren, wenn die Last unter einen Schwellwert sinkt. Dies funktioniert bei zustandslosen Servern, wie zum Beispiel bei Webservern sehr gut. Bei einem zustandsbehaftetem Dienst, wie z.B. ein Datenbankserver funktioniert dies nicht.

## Demo

- Hinzufügen einer zusätzlichen Disk



# SQL Server

## SQL Server aus dem Marketplace

- Aufräumen! Aufräumen!! Aufräumen!!!
- Standardmäßig kein Zugriff über Netzwerk
  - Nur Shared Memory aktiv
  - TCP/IP muss manuell aktiviert werden
- SQL Server Default Instance installiert
  - Weitere Instanzen können installiert werden
  - Sourcen liegen auf Betriebssystemdisk
- Customer Experience Improvement Program



### **Aufräumen! Aufräumen!! Aufräumen!!!**

Eine SQL Server VM auf Azure kann man aus bereits vorbereiteten Images erzeugen. Dies hat zwei große Vorteile, man muss den SQL Server nicht manuell installieren und viel wichtiger, die SQL Server Lizenz wird mitgeliefert und Minuten genau abgerechnet. Das heißt, ist die SQL Server VM heruntergefahren, fallen auch keine Lizenzkosten an.

Es gibt aber auch ein Problem mit den Images, es sind alle Komponenten des SQL Servers installiert. Bevor man also ein Platform Image nutzen kann, sollte man um die Performance und Stabilität zu steigern, sowie die Sicherheit zu erhöhen, alle nicht benötigten Komponenten deinstallieren. Auch das installieren der aktuellsten Updates ist sinnvoll.

### **Standardmäßig kein Zugriff über Netzwerk erlaubt**

Bei einem SQL Server der aus einem Platform Image erstellt wurde, ist standardmäßig nur Shared Memory als Zugriffsprotokoll aktiviert. Dies verhindert den Zugriff auf den SQL Server von aussen und erlaubt nur Zugriffe vom Host selbst. Um nun den Zugriff über das Netzwerk zu aktivieren, muss man im SQL Server Configuration Manager das TCP/IP Protokoll für die Instanz aktivieren und den Dienst neu starten.

## **SQL Server Default Instance installiert**

Die SQL Server Instanz die in einem SQL Server Platform Image vorinstalliert mitkommt, ist eine Default Instanz (MSSQLSERVER). Es ist möglich, weitere Instanzen auf der gleichen VM zu installieren. Die Installationssourcen liegen auf der Betriebssystemdisk C:\.

## **Customer Experience Improvement Program (CEIP) aktiviert**

Wie viele Hersteller nutzt auch Microsoft die Möglichkeit über seine Software Nutzungsinformationen zu sammeln. Generell ist dies erstmal nicht tragisch, da die Nutzungsdaten anonymisiert übermittelt werden. Bei den Platform Images ist diese Funktion standardmäßig aktiv. Selbstverständlich kann man das Sammeln der Daten deaktivieren.

## SQL Server Service Konfiguration

- SQL Server Dienste mit Service Accounts betreiben
- Service Account des SQL Server Dienstes berechtigen
  - Perform Volume Maintenance Tasks
  - Lock Pages in Memory
- Trace Flags setzen
  - -T1117 (Ab SQL Server 2016 Filegroup-Option)
  - -T1118 (Ab SQL Server 2016 Datenbank-Option)
  - -T3226
- Windows Power Plan auf High Performance



### SQL Server Dienste mit Service Accounts betreiben

Um die Sicherheit des Systems weiter zu erhöhen, sollte ein SQL Server nie mit „Local System“ oder Ähnlichem als Service Account betrieben werden. Diese Accounts haben in der Regel von Haus aus zu viele Berechtigungen. Hier sollte man einen eigenen Service Account anlegen, der so wenig Rechte wie möglich besitzt.

### Service Account des SQL Server Dienstes berechtigen

Es gibt zwei Berechtigungen im Windows Betriebssystem, welche für den Betrieb des SQL Server zwar nicht zwingend erforderlich sind, dennoch einen riesigen Vorteil bringen wenn man den Service Account entsprechend berechtigt.

- Mit dem Recht „Perform Volume Maintenance Tasks“ kann der SQL Server Instant File Initialization nutzen. Bedeutet, dass wenn der SQL Server eine Datenbankdatei erstellt, eine Datenbankdatei vergrößert oder ein Restore durchführt, kann der Datenbankdienst den Speicherplatz schnell und einfach allokieren. Vergibt man dieses Recht nicht, muss der SQL Server den neu allokierten Speicherplatz zuerst mit Nullen überschreiben, was Zeit kostet und viele IOPS erzeugt. Dies gilt nicht für

Transactionlogs.

- Das Recht „Lock Pages in Memory“ erlaubt dem SQL Server Dienst seine Bereiche im Arbeitsspeicher zu sperren. So kann Windows die Speicherseiten des SQL Servers nicht aus dem RAM in das Pagefile auslagern. Dies sorgt nicht nur für bessere Performance des SQL Servers, sondern verringert auch die IOPS.

### **Trace Flags setzen**

Trace Flags sind Konfigurationsparameter die ein grundlegendes Verhalten des SQL Servers ändern. Generell sollte man Trace Flags nicht zu leichtfertig setzen, es gibt jedoch 3 Trace Flags die immer ohne Bedenken aktiviert werden können.

- -T1117 – Erlaubt es dem SQL Server immer sämtliche Dateien einer Filegroup zusammen zu vergrößern (Autogrowth). Dies sorgt dafür das die Parallelität die durch das Aufteilen von Datenbanken auf mehrere Dateien erreicht wird erhalten bleibt.
- -T1118 – Steigert die Performance der Datenbank durch die Allokation von Unified Extents. Dies verringert die Contention auf den Datenbanken.
- -T3226 – Dieser Parameter verhindert das Meldungen zu erfolgreichen Backups ins SQL Server Errorlog geschrieben werden. Das Log wird dadurch übersichtlicher.

### **Windows Power Plan auf High Performance**

Auch in einer VM versucht Windows Strom zu sparen und nutzt die vom Hypervisor zur Verfügung gestellten Funktionen. Um dies zu verhindern und die volle Performance der VM zu bekommen, sollte man den Power Plan in den Windows Energy Settings auf „High Performance“ stellen.

## SQL Server Instanz anpassen

- Sämtliche Systemdatenbanken verschieben
- XEvents, Trace Files und Error Log verschieben
- TempDB auf mehrere Dateien aufteilen
- SQL Server Memory Settings optimieren
- Index Fill Factor anpassen
- Die Parallelisierung optimieren
- „Optimize for Ad Hoc Workloads“ einschalten



### Sämtliche Systemdatenbanken verschieben

Um die Performance des SQL Servers in Azure zu steigern sollten sämtliche Systemdatenbanken (master, model, msdb, tempdb) auf virtuelle Disks ohne Caching verschoben werden. Bei der TempDB ist auch der temporäre Storage der VM eine Option, falls dieser auf SSDs beruht.

### XEvents, Trace Files und Error Log verschieben

Auch das verschieben der Logdateien (Extended Events, Traces, Errorlogs und Audits) weg von der Betriebssystemdisk auf eine anderer virtuelle Disk ohne Caching ist sinnvoll, da auch ein langsames Logging einen SQL Server ausbremsen kann.

### TempDB auf mehrere Dateien aufteilen

Wie bei jedem anderen SQL Server System auch, muss auch in der Cloud die TempDB auf mehrere Datenbankdateien aufgeteilt werden. Dies reduziert die Contention und steigert die Performance, gerade wenn die TempDB auf einer lokalen SSD oder im Premium Storage abgelegt wurde.

## **SQL Server Memory Settings optimieren**

Standardmäßig sind die Min und Max Server Memory Einstellungen des SQL Servers nicht konfiguriert. Hier sollte man jedoch unbedingt eine Einstellung, zumindest am Max Server Memory vornehmen, da der SQL Server sonst sämtlichen Arbeitsspeicher verbraucht und dem Betriebssystem so keine Luft zum Atmen lässt. Eine schlechte Performance ist so vorprogrammiert.

## **Index Fill Factor anpassen**

Nach der Installation des SQL Servers steht der Default Index Fill Factor auf 0, was einem Füllgrad von 100% entspricht. Wird somit kein Füllgrad beim erstellen eines Indexes mitgegeben, wird immer 100% verwendet. Dies kann zu einer schnelleren Fragmentierung von Indexes führen und somit mehr I/Os verursachen.

## **Die Standardvorgaben der Parallelisierung optimieren**

Der Standardwert der SQL Server Parallelisierung steht nach der Installation auf 0, also automatisch. Diesen Wert sollte man einschränken auf die Maximale Anzahl pro Cores einer CPU, oder je nach Workload auf einen anderen Standard. Ich empfehle hier mit einem Wert von 4 (bei 4 oder mehr Cores) für eine Standard Workload zu starten. Dies beschränkt zwar die maximale Parallelität einer Abfrage, wodurch diese langsamer werden kann, sorgt aber auch dafür, dass nicht eine Abfrage sämtliche Cores belastet. Zu bedenken ist, dass diese Vorgabe z.B. mit einer Option im Select Statement überschrieben werden kann.

Zusätzlich zum Parameter „Max Degree of Parallelism“ gibt es noch einen weiteren Parameter der die Parallelität steuert, der „Cost Threshold for Parallelism“. Hiermit wird definiert wie „teuer“ ein Query sein muss, bevor der Optimizer über parallelisierung nachdenkt. Dieser Wert steht auf 5, was für heutige Workloads nicht mehr adäquat ist. Ich empfehle hier mit einem Wert von 40 für OLTP und einem Wert von 25 für DWHs zu starten. Ist die Workload gemischt oder unbekannt, sollte man mit 25 starten. Bei diesen Werten nutzen kleine Queries keine Parallelisierung mehr, was sie in der Regel beschleunigt. Große Queries können jedoch immer noch mehrere Kerne nutzen.

### **„Optimize for Ad Hoc Workloads“ einschalten**

Für jede dem SQL Server gestellte Abfrage erzeugt der Optimizer einen Query Plan, der dann für eine spätere Wiederverwendung gecached wird. Dies kann bei vielen Queries die nur einmal ausgeführt werden zu einer hohen Speicherbelastung im Plan Cache führen. Mit der Aktivierung der Option „Optimize for Ad Hoc Workloads“ wird beim ersten Ausführen des Plans nicht der komplette Query Plan im Cache gespeichert, sondern nur ein Platzhalter. Sollte der Plan ein weiteres mal benötigt werden, wird er dann vollständig im Plan Cache hinterlegt.

## Datenbanken optimieren

- DBs mit mehr als einer Datenbankdatei angelegen
- Datenbankwartung steigert die Performance
- Wenn möglich Database Compression verwenden
- AUTO\_SHRINK und AUTO\_CLOSE deaktivieren
- AUTO\_CREATE\_STATISTICS und AUTO\_UPDATE\_STATISTICS aktivieren
- PAGE\_VERIFY mittels CHECKSUM



### **DBs mit mehr als einer Datenbankdatei angelegen**

Eine Datenbank sollte immer mit mehr als einer Datendatei angelegt werden um die Zugriffe auf mehrere Threads zu verteilen und zu parallelisieren. Gerade bei Premium Storage Accounts bringt das Performancevorteile, da Premium Storage für mehrere parallele Datenstreams optimiert wurde. Obendrein wird durch das Aufteilen der Datenbank auf mehrere Dateien die Contention verringert.

### **Datenbankwartung steigert die Performance**

Im Laufe der Zeit, während der Nutzung einer Datenbank, fragmentieren die Indizes der Tabellen. Page Splits und Forwarding Records sorgen für eine höhere I/O Last. Diesem Verhalten kann man mit regelmäßiger Indexwartung (Reorg oder Rebuild) entgegenwirken.

### **Wenn möglich Database Compression verwenden**

SQL Server Enterprise Edition bietet die Möglichkeit Daten mittels Row oder

Page Compression zu komprimieren. Gerade in einer Azure VM bringt dies riesige Vorteile. Nicht nur das die Datenmenge durch die Komprimierung verkleinert wird und somit mehr Daten auf eine virtuelle Disk passen, auch die IOPS und der benötigte RAM Verbrauch werden verringert.

### **AUTO\_SHRINK und AUTO\_CLOSE deaktivieren**

Wie auch On-Premise sind die Datenbankoptionen AUTO\_SHRINK und AUTO\_CLOSE ein NoGo. Nicht nur das durch einen AUTO\_SHRINK die I/O Last unnötig in die Höhe getrieben wird, ein AUTO\_CLOSE würde die Datenbank beim Disconnect des letzten Users schließen und beim ersten Connect wieder öffnen. Das führt ebenfalls zu höheren I/Os und langen Wartezeiten beim ersten Connect zur DB, da diese erst geöffnet werden muss.

### **AUTO\_CREATE\_STATISTICS und AUTO\_UPDATE\_STATISTICS aktivieren**

Sollte die Applikation einer Datenbank nicht explizit als Anforderung haben die Datenbank Optionen AUTO\_CREATE\_STATISTICS und AUTO\_UPDATE\_STATISTICS zu deaktivieren, sollten diese immer aktiv sein. Beide Optionen sorgen dafür, dass der Optimizer des SQL Servers immer alle benötigten Statistiken zur Verfügung hat und diese auch aktuell gehalten werden.

### **PAGE\_VERIFY mittels CHECKSUM**

Mit CHECKSUM als PAGE\_VERIFY Datenbankoption erlaubt man dem SQL Server beim schreiben von Daten auf die Disk eine Checksum über eine Page zu bilden und diese im Header zu speichern. Beim Lesen der Page von Disk wird dann die Checksum wieder ausgelesen und mit der Page abgeglichen. So könnte relativ schnell, mit geringem Overhead Datenkorruption festgestellt werden. Die weiteren Optionen NONE und TORN\_PAGE\_DETECTION sind hier keine Option.

## Backups sind auch in der Cloud notwendig

- Backup direkt auf Azure Blob Storage
  - Bis zu 1TB pro Datei und 500TB pro Storage Account
- Eigene Storage Accounts für Backups
  - Georeplikation für Backup Account möglich
- Jedes Backup in eine eigene Datei schreiben
- File-Snapshot Backups ab SQL Server 2016
- Immer aktivieren
  - Backup Compression
  - Backup Checksum



### Backup direkt auf Azure Blob Storage

Der SQL Server bietet seit SQL Server Version 2012 SP1 CU2 die Möglichkeit Backups direkt in einem Azure Blob Storage Account zu speichern. Dies ist eine tolle Möglichkeit SQL Server Backups in der Cloud unabhängig von der VM selbst zu speichern, was die Datensicherheit erhöht. Auch ist man damit für die Backups unabhängig von den Einschränkungen der VMs bzgl. der Disks. Dabei kann eine einzelne Backups bis zu 12TB groß werden. Insgesamt können 500TB an Backups in einem Storage Account abgelegt werden. Dies funktioniert nicht mit Premium Storage Accounts.

### Eigene Storage Accounts für Backups

Es empfiehlt sich für Backups, egal ob direkt im Blob Storage oder direkt in einer virtuellen Disk, einen eigenen Storage Account zu verwenden. Dadurch erhöht sich die Datensicherheit der Umgebung. Sollte es Probleme mit dem Storage Account geben in dem die Datenbankdateien und Transactionlogs liegen, ist der Account mit den Backups weiterhin funktionsfähig. Auch kann dieser Storage Account zur Absicherung mit Georeplikation weiter abgesichert werden.

## Jedes Backup in eine eigene Datei schreiben

Dies ist eine Empfehlung nicht nur für die Cloud. Der SQL Server bietet zwar die Möglichkeit mehrere Backups in einer Datei zu speichern, es ist jedoch davon abzuraten diese Funktion zu nutzen. Sollte eine Datei mit mehreren Backups nicht mehr lesbar sein, verliert man gleich mehrere, wenn nicht sogar alle Backups. Besser ist es pro Backup eine eigene Datei zu verwenden.

## Immer aktivieren

Beide Optionen COMPRESSION und CHECKSUM sollten bei einem SQL Server Backup aktiviert sein. Durch das Aktivieren der Optionen wird zwar ein wenig mehr CPU Last erzeugt, jedoch wird während des Backups die Konsistenz der gesicherten Daten überprüft und für das gesamte Backup eine Checksum erzeugt. Obendrein sorgt die Option COMPRESSION dafür, dass die Größe des Backups verringert wird, was natürlich auch die IOPS reduziert.

Beide Optionen können in den SQL Server Instanz Settings als Standard Konfiguriert werden. Backup Checksum jedoch erst ab SQL Server 2014.

## Demo

- Backup direkt auf Azure Blob Storage
- Restore von Azure Blob Storage



# Sicherheit

## Sicherheit der Infrastruktur

- Container Security in Storage Account auf Private stellen
- Zugriff auf VMs einschränken
  - Zugriff über Site2Site, Remote Access oder ExpressRoute
  - Keine Endpoints, keinen direkten Zugriff aus dem Internet
  - Zugriff mittels Network Security Groups einschränken
  - Eventuell Force Tunneling verwenden
- Starke Passwörter für Azure Accounts und VMs
- Sammlung von anonymen Azure Telemetrie Daten
  - HKLM\SOFTWARE\Microsoft\Windows Azure\AzPerfMonitor



### **Container Security in Storage Account auf Private stellen**

Alle Dateien die in einem Storage Account gespeichert werden müssen in Container geordnet werden. Container sind vergleichbar mit Verzeichnissen, wobei eine Verschachtelung nicht möglich ist. Für jeden Container kann der Zugriff gesondert eingeschränkt werden. Container, Blob und Private sind Optionen die zur Verfügung stehen. Gewählt werden sollte jedoch nur Private, da diese Option nur autorisierte Zugriffe zulässt.

### **Zugriff auf VMs einschränken**

Ein wichtiger Sicherheitsaspekt ist der Zugriff auf die VMs in Azure. Ein Zugriff auf die SQL Server VMs sollte nur über eine Site2Site VPN, ein Remote Access VPN oder über eine ExpressRoute möglich sein. Ein direkter Zugriff aus dem Internet sollte in keinem Fall eingerichtet werden.

Auch sollte man überlegen Forced Tunneling in seinem virtuellen Netzwerk einzusetzen, wodurch sämtlicher Traffic von Azure VMs durch den VPN-Tunnel oder die Express-Route ins eigene Rechenzentrum muss.

Zusätzlich bieten Network Security Groups die Möglichkeit einer ACL / Firewall auf Netzwerk Interface oder virtueller Netzwerkebene.

Zu guter Letzt sollte man den Zugriff auf die VMs noch mittels Windows-Firewall einschränken.

### **Starke Passwörter für Azure Accounts und VMs**

Dies ist selbstklärend... Die Passwörter für den Zugriff auf das Azure Portal und für den Zugriff auf die VMs sollten einer starken Richtlinie entsprechen. Optimal ist es die Anmeldung am Azure Portal und an den VMs über das eigene Active Directory zu steuern. Auch die Nutzernamen sollten nicht einfach zu erraten sein. Admin, Administrator, Superuser, Root sind hier keine gute Wahl.

### **Sammlung von anonymen Azure Telemetrie Daten**

Microsoft sammelt zur Verbesserung der Azure Services anonyme Leistungstelemetriedaten von VMs. Über diese Daten sind keine Rückschlüsse auf die eigene Person oder das Unternehmen möglich. Die Sammlung ist standardmäßig aktiv und kann über den Registrieschlüssel HKLM\SOFTWARE\Microsoft\Windows Azure\AzPerfMonitor deaktiviert werden.

## Sicherheit des SQL Servers

- Aktives Release Management
- Nur Windows Authentifizierung verwenden
- SA – Account deaktivieren und umbenennen
- Transportwegverschlüsselung nutzen
- Transparent Data Encryption einsetzen
- Monitoring auf ungewöhnliche Workload
- Always Encrypted



### Aktives Release Management

Um die Angriffsmöglichkeiten auf eine SQL Server Instanz so gering wie möglich zu halten, sollte ein Release Management Prozess implementiert werden. Dies bedeutet aktive Überwachung auf neue Service Packs, Cumulative Updates und Hotfixes, sowie deren rasche Implementierung falls notwendig.

### Nur Windows Authentifizierung verwenden

Der SQL Server bietet 2 Authentifizierungsmechanismen an. Windows und SQL Server Authentication. Die Methode SQL Server Authentication sollte nur eingesetzt und aktiviert werden, wenn sie für den Betrieb einer Applikation zwingend erforderlich ist. SQL Server Authentication speichert Benutzernamen und Passwörter in der MASTER-Datenbank, was nur ein Grund ist und diese Authentifizierungsmethode angreifbar macht.

### SA – Account deaktivieren und umbenennen

Jede SQL Server Instanz besitzt einen Account mit dem Namen SA welcher

die Rechte eines SYSADMINS besitzt. Um die Angriffsfläche der Instanz weiter zu verkleinern sollte man diesen Account umbenennen und deaktivieren. Da dieser Benutzer allgemein bekannt ist, bietet er einen optimalen Ansatzpunkt für Angriffe.

### **Transportwegverschlüsselung des SQL Servers nutzen**

Der SQL Server unterstützt das verschlüsseln der Daten die über das Netzwerk verschickt werden. Standardmäßig werden nur Passwörter verschlüsselt versandt, durch die Aktivierung dieser Funktion werden jedoch alle Netzwerkpakete mittels Secure Socket Layer (SSL) verschlüsselt. Dadurch wird das mitschneiden von Paketen verhindert.

### **Transparent Data Encryption einsetzen**

Auch ist es möglich im SQL Server (Enterprise Edition) die Dateien einer Datenbank zu verschlüsseln. Die Funktion Transparent Data Encryption (TDE) ist völlig transparent für die Applikation und kann somit ohne Probleme verwendet werden. Durch TDE werden die Daten vor Diebstahl, zum Beispiel durch kopieren der virtuellen Disks oder der Datendateien durch Dritte geschützt. Um sich auch vor dem Diebstahl einer kompletten VM zu schützen sollte man jedoch die Keys, die zur Verschlüsselung verwendet werden, in ein Extensible Key Management Module auslagern. Das kann eine Lösung On-Premise sein, die durch einen Site2Site VPN-Tunnel oder eine Express Route angesprochen wird, oder ein Azure Key Vault.

<http://blogs.technet.com/b/kv/archive/2015/01/12/using-the-key-vault-for-sql-server-encryption.aspx>

### **Monitoring auf ungewöhnliche Workload**

Ob die SQL Server Instanz On-Premise läuft oder in der Cloud, selbstverständlich muss die Instanz überwacht werden. Dabei geht es nicht nur darum Probleme frühzeitig erkennen zu können, auch die Sicherheit kann man damit massiv erhöhen. Wird zum Beispiel ein über den Maßen hoher Netzwerktraffic oder mehr Transaktionen als normal aufgezeichnet kann dies auf einen Angriff hindeuten.

