Inside Azure Storage Demystified

Vinod Kumar M (Technical Architect – MTC)

@vinodk_SQL

http://blogs.ExtremeExperts.com

Session Objectives And Takeaways

Session Objectives

- ✓ Enjoy good geeky fun
- ✓ Empower you to articulate the dependence of Virtual Machines on Azure Storage
- ✓ Enlighten you on the possibilities of Azure File Service
- ✓ Enable you to configure storage for peak VM performance

Session Takeaways

- ➢ Storage fuels Azure
- ➢ Azure File Service is a great option
- ➢ You can improve performance

Microsoft Azure Storage

- Cloud Storage - Anywhere and anytime access
    - Blobs, Disks, Tables, Queues and Files
- Highly Durable, Available and Massively Scalable
    - Easily build "Internet scale" applications
    - More than 25 trillion stored objects
    - 2.5+ Million requests/sec on average
- Pay for what you use
- Exposed via easy and open REST APIs, Client Libraries and Tools

Abstractions – Blobs and Disks

- Blobs – Massively scalable object store in the cloud
  - Simple REST interface (Put, Get, Delete)
  - Data sharing – share documents, pictures, video, music, etc.
  - Big Data – store raw data/logs and compute/map reduce over data
  - Backups – data and device backups

- Disks – Network mounted durable disks for VMs in Azure
  - Move on-premises applications to cloud
  - Mounted disks are VHDs stored in Azure Blobs

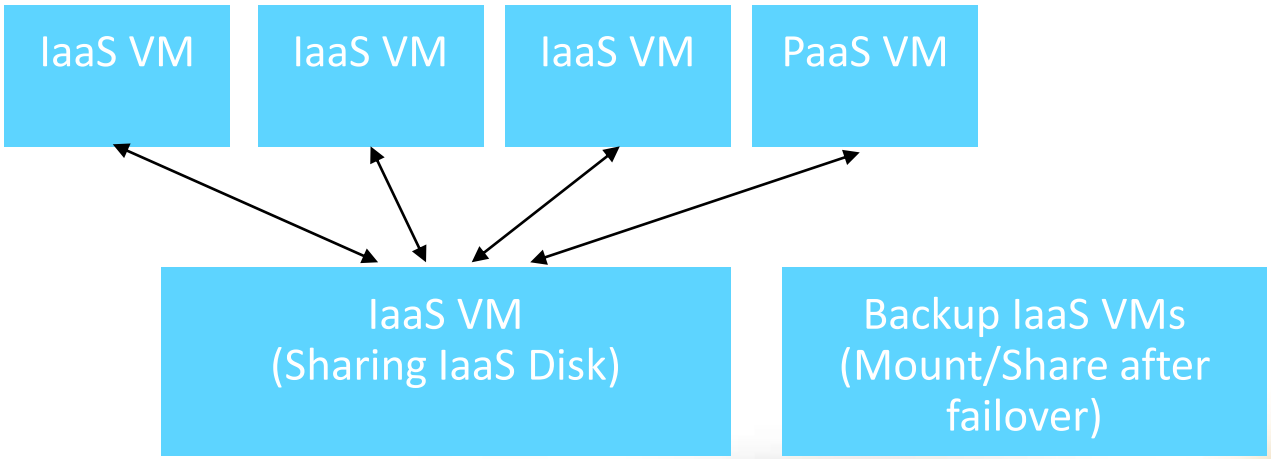
Abstractions – Tables and Queues

- Tables – Massively scalable NoSQL cloud store
  - Key/Attribute(s) store at scale
  - Auto load balances partitions to meet traffic needs
  - Store user, device or any type of metadata for your service
  - OData protocol (AtomPub or JSON)

- Queues – Reliable messaging system
  - Reliable, low latency, high throughput messaging system
  - Decouple components/roles
    - Web role to worker role communication
    - Allows roles to scale independently
  - Implement scheduling of asynchronous tasks
  - Building process/work flows

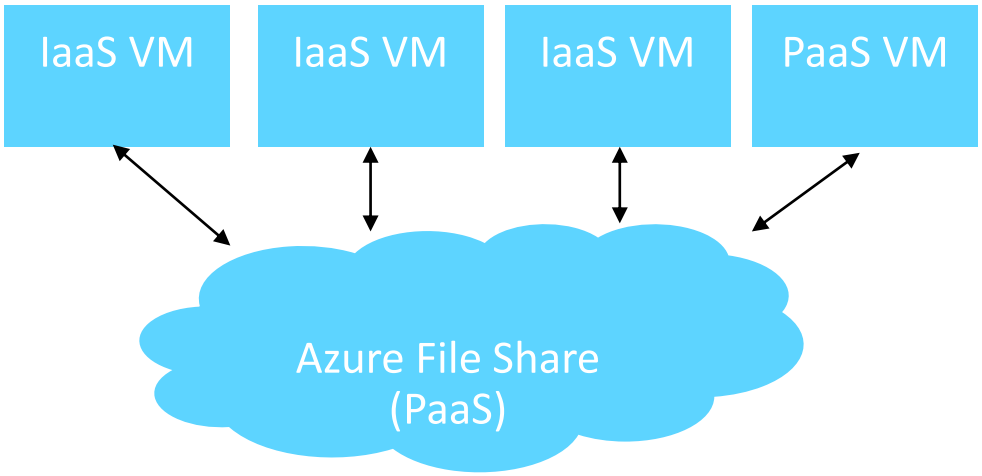
Sharing Files – The old way

- Setup an IaaS VM to host a File Share backed by an IaaS Disk
- Write code to find the IaaS File Share from the rest of the VMs in your service.
- Write some code to provide high availability
  - Handle host upgrades, node failures
- You can only access the File Share from other VMs



Azure Files

- Shared Network File Storage for Azure
- Availability, durability, scalability are managed automatically
- Supports two interfaces: SMB and REST



Azure Files vs Blobs

Description	Azure Blobs	Azure Files
Durability Options	LRS, ZRS, GRS (and RA-GRS for higher availability)	LRS, GRS
Accessibility	REST APIs	SMB 2.1 (standard file system APIs) REST APIs
Connectivity	REST – Worldwide	SMB 2.1 - Within region REST – Worldwide
Endpoints	http://myaccount.blob.core.windows.net/mycontainer/myblob	\\myaccount.file.core.windows.net\myshare\myfile.txt http://myaccount.file.core.windows.net/myshare/myfile.txt
Directories	Flat namespace however prefix listing can simulate virtual directories	True directory objects
Case Sensitivity of Names	Case sensitive	Case insensitive, but case preserving
Capacity	Up to 500TB containers	5TB file shares
Throughput	Up to 60 MB/s per blob	Up to 60 MB/s per share
Object size	Up to 1 TB/blob	Up to 1 TB/file
Billed capacity	Based on bytes written	Based on file size

Azure Files vs Disks

Description	Disk	Azure Files
Relationship with Azure VMs	Required for booting (OS Disk)	
Scope	Exclusive/Isolated to a single VM	Shared access across multiple VMs
Snapshots and Copy	Yes	No
Configuration	Configured via portal/Management APIs and available at boot time	Connect after boot (via net use on windows)
Built-in authentication	Built-in authentication	Set up authentication on net use
Cleanup	Resources can be cleaned up with VM if needed	Manually via standard file APIs or REST APIs
Access via REST	Can only access as fixed formatted VHD (single blob) via REST. Files stored in VHD cannot be accessed via REST.	Individual files stored in share are accessible via REST
Max Size	1TB Disk	5TB File Share 1TB file within share
Max 8KB IOps	500 IOps	1000 IOps
Throughput	Up to 60 MB/s per Disk	Up to 60 MB/s per File Share

Additional Services, Tools and Libraries

Azure Import/Export

- Move TBs of data into and out of Azure Blobs by shipping disks
- Submit and monitor jobs via REST and Portal
- All disks encrypted with BitLocker

Tools and Libraries

- Client libraries
  - .NET, Java, C++, Node.js
  - Windows Phone & Windows Runtime
- PowerShell commands
- CLI tools
- AzCopy – copy blobs and disks (tables later this year)
  - For backups, copying between accounts, and between on premise and cloud

Best Practices

Design Principles for Service

Use the right abstraction

- Key Lookups at scale for structured data – Tables
- Scans/retrieve large amount of raw data like for analytics/metrics etc. – Blobs
- Simple messaging between roles – Queues
- Pub/Sub, Ordered delivery etc. – Service Bus
- Highly relational data with sproc capabilities – Relational DB like SQL Server

Monitor your service

- Storage provides hourly and minute level metrics to figure anomalies
- Storage provides logs for detailed analysis and debugging
- Storage Libraries provide end to end tracing – Use client request id

Design Principles for Resiliency

Failures can occur

- Retry on Network/Connection failures
- Exponential back-off on timeout and throttling failures

Regional failures can occur

- Design for regional redundancy
- RA-GRS: Allows apps to continue with read only access to eventual consistent data

Large latencies can occur

- Latency sensitive apps should use cache. Example – XBOX/Skype etc.

Practical Patterns for Scalable & Resilient Applications

Pattern for pre-processing resources

Scenario: Encode images that are being uploaded

Add a message to the queue with blob Uri and maximum timeout to wait for blob upload

Upload the blob

Worker role processes messages

- If blob not present, wait until the conservative max timeout
- If blob exists,
  - Encode image and store encoded image
  - Delete the original image
  - Delete the message

Pattern – Scale beyond single storage account

Scenario: OneDrive needs to expand beyond a single account

Create only as many accounts as needed today

Keep a map of user name+ bucket id to a storage account name

When a bucket id fills up or account reaches limits (capacity/throughput), create a new bucket id and pick a storage account from pool for storing data

User Name	Bucket Id	Storage Account	Storage Resource	User Resource
brunopitman	1	jaidemo	jaidemo/container/brunopitman/photo1.jpg	brunopitman/1/photo1.jpg
selmaramsey	1	jaidemo1	jaidemo1/container/selmaramsey/video1.wmv	selmaramsey/1/video1.wmv
selmaramsey	1	jaidemo1	jaidemo1/container/selmaramsey/video1.wmv	selmaramsey/1/video1.wmv
brunopitman	2	jaidemo1	jaidemo1/container/brunopitman/backup.dat	brunopitman/2/backup.dat
selmaramsey	2	jaidemo22	jaidemo22/container/selmaramsey/resume.doc	selmaramsey/1/resume.doc

Summary

- Which abstraction works best for a given scenario
- Patterns that applications can use to scale
- Highly Available & Scalable Storage Service
- What differentiates us:
  - Strong consistency: Easier for applications to consume
  - GRS: Data replicated to DC 100s of miles apart
  - RA-GRS: Eventual consistent access to your data
  - Disks & Azure Files: Enables "lift & shift" of applications
  - Strong Development tools and libraries:
  - AzCopy, PowerShell, CLI, Client libraries in various languages

Thank you!