Windows Azure - Speed and Scale in the Cloud

Joe Giardino
Senior Development Lead
Windows Azure Storage

joegiard@microsoft.com
Agenda

What is Windows Azure
Abstractions
Dev Experience
Demo
Performance and Best Practice
Q & A
Windows Azure Storage

Abstractions

**Blobs** – File system in the cloud  
**Tables** – Massively scalable NoSQL storage  
**Queues** – Reliable storage and delivery of messages  
**Disks** – Durable NTFS volumes for Windows Azure VMs

Easy client access

Easy to use REST APIs and Client Libraries  
.NET, Java, Node.js and PHP, and more  

Existing NTFS APIs for Windows Azure VM Disks
Windows Azure Storage Characteristics

A “pay for what you use” cloud storage system

**Durable:** Store multiple replicas of your data

- **Local replication:**
  - Synchronous replication before returning success
- **Geo replication:**
  - Replicated to data center at least 400+ miles apart
  - Asynchronous replication after returning success to user.

**Available:** Multiple replicas are placed to provide fault tolerance

**Scalable:** Automatically partitions data across servers to meet traffic demands

**Strong consistency:** Default behavior is consistent reads once data is committed
Pricing

$0.095 to $0.037 /GB per month depending on usage and redundancy

- Locally redundant is ~27% cheaper
- Pay for what you use
- All abstractions share the same billing

$0.01 per 100k transactions

Free Account available to MSDN subscribers

90 day Trial Accounts available

Available as part of Enterprise Agreements

Develop for free with the Storage Emulator

Windows Azure Storage – How is it used?

**Xbox:** Uses Blobs, Tables & Queues for applications like Cloud Game Saves, Halo 4, Music, Kinect data collection etc.

**SkyDrive:** Uses Blobs to store pictures, documents etc.

**Bing:** Uses Blobs, Tables and Queues to implement an ingestion engine that consumes Twitter and Facebook public status feeds and provides it to Bing search

**Skype:** Uses Blobs, Tables and Queues for Skype video messaging
Abstractions
Windows Azure Blobs

http://<account>.blob.core.windows.net/<container>/<blobname>

Account
- Contoso

Container
- Images
- Video

Blob
- PIC01.JPG
- PIC02.JPG
- VID1.AVI

BLOB Storage is the simplest way to store large amounts of unstructured text or binary data such as video, audio and images with the fastest read performance.

- Highly scalable, durable, available file system.
- Blobs can be exposed publically over http.
- Can securely lock down permissions to blobs.
Blobs – File system in the cloud

Create, Delete, SetAcl Containers
Put/Get Blobs/Set Metadata
  Parallel uploads & Single range gets
Copy Blob
  Asynchronous copy – copy between accounts, web
Snapshots
  Read only version of a blob
  Promote a version as base blob

Lease on a container or blob
  15-60s lease on blob
    Useful for master election scenarios
  Infinite leases - Locks
Blob Containers

Number of Blob Containers
Can have many Blob Containers that will fit within the storage account limit

Blob Container
A container holds a set of blobs
Set access policies at the container level
  Private or Public accessible or Store Shared Access Identifiers
Associate Metadata with Container
  Metadata are <name, value> pairs
  Up to 8KB per container
List the blobs in a container
Windows Azure Queues

http://<account>.queue.core.windows.net/

Queues offer a guaranteed message delivery mechanism that can be used to distribute work across roles. Utilizing a two-phase delete process ensures that even if a role which retrieves a message goes down the message will become visible again and subsequently processed by another worker.
Windows Azure Queues - Operations

Create, Delete, SetAcl Queues
Enqueue Messages – 64KB messages
Lease mechanism for processing
  Get Message(s) – provide lease time
  Delete Message
Metadata
  Message count – Enables auto scaling
  Dequeue count – Detect poison messages
Sharing Scenarios
  Private access or Shared Access Signatures (Signed URL)
Windows Azure Tables

- http://<account>.table.core.windows.net/

Table Storage a NoSQL key-value store technology used by applications requiring storing large amounts of data storage that need additional structure at no performance cost.
Windows Azure Table – Basics

Optimistic concurrency
   Timestamp – A read only property maintained by server

Query
   Single entity lookup
   Scans: Range based query
   Performance depends on selectivity

Atomic Transactions
   Entities with same partitioning key can be part of single batch request

Exposed via RESTful APIs
   OData protocol (Now with JSON Light support!)
   Java, Node.js, PHP, .NET client libraries
Window Azure Tables - Basics

All Entities define 3 reserved properties
- PartitionKey
- RowKey
- Timestamp

Entities can contain 255 properties
Properties can be up to 64 KB each
A single entity can be up to 1 MB
A single atomic batch operation can be up to 4 MB

**Key Selection is critical**

Dev Experience
Dev Experience – Powerful Client SDKs

Basic REST libraries available for PHP, Node.js, Ruby, Python...

Full featured client SDKs - “Best practice in the box” experience

- Diagnostics / Logging
- Parallelism
- Data Integrity
- Performance
- Extensibility
- Fault Tolerance
- And more...

Full featured SDK Platform support is growing

- Java
- .Net, Windows Phone, Windows RunTime
- C++ by EOY (will work cross plat)
- Node.js
- ...And many more

All Open Source - https://github.com/WindowsAzure/
Dev Experience – How do I ...

Upload 100 GB stream
Upload 100 GB File...

...In Parallel
...Asynchronously
...with Cancellation
...with Configurable Fault Tolerance
...with Transactional Integrity Checks
...with latency guarantees
...with Logging
...with client request tracking
...with custom headers
...only if it hasn’t been modified

...In one line (ok maybe 2 😄)

```
blob.ServiceClient.ParallelOperationThreadCount = 8;

CloudBlobClient client = account.CreateCloudBlobClient();
CloudBlockBlob blob =
    client.GetRootContainerReference().GetBlockBlobReference("myBlob.txt");
	n blob.UploadFromStream(myStream);
	n blob.UploadFromFile("foo.bar", FileMode.Open);

client.ParallelOperationThreadCount = 8;

await blob.UploadFromFileAsync("foo.bar", FileMode.Open);
await blob.UploadFromFileAsync("foo.bar", FileMode.Open, token);

BlobRequestOptions options = new BlobRequestOptions();
options.RetryPolicy = new LinearRetry(TimeSpan.FromSeconds(3), 3);
options.UseTransactionalMD5 = true;
options.MaximumExecutionTime = TimeSpan.FromMinutes(60);

OperationContext ctx = new OperationContext() { LogLevel = LogLevel.Verbose;
ctx.ClientRequestId = "my request id";
ctx.UserHeaders.Add("myheader", "value");
```
Dev Experience – How do I …

Download 200 GB blob

...Asynchronously
...with Cancellation
...with Configurable Fault Tolerance
...with Transactional Integrity Checks
...with latency guarantees
...with Logging
...with client request tracking
...with custom headers
...only if it hasn’t been modified
...with automatic Geo Retry to secondary Datacenter...

...In one (longish) line of code

```csharp
CloudBlobClient client = account.CreateCloudBlobClient();

CloudBlockBlob blob =
    client.GetRootContainerReference().GetBlockBlobReference("myBlob.txt");

blob.DownloadToFile("foo.bar", FileMode.OpenOrCreate);

await blob.DownloadToFileAsync("foo.bar", FileMode.OpenOrCreate);

await blob.DownloadToFileAsync("foo.bar", FileMode.OpenOrCreate, token);

BlobRequestOptions options = new BlobRequestOptions();

options.RetryPolicy = new LinearRetry(TimeSpan.FromSeconds(3), 3);

options.UseTransactionalMD5 = true;

options.MaximumExecutionTime = TimeSpan.FromMinutes(60);

OperationContext ctx = new OperationContext()
    { LogLevel = LogLevel.Verbose, ClientRequestId = "my request id" }

ctx.UserHeaders.Add("myheader", "value");

AccessCondition condition = new AccessCondition()
    { IfNotModifiedSinceTime = DateTimeOffset.UtcNow.AddDays(-1) };

options.RetryMode = RequestRetryMode.PrimaryThenSecondary;

await blob.DownloadToFileAsync("foo.bar", FileMode.OpenOrCreate,
    new AccessCondition() { IfNotModifiedSinceTime = DateTimeOffset.UtcNow.AddDays(-1) },
    new BlobRequestOptions() { MaximumExecutionTime = TimeSpan.FromMinutes(60), UseTransactionalMD5 = true, RetryPolicy = new LinearRetry(TimeSpan.FromSeconds(3), 3 ), RetryMode = RequestRetryMode.PrimaryThenSecondary },
    new OperationContext() { LogLevel = LogLevel.Verbose, ClientRequestId = "my request id", UserHeaders = customHeaders },
    token);
```
Shared Access Signatures (SAS)
SAS allows you to provide granular access to third parties without exposing your private keys
Can be tied to a Container/Table/Queue ACL policy to provide revocability
A pre-authenticated token for a given resource expressed in the query of a given URI
Available for Blobs, Tables, and Queues
Anatomy of a SAS Token (based on policy)

?-sv=2012-02-12
&s=\b
&si=readPolicy
&sig=OoRHDwIm5aiPukNhUtWQF3tb5dQmnEef43Sq6ZPQj3o%3D
Demo – Let’s build something
Shared Access Signatures – Best Practice

Use HTTPS
   Securely use/transport SAS tokens

Use minimum permissions needed and restrict time period for access

SAS Provider Model – Have clients “renew” SAS with sufficient margin prior to expiry

Revocable SAS - Use policy to store SAS permissions, expiry etc.
   5 policies can be associated with container
   When removing and recreating policies, change policy IDs
Server side - Storage Analytics

Metrics provides usage data per abstraction
Logging provides per request logging, including client trace id’s
Logs / Metrics are stored in your storage account
   Can specify a retention policy to auto delete after some period of time
Enable via code or the Azure Portal
Provides the ability to monitor and diagnose applications
Especially useful when utilizing SAS / SAS provider model
Storage Analytics Logs

Log records for requests are stored in Windows Azure Blobs

The Log blobs are text files with one log entry per line
Each blob can contain one to many request records
A request typically appears in the log within 15 minutes after it completes execution

Configure the logging levels separately for Blob, Table and Queues
read (GET), write (PUT/POST/MERGE), delete (DELETE) requests or any combination

Best effort logging
**Storage Analytics Data Fields Logged**

The following are some of the fields logged for each record:

<table>
<thead>
<tr>
<th>Log Version</th>
<th>Operation Number</th>
<th>Request Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accessing Account</td>
<td>Request Version</td>
<td>HTTP Status Code</td>
</tr>
<tr>
<td>Owner Account</td>
<td>Operation Type</td>
<td>Client IP</td>
</tr>
<tr>
<td>Service Type</td>
<td>Start Time</td>
<td>User Agent</td>
</tr>
<tr>
<td>Request URL</td>
<td>Application End to End</td>
<td>Referrer</td>
</tr>
<tr>
<td>Object Key</td>
<td>Latency</td>
<td>Client Request ID</td>
</tr>
<tr>
<td>Request ID</td>
<td>Storage Server Latency</td>
<td>ETag</td>
</tr>
<tr>
<td>Request MD5</td>
<td>Authentication Type</td>
<td>LMT</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Request Packet Size</th>
<th>Request Header Size</th>
<th>Response Packet Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Response Header Size</td>
<td>Request MD5</td>
<td>Server MD5</td>
</tr>
<tr>
<td>Conditions Used</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Best Practice
Scalability Targets - Storage Account

Storage Account level targets
Applies to accounts created after June 7th 2012

Capacity – Up to 200 TBs

Transactions – Up to 20,000 entities/messages/blobs per second

Bandwidth for a Geo Redundant storage account
- Ingress - up to 5 Gbps
- Egress - up to 10 Gbps

Bandwidth for a Locally Redundant storage account
- Ingress - up to 10 Gbps
- Egress - up to 15 Gbps
Scalability Targets – Partition

Partition level Targets
Applies to accounts created after June 7th 2012

Single Queue – Account Name + Queue Name
Up to 2,000 messages per second

Single Table Partition – Account Name + Table Name + PartitionKey
Up to 2,000 entities per second

Single Blob – Account Name + Container Name + Blob Name
Up to 60 MB/s
Key Concept: Automatic RangePartition Load Balancing

Legend
- RangePartition
- Server Load

Load balancing is triggered based on hot Range Partitions or Partition Servers
No data is moved on disk for the reassignment
Only changing the index assignment for the Partition Servers
General Server Best Practices

Locate Storage accounts close to users
  Co-locate compute in same DC (intra-DC bandwidth is free)
  For wide spread client apps consider using more than one storage account in different physical DCs

Understand Account Scalability targets
  Account Scalability targets are different depending on Geo-Replication
  Geo Redundant: 5 Gbps ingress / 10 Gbps egress
  Locally Redundant : 10 Gbps ingress / 15 Gbps egress

Enable Logging & Metrics on each storage service
  Can be done via REST, Client API, or Portal
  Enables clients to self diagnose issues, including performance related ones
  Data can be automatically GC’d according to a user specified retention interval
General Server Best Practices (cont.)

Optimize what you send & receive
  Blobs: Range reads, Metadata, Head Requests
  Tables: Upsert, Merge, Projection, Point Queries
  Queues: Update Message, Batch size

Distribute load over many partitions
  Note: Table Entity Group Transaction (batch) only applies to entities in the same partition

Control Parallelism at the application layer
  Unbounded Parallelism can lead to slow latencies and throttling

Blend Traffic Types
  In general Blob workloads tend to be IO bound as Table workloads tend to be CPU limited. By blending traffic in roles you can make the most of your compute resources
General Server Best Practice – Access Pattern

- Graphs show a dot for every request over time (X-axis)
- Y-axis is NormalizedKey for each request, where partition key is indexed into Lexically ordered key range

Bad 😞

Good 😊
Summary
How we stack up

Nasuni recently released their “State of the Cloud Storage Industry” Report for 2013 in February

Windows Azure consistently performed better than all other CSPs in the study, which included Amazon S3, Google Cloud Storage, HP Cloud Object Storage, and Rackspace Cloud Files.

Windows Azure delivered the best Write/Read/Delete speeds across a variety of file sizes, with the fastest response times and fewest errors.

Windows Azure not only outperformed the competition significantly during the raw performance tests, but it was the only CSP to post zero errors during 100 million reads and writes.

To Read More

http://blogs.technet.com/b/stbnewsbytes/archive/2013/02/19/windows-azure-named-cloud-storage-leader.aspx
Resources

Getting Started
https://www.windowsazure.com/en-us/develop/overview/

Pricing information

Storage team blogs
http://blogs.msdn.com/b/windowsazurestorage/

Getting the Most out of Windows Azure Storage

Getting the Most out of Windows Azure Tables

Windows Azure Storage Architecture Details in SOSP Paper
One-Click Install Per Language Gives You Everything You Need to Start Developing on Azure

- **.NET Tools & SDK**
  - Visual Studio Tooling
  - .NET Client Libraries
  - Core SDK

- **Java SDK**
  - Eclipse Tooling
  - Java Client Libraries
  - Core SDK

- **PHP SDK**
  - PHP Client Libraries
  - Core SDK

- **Node.js SDK**
  - Node.js Client Libraries
  - Core SDK

Core SDK = Emulator, Packaging Tools, etc.
Questions?