Windows Azure Storage (WAS) is a cloud storage system that provides customers the ability to store seemingly limitless amounts of data for any duration of time.

- Used inside Microsoft
- In addition, there are thousands of customers outside Microsoft using WAS
- Has three forms of data
  - Blobs (user files)
  - Tables (structured storage)
  - Queues (message delivery)
- Has some key design features, include:
  - Strong consistency
  - High availability
  - Partition tolerance
  - Global and Scalable Namespace/Storage
  - Disaster Recovery
Global Partitioned Namespace

Overview

A key goal of AWS is to provide a single global namespace that allows clients to address all of their storage in the cloud and scale to arbitrary amounts of storage needed over time.

- AWS leverage DNS as part of the storage namespace and break the storage namespace into three parts:
  - account name
  - partition name
  - object name

As a result, all data is accessible via a URI of the form:

```
http(s)://AccountName.<blob/table/queue>.core.windows.net/PartitionName/ObjectName
```
Global Partitioned Namespace

Overview

**AccountName** Is used to locate the primary storage *cluster* and *data center* where the data is stored. This primary location is where all requests go to reach the data for that account. An application may use multiple AccountNames to store its data across different locations.

**PartitionName** Locates the data once a request reaches the storage cluster. The PartitionName is used to scale out access to the data across *storage nodes* based on traffic needs.

**ObjectName** Identifies individual objects within that partition. The system supports atomic transactions across objects with the same PartitionName value.
Global Partitioned Namespace

Overview

- For **Blobs**, the full blob name is the PartitionName.
- For **Tables**, each entity (row) in the table has a primary key that consists of two properties: the PartitionName and the ObjectName. This distinction allows applications using Tables to group rows into the same partition to perform atomic transactions across them.
- For **Queues**, the queue name is the PartitionName and each message has an ObjectName to uniquely identify it within the queue.
High Level Architecture

WAS Architectural Components
High Level Architecture
WAS Architectural Components

Storage Stamps  A storage stamp is a cluster of $N$ racks of storage nodes, where each rack is built out as a separate fault domain with redundant networking and power.

Location Service (LS)  Manages all the storage stamps. It is also responsible for managing the account namespace across all stamps. The LS allocates accounts to storage stamps and manages them across the storage stamps for disaster recovery and load balancing. The location service itself is distributed across two geographic locations for its own disaster recovery.
High Level Architecture

Three Layers within a Storage Stamp

Stream Layer stores the bits on disk and is in charge of distributing and replicating the data across many servers to keep data durable within a storage stamp. The stream layer can be thought of as a distributed file system layer within a stamp.

Partition Layer This layer manages which partition server is serving what PartitionName ranges for Blobs, Tables, and Queues. In addition, it provides automatic load balancing of PartitionNames across the partition servers to meet the traffic needs of the objects. The Partition Layer stores data on top of the Stream Layer.

Front-End (FE) layer Consists of a set of stateless servers that take incoming requests. The system maintains a Partition Map that keeps track of the PartitionName ranges and which partition server is serving which PartitionNames.
Stream Layer

Overview

Diagram:
1. Write to Partition Layer/Client
2. Create extent
3. Allocate extent replica set
4. Primary EN
5. Secondary EN
6. Acknowledgment
7. EN

SM (State Machine)

Paxos protocol
Stream Layer

Stream

```
Stream //foo

Pointer to Extent E1

B_{11} B_{12} \ldots B_{1x}
Extent E1 - Sealed

Pointer to Extent E2

B_{21} B_{22} \ldots B_{2y}
Extent E2 - Sealed

Pointer to Extent E3

B_{31} B_{32} \ldots B_{3z}
Extent E3 - Sealed

Pointer to Extent E4

B_{41} B_{42} B_{43}
Extent E4 - Unsealed
```
Stream Layer

Stream

A stream is the "file" of WAS's filesystem. It provides a file system like namespace and API, except that all writes are append-only. A stream is an ordered list of **extent** pointers, and an extent is a sequence of append blocks.

- **Block** is the minimum unit of data for writing and reading.
- **Extent** is the unit of replication in the stream layer, and the default replication policy is to keep three replicas within a storage stamp for an extent. Each extent is consists of a sequence of blocks. The target extent size used by WAS is 1GB and when it fills up to that size the extent is **sealed** at a block boundary. Once an extent is sealed it becomes immutable.

- **Stream** A stream has a name in the hierarchical namespace maintained at the stream layer. A stream is an ordered list of **pointers to extents**. Only the last extent in the stream can be appended to. All of the prior extents in the stream are immutable.
Stream Layer

Stream Manager (SM)

- Keeps track of the stream namespace, what extents are in each stream, and the extent allocation across the Extent Nodes (EN)
- A standard Paxos cluster
- Responsible for:
  - maintaining the stream namespace and state of all active streams and extents
  - monitoring the health of the ENs
  - creating and assigning extents to ENs
  - performing the lazy re-replication of extent replicas that are lost due to hardware failures or unavailability
  - garbage collecting extents that are no longer pointed to by any stream
Stream Layer
Stream Manager (SM)

- The SM periodically polls the state of the ENs and what extents they store. If the SM discovers that an extent is replicated on fewer than the expected number of ENs, a re-replication of the extent will lazily be created by the SM to regain the desired level of replication.
- The SM does not know anything about blocks, just streams and extents.
- The SM is off the critical path of client requests and does not track each block append, since the total number of blocks can be huge and the SM cannot scale to track those.
Stream Layer
Extent Nodes (EN)

- Each extent node maintains the storage for a set of extent replicas assigned to it by the SM
- An EN knows nothing about streams, and only deals with extents and blocks
- ENs only talk to other ENs to replicate block writes (appends) sent by a client, or to create additional copies of an existing replica when told to by the SM
Stream Layer

Intra-Stamp Replication

Provides synchronous replication and is focused on making sure all the data written into a stamp is kept durable within that stamp. It keeps enough replicas of the data across different nodes in different fault domains to keep data durable within the stamp in the face of disk, node, and rack failures.

Strong consistency is built upon the following guarantees:

- Once a record is appended and acknowledged back to the client, any later reads of that record from any replica will see the same data (the data is immutable)
- Once an extent is sealed, any reads from any sealed replica will always see the same contents of the extent
Intra-Stamp Replication

Replication Flow

- When a stream is first created, the SM assigns three replicas for the first extent (one primary and two secondary) to three extent nodes.
- Writes to an extent are always performed from the client to the primary EN, and the primary EN is in charge of coordinating the write to two secondary ENs and to acknowledge the client when the write has been completed.
- A client can read from any replica, even for unsealed extents.
  - The primary EN and the location of the replicas never change for an extent while it is being appended to (while the extent is unsealed). Therefore, no leases are used to represent the primary EN for an extent, since the primary is always fixed while an extent is unsealed.
- When the SM allocates the extent, the extent information is sent back to the client, which then knows which ENs hold the three replicas and which one is the primary.
- When the last extent in the stream that is being appended to becomes sealed, the same process repeats.
Stream Layer

On an EN failure

- If during writing, one of the replica’s ENs is not reachable or there is a disk failure for one of the replicas, a write failure is returned to the client.
- The client then contacts the SM, and the extent that was being appended to is sealed by the SM at its current state.
- The SM will then allocate a new extent with replicas on different (available) ENs, and returns the information for this new extent to the client.
  - A key point here is that the client can continue appending to a stream as soon as the new extent has been allocated, and it does not rely on a specific node to become available again.
Stream Layer
On an EN failure

- When sealing the extent, the SM will choose the smallest commit length based on the available ENs it can talk to.
- When the faulty EN becomes reachable, the SM will force it to synchronize the given extent to its sealed length. This ensures that once an extent is sealed, all its available replicas (the ones the SM can eventually reach) are bitwise identical.
Partition Layer

Overview

The partition layer stores the different types of objects and understands what a transaction means for a given object type (Blob, Table, or Queue). The partition layer provides:

- data model for the different types of objects stored
- logic and semantics to process the different types of objects
- massively scalable namespace for the objects
- load balancing to access objects across the available partition servers
- transaction ordering and strong consistency for access to objects
The partition layer provides an important internal data structure called an Object Table (OT)

- A massive table which can grow to several petabytes
- Dynamically broken up into RangePartitions and spread across Partition Servers in a stamp
- Massively scalable namespace for the objects
  - A RangePartition is a contiguous range of rows in an OT from a given low-key to a high-key
  - All RangePartitions for a given OT are non-overlapping, and every row is represented in some RangePartition
Partition Layer

Data Model

The partition layer provides an important internal data structure called an Object Table (OT)

- The partition layer has the following OTs:
  - **Account Table**: Stores metadata and configuration for each storage account assigned to the stamp
  - **Blob Table**: Stores all blob objects for all accounts in the stamp
  - **Entity Table**: Stores all entity rows for all accounts in the stamp
  - **Message Table**: Stores all messages for all accounts’ queues in the stamp
  - **Schema Table**: Keeps track of the schema for all OTs
  - **Partition Map**: Keeps track of the current RangePartitions for all Object Tables and what partition server is serving each RangePartition. This table is used by the Front-End servers to route requests to the corresponding partition servers.
Partition Layer
Architecture
Partition Layer
Partition Manager (PM)

- Responsible for keeping track of and splitting the massive Object Tables into RangePartitions and assigning each RangePartition to a Partition Server to serve access to the objects.
- Ensures that each RangePartition is assigned to exactly one active partition server at any time, and that two RangePartitions do not overlap.
- Each stamp has multiple instances of the PM running, and they all contend for a leader lock that is stored in the Lock Service.
Partition Layer
Partition Server (PS)

- Responsible for serving requests to a set of RangePartitions assigned to it by the PM to the objects
- Stores all the persistent state of the partitions into streams and maintains a memory cache of the partition state for efficiency
- No two partition servers can serve the same RangePartition at the same time. Thus, providing strong consistency and ordering of concurrent transactions to objects in a RangePartition.
Partition Layer

Lock Service

- A Paxos Lock Service [3,13] is used for leader election for the PM.
- Each PS also maintains a lease with the lock service in order to serve partitions.
Partition Layer

Data Flow

- When the PS receives a write request to the RangePartition, it appends the operation into a **commit log**
- Then it puts the newly changed row into a **memory table**
  - At this point success can be returned back to the client
- When the size of the memory table reaches its threshold or the size of the commit log stream reaches its threshold, the partition server will write the contents of the memory table into a **checkpoint** stored persistently in the row data stream
- To control the total number of checkpoints for a RangePartition, the partition server will periodically combine the checkpoints into larger checkpoints
WAS Vs BigTable

Looks familiar???