Window Azure Storage: A Highly Available Cloud Storage Service with Strong Consistency

ABSTRACT
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. WAS customers have access to their data from anywhere at any time and only pay for what they use and store. In WAS, data is stored durably using both local and geographic replication to facilitate disaster recovery. Currently, WAS storage comes in the form of Blobs (files), Tables (structured storage), and Queues (message delivery). In this paper, we describe the WAS architecture, global namespace, and data model, as well as its resource provisioning, load balancing, and replication systems.

Notes
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.

Lessons Learned
Windows Azure uses two forms of replication:

- Intra-stamp replication provides synchronous replication it ensures that all data written to a stamp is durable within that stamp. When a client writes to a primary server (Extent Node), that primary is responsible for replicating the data written to two other servers within the stamp. This replication is on the critical path of client write requests and a successful response to a write request does not return until the replication is complete.

- Inter-stamp Replication provides asynchronous replication. It focuses on replicating data across stamps. This form of replication is not in the critical path of a client’s write request and instead runs as a background process.

This two pronged approach to consistency seems very powerful. Intra-stamp replication provides protection against hardware failures and Inter-stamp replication provides protection against geo disasters (provided that data is being replicated to geographically disperse stamps).

Also, unlike GFS, Azure’s consistency model guarantees bitwise equivalence across replication which may ease the burden on client developers.
The amount of storage savings that can be realized when erasure coding is used.

**Possible Extensions**
A possible extension of this paper may be to explore the use of different erasure coding algorithms and their effects of both speed and storage requirements within Azure.

Exploration of various garbage collection algorithms may be interesting.

Azure uses range partitioning, the development and assessment of other partitioning algorithms could prove to be a valuable extension.

**Criticisms**
Azure is an append only file system. While this allows for a more simplistic design, it requires some form of garbage collection to be constantly running in order to reclaim unused space (extents). This may lead to wasted space and/or an overworked garbage collection service. (having read the entire paper now, they do recognize this).

Some forms of failures in Azure can cause duplicate records/extent to be written. Handling the occurrence of these duplicates is left to the client applications; they are not resolved by the Azure stack.

Replication to two other Extent Nodes being in the critical path of writes likely reduces performance