
OpenStack Swift as
ownCloud Primary Storage
Multi-Region Eventual Consistency Storage
David Jericho, Solutions Architect, AARNet

CloudStor+: A brief history

CloudStor+ is AARNet's project to provide ownCloud access to managed storage and external storage for researchers

Growth has outstripped scalability of file systems, storage capability is intended to exceed 3PB in 2015

Australia is big, meaning centralisation doesn't work

Data Proximity: Why things need to be close

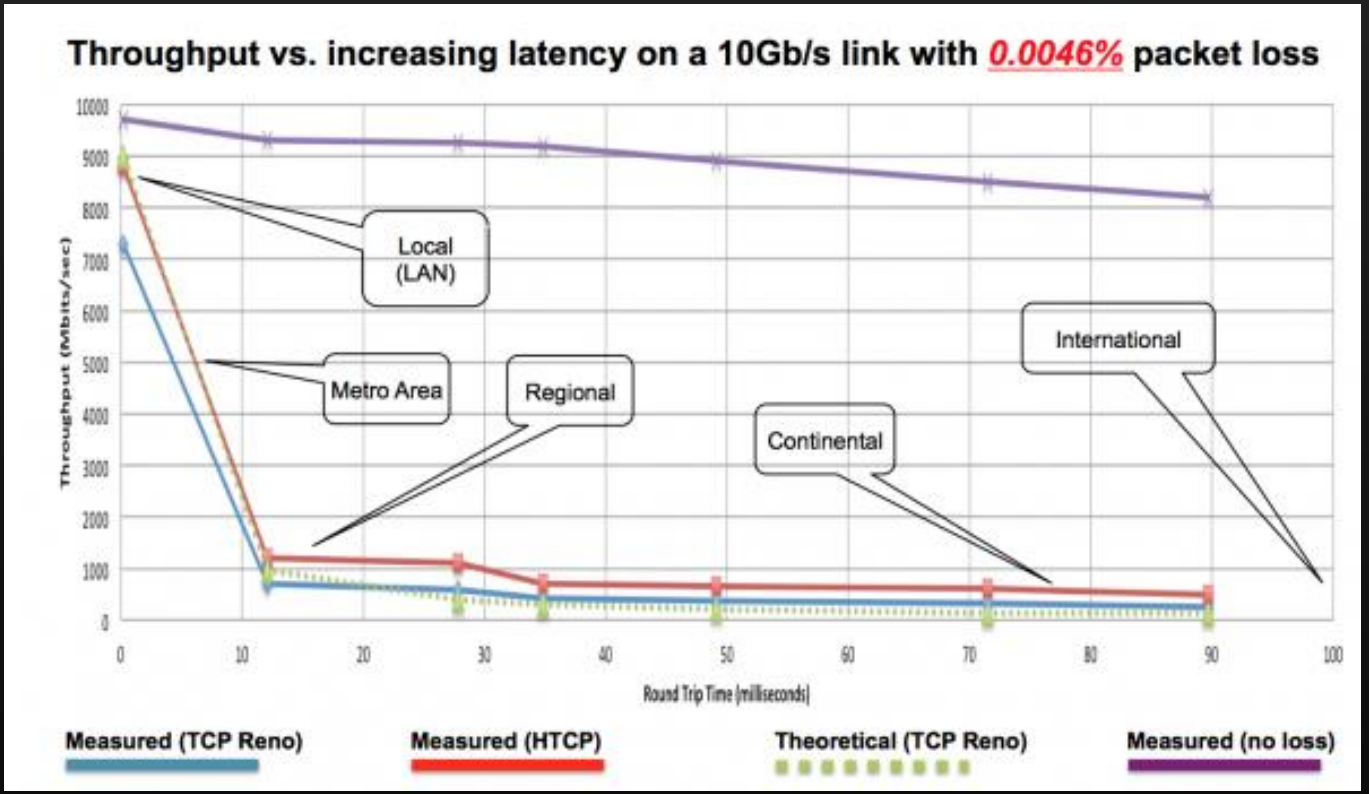


Image source: http://fasterdata.es.net/network-tuning/tcp-issues-explained/packet-loss/

Data Proximity: Australia is 2986.1 square milliseconds

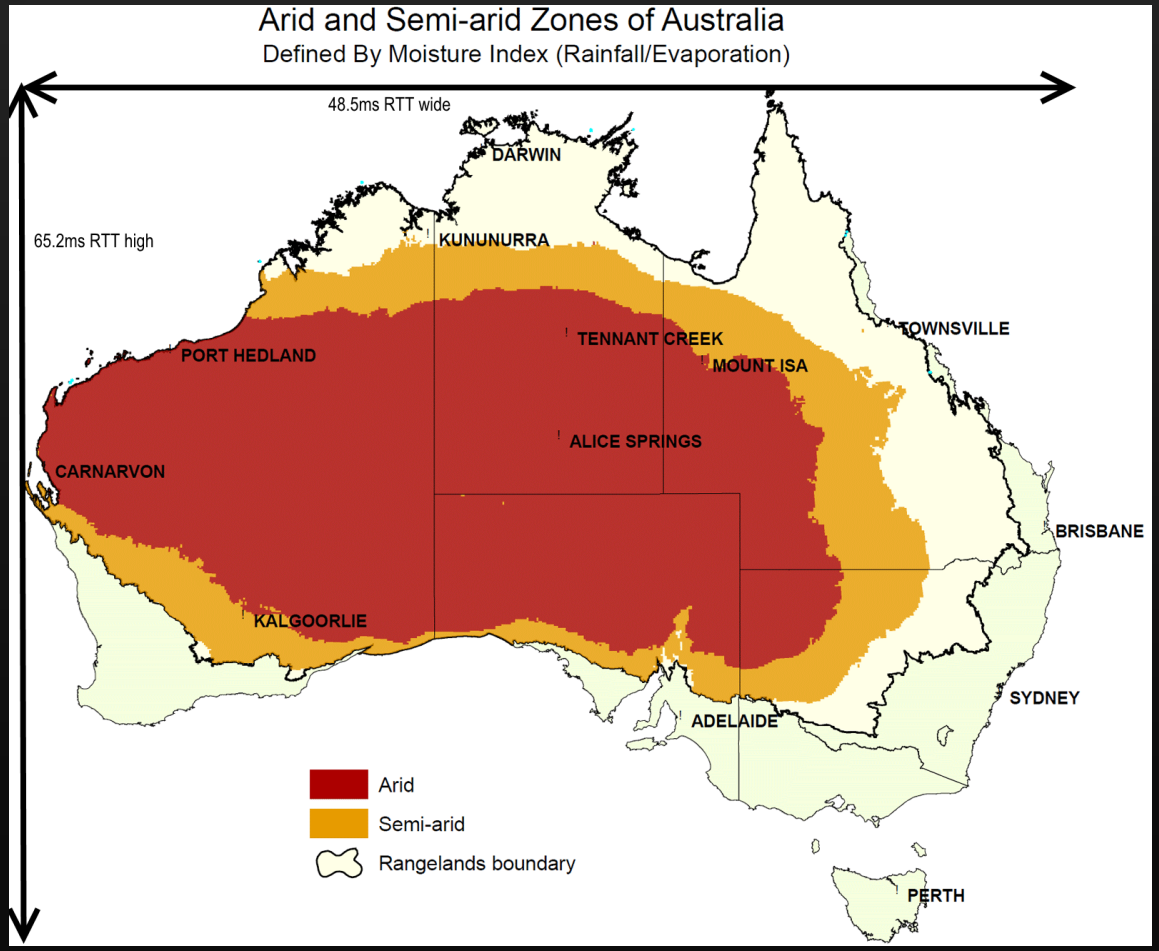
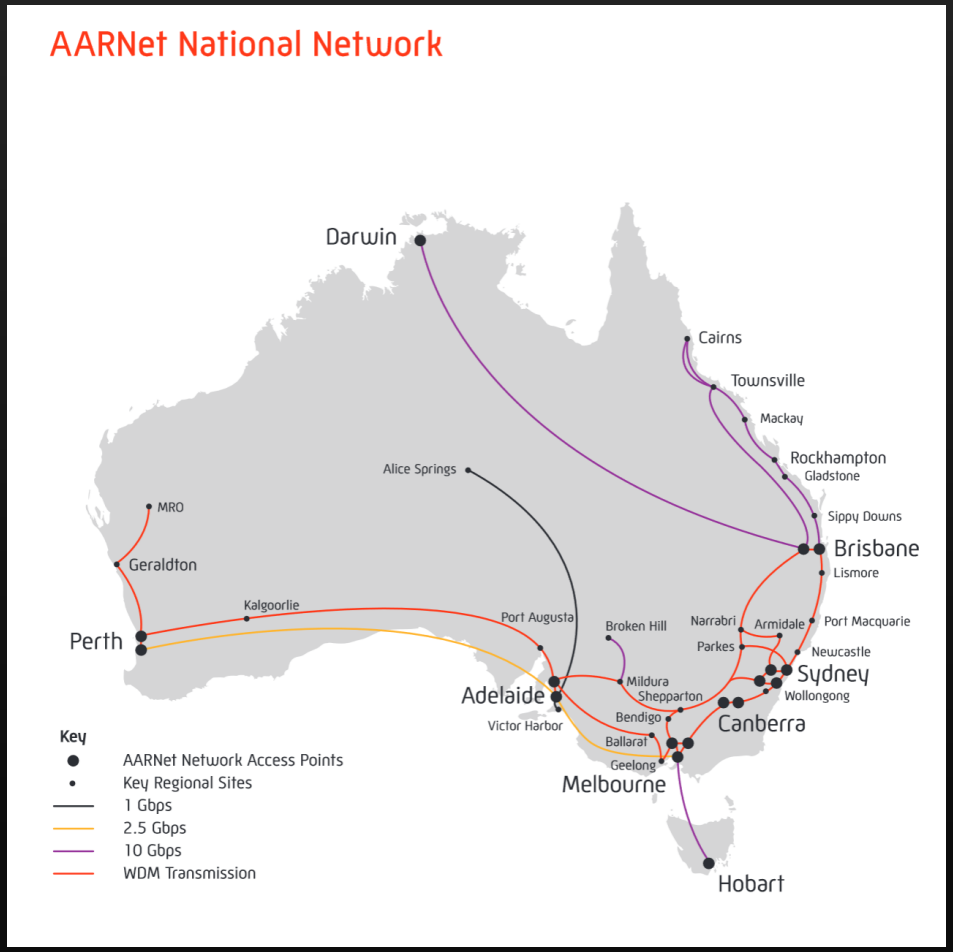


Image source: http://mapsof.net/map/arid-and-semi-map-australia

Data Proximity: Population location makes distances larger again



Source: http://www.aarnet.edu.au/network-and-services/the-network

Data Proximity: In short, geography makes everything harder

Source: Wombat Awareness Organisation

CloudStor+: Seeing as we're working to do it at 88ms…

Work to extend services to New Zealand (23.0ms from Sydney)

Why not go further?

Intend to experiment as far as we can reach with good networks

Still need to handle network splits

Swift in a Nutshell

On Swift: Strict consistency isn't fast enough at scale or distance

Hadoop, Ceph, Gluster, XtreemFS, and others try to resolve CAP, use Paxos, or similar behaviours

Incoming data exceeding the ability to replicate, results in cluster reliability failures as consistency is enforced

Cluster failure is bad, in part because recovery extends the effective outage period

On Swift: Eventual consistency rings in Swift

"One Ring to rule them all, One Ring to find them, One Ring to bring them all and in the darkness bind them" – J R R Tolkien

Really three rings, an account ring, a container ring, an object ring

Eventual consistency makes rings of rings much easier

BASE (Basically Available, Soft state, Eventual consistency) – we can tweak around this

On Swift: Swift and writing replicas

System “peels” off locally written replicas to remote regions

Step 1 very easily could be a single host with 3 PCI-e flash devices

Policies then have the data moved to cheaper bulk storage behind the scenes

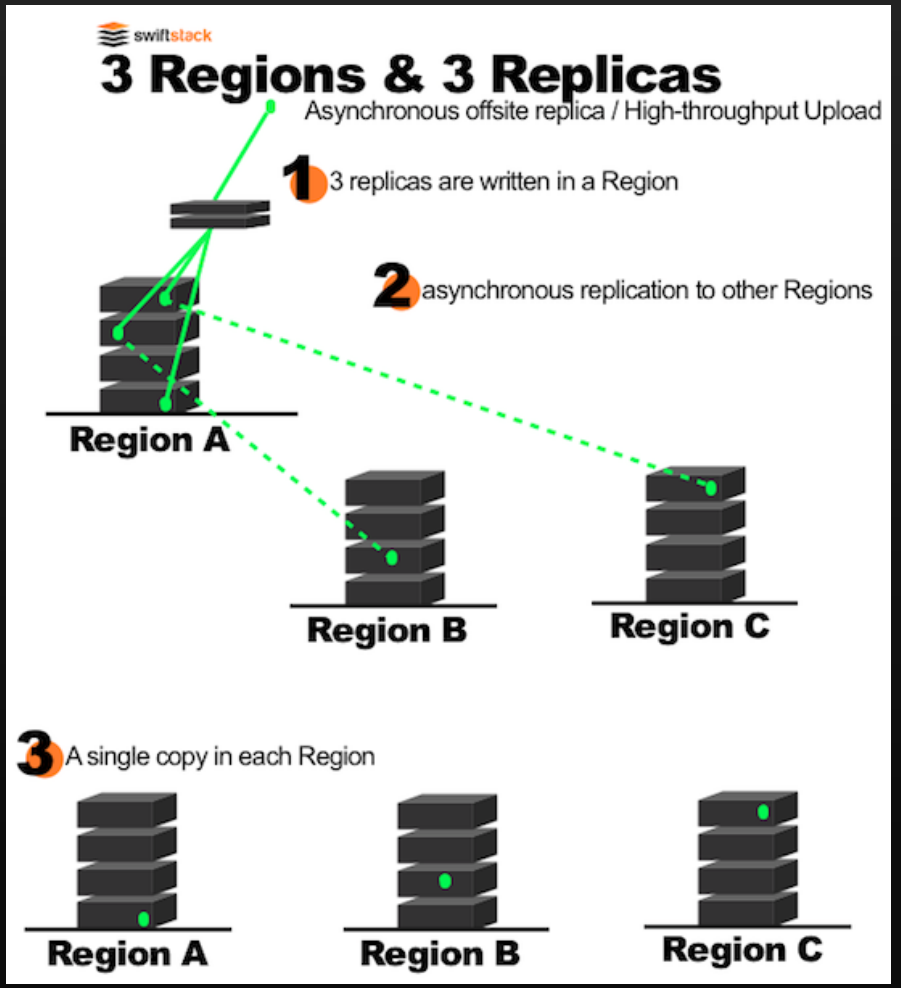


Image source: https://swiftstack.com/blog/2012/09/16/globally-distributed-openstack-swift-cluster/

On Swift: Other neat things Swift lets us do

Extremely user proximal write affinity – under the users desk

Allows location policies on data storage

Fully utilise a 10Gbps link and beyond without hurting general use

It's not actually that scary – rather safe actually

Deploy It: Sounds great, what's the problem?

Adapting software to talk Swift or S3 natively, without a local spool

Objects should be streamed, not spooled – becomes more important at larger sizes

API access to data seems to be confusing to many programmers; they fall back to direct file system access

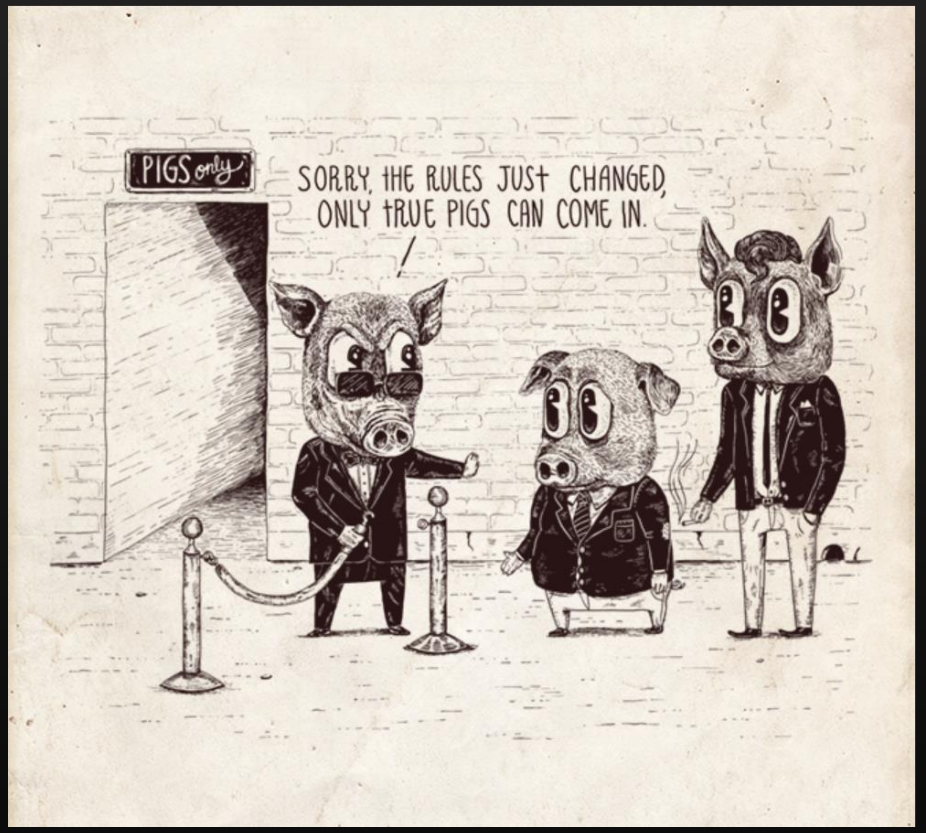


Image source: https://bookofbadarguments.com/

Deploy It: Object Behaviours

Can't append to objects in Swift – WebDAV doesn't append either

But it does mean we can run in-store versioning in the object store

Need to attach metadata, as we need to sanity check file cache against what's actually there – system scrub

Segmented objects can be handled differently and in parallel

Deploy It: Large Objects

Large objects over 5GB are handled in segments

Segments can be uploaded in parallel, fitting nicely with chunked upload

Of note! Swift in Red Hat RDO could corrupt when segments were small; out of order chunks

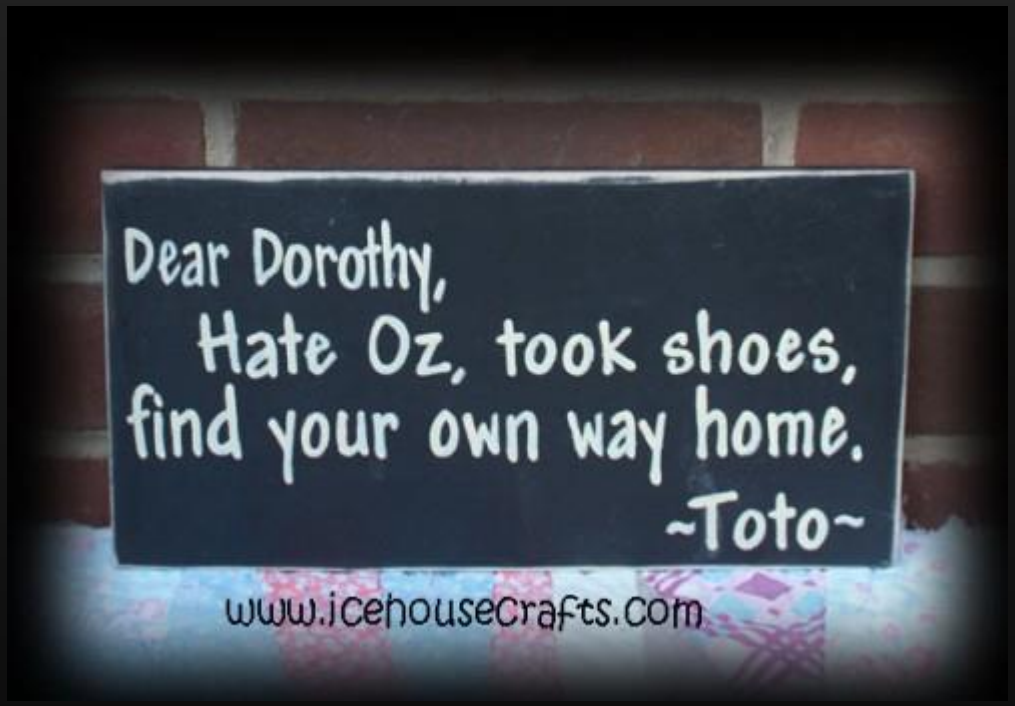
Possible to do partial versioning of segments rather than whole file – in effect, cheap de-duplication

Deploy It: Larger Objects

Users have requested single 6TB objects to date

Multiple users all doing this at once from multiple regions

Um… we're not on a LAN anymore Toto



Deploy It: Container Limits

6TB objects are 10% of the recommended container object count

Users have placed in some cases, 130,000 files in one directory

Um…

Photographer: http://commons.wikimedia.org/wiki/User:Tomomarusan

Deploy It: Fixing container limits

Containers spread across paths or other hash

Multiple containers per user

Tenant support for tiers and policies

Image source: https://maxmueller.wordpress.com/2009/12/30/evelyn-maersk/

Deploy It: Other tweaks to Swift

By default, Swift uses rsync to sync and validate replicas

Large objects, or many small objects then obviously replicate slowly

Replace rsync calls with a wrapper around UDR (rsync over UDT)

Experimenting with Facebook's mcrouter to do distributed memcache replication and hot loading

Summary: Give me my petabytes

Swift integration is working…

…with some limits, both Swift and ownCloud related

Not using all the features, but easy enough to re-architect

Doing it anyway, as there are few other ways to handle the growing scale of data at large distances and with ad-hoc usage