SHUTTERFLY INC

MSST 2016:

Shutterfly Image Archive:

Leveraging Large-Scale Disk Systems in a Web-based Photo Archive: Design, Operations, and Future Plans

Mike Kugler
Senior Manager, Storage Engineering
Shutterfly Image Archive
mkugler@shutterfly.com

Outline

- Who Am I?
- What Is the Shutterfly Image Archive?
- Basic Stats
- Disk Fleet
- Architecture
- Operations
- Interesting Events
- Disk Failures & AFRs
- Image Compression
- Future Plans

Who Am I?

- Manager, Storage Engineering, Shutterfly 2014-2016
- Storage Engineer, Shutterfly, 2010-2014
- Storage Engineer, eBay/PayPal, 2004-2009
- Various places around the Valley since before Y2K
- Very much an ‘Ops’ person at heart; I like things to run well.

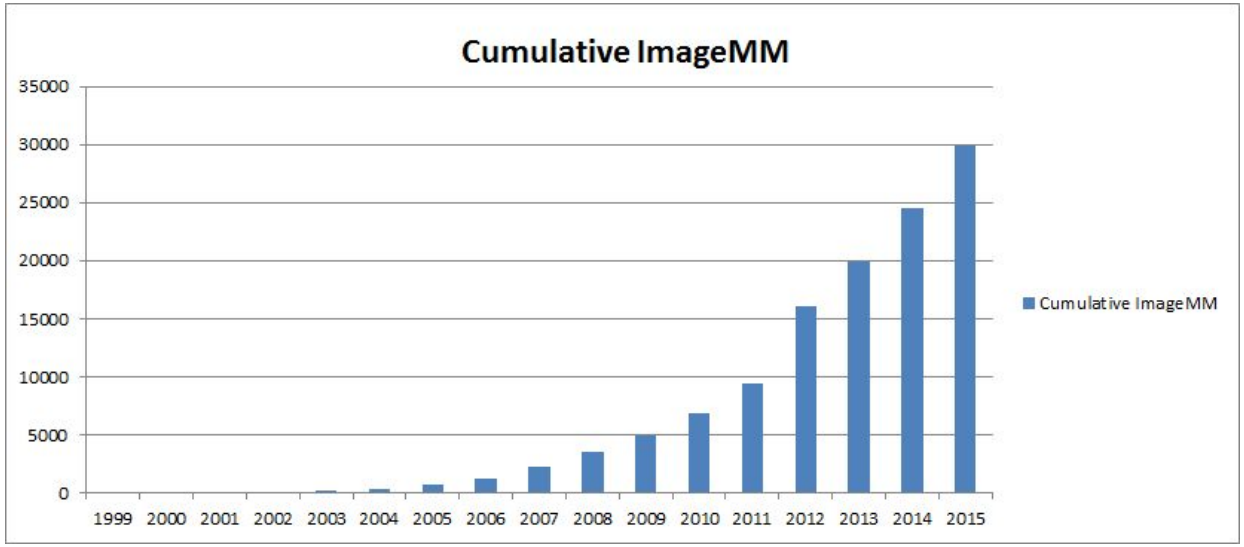
What is the Shutterfly Image Archive?

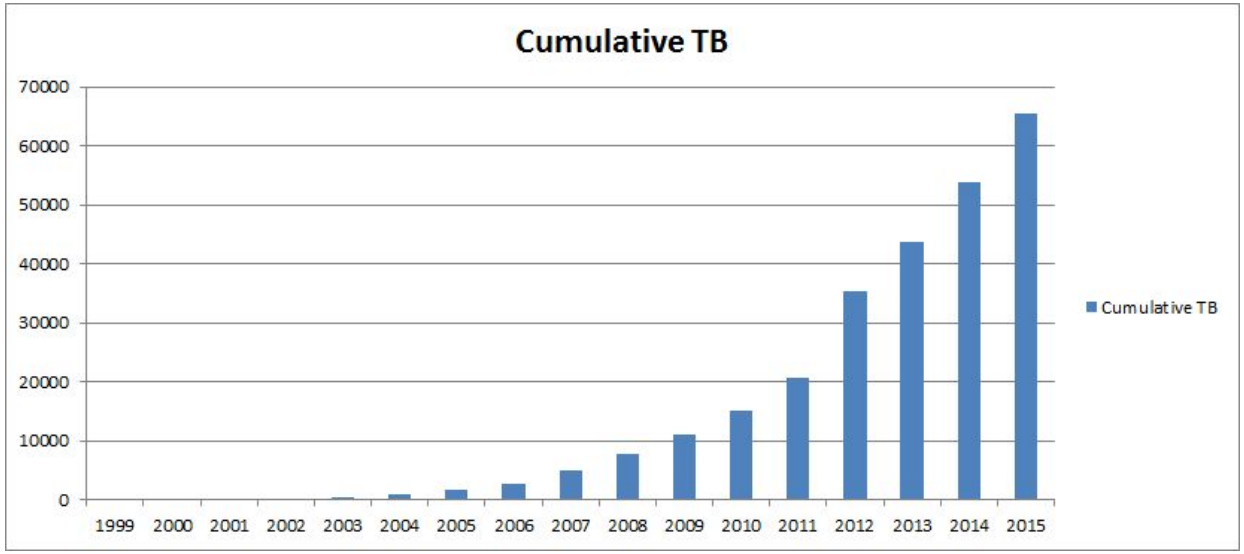
- Shutterfly's internal 'cloud' for all customer images
- Simple http endpoint for all client applications: PUT/GET
  - What happens behind the scenes is Not Their Problem
- 90% Object Store: Scalable, reliable, inexpensive storage
- 10% Legacy FC SAN: somewhat less of all of the above
- Actually 4 separate use cases:
  - Long term storage of customer pictures - 16 years and counting.
  - Short term 2nd copy of customer pictures - 90-180 days
  - Long term storage of high resolution thumbnail
  - Backups of original customer pictures
- Keeper Of The Customer Promise:
  - "Our photo storage is free and unlimited. We securely store your images at full resolution."

Basic Stats: Shutterfly Image Archive - 2012 vs. 2016

	2012	2016	% Incr			
Bn Images	16	29	81.3%			
PB User Data	35	135	285.7%	*2016 includes backup data		
PB Hardware (raw)	70	207	195.7%			
PB SAN	27	16	-40.7%			
PB Object	43	191	344.2%			
Disks	39,630	50,360	27.1%			
SAN	22,800	4,200	-81.6%			
Object	16,830	46,160	174.3%			
Devices	700	1,100	57.1%			
TB Smallest Silo/Pool	480	1,140	137.5%			
TB Largest Silo/Pool	3,510	21,504	512.6%			
TB Smallest Node	90	90	n/c			
TB Largest Node	135	672	397.8%			
Admins	4	5	25.0%			
PB/Admin	17.5	41.4	136.6%			

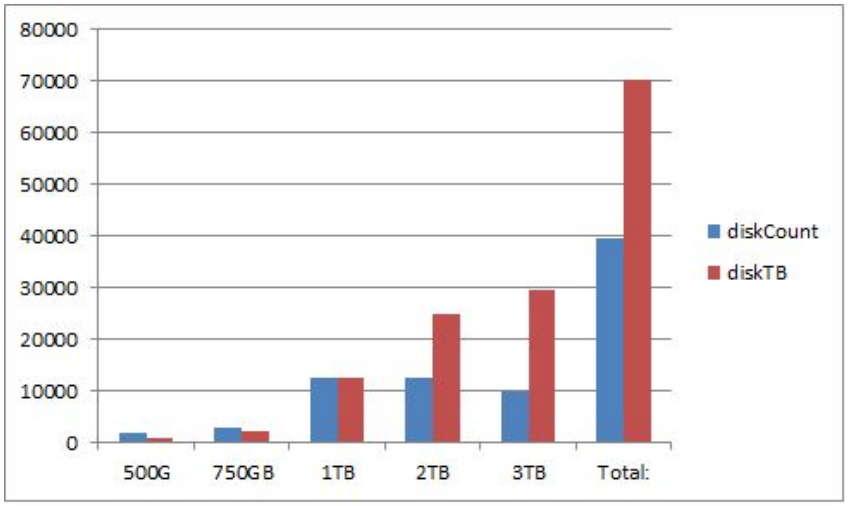
15 year Image Count & Data Usage



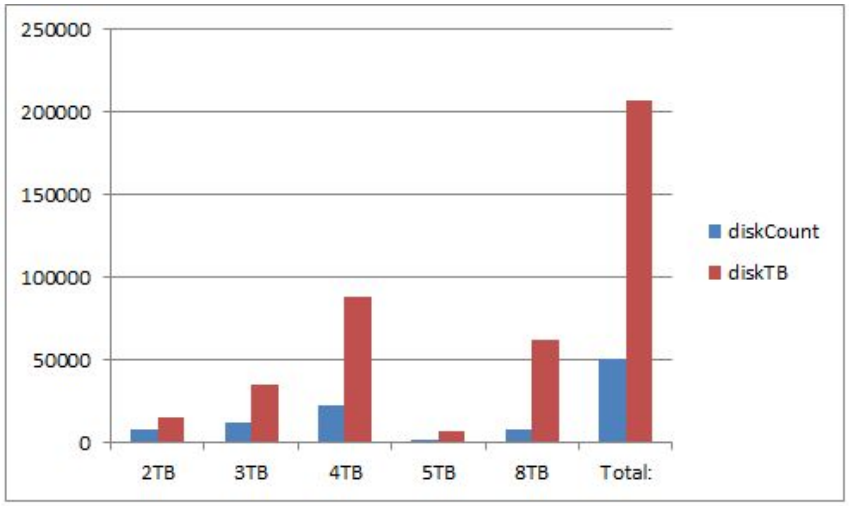


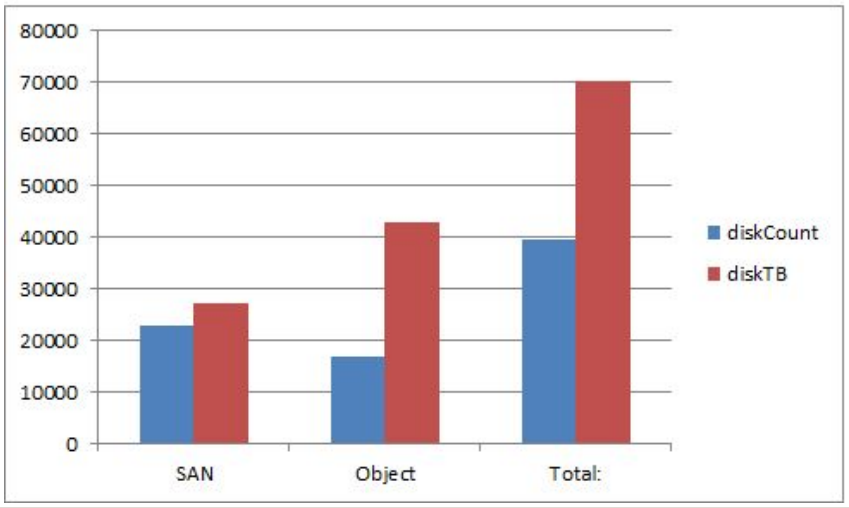
Disk Fleet - 2012 vs 2016

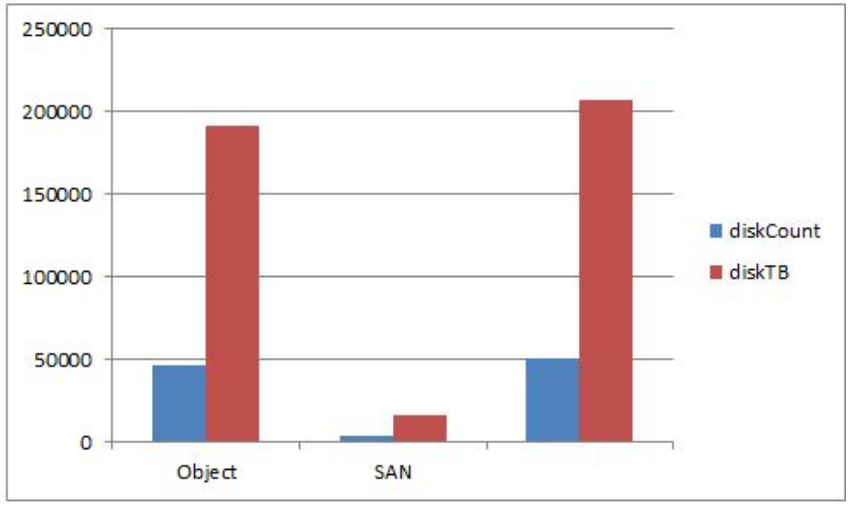
2012 Disk Counts & Sizes



2016 Disk Counts & Sizes





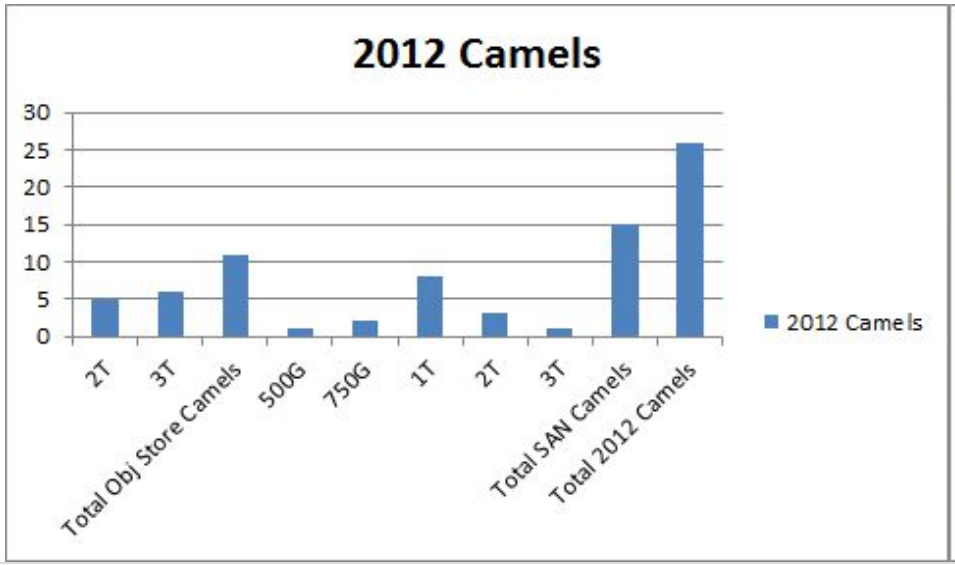


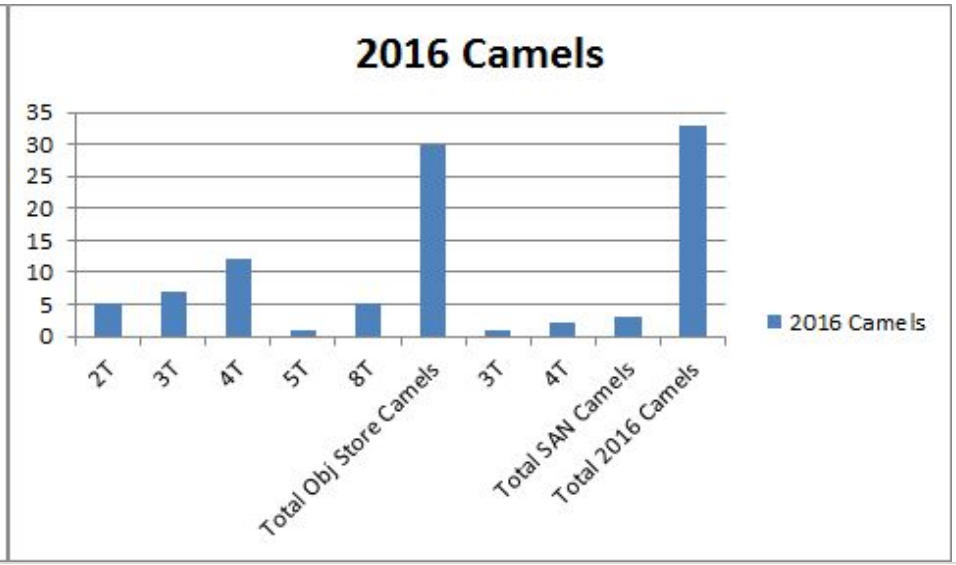
Disks Sizes Purchased by Year

Disk Fleet - Camelus Bactrianus - 2012 vs 2016

Disk quantities and TB are not the only way to measure deployed capacity though. If we consider that the average disk drive weighs 1.4lbs, we can characterize our fleet by weight. The semi-official mascot of the Image Archive is the double-hump camel, Camelus Bactrianus. The adult male can reach 2,200 lbs.

Our fleet of Object Store Camels has grown by nearly 3x since 2012. SAN Camels have declined by 5x, and the data density distribution of our camels has shifted, as would be expected.





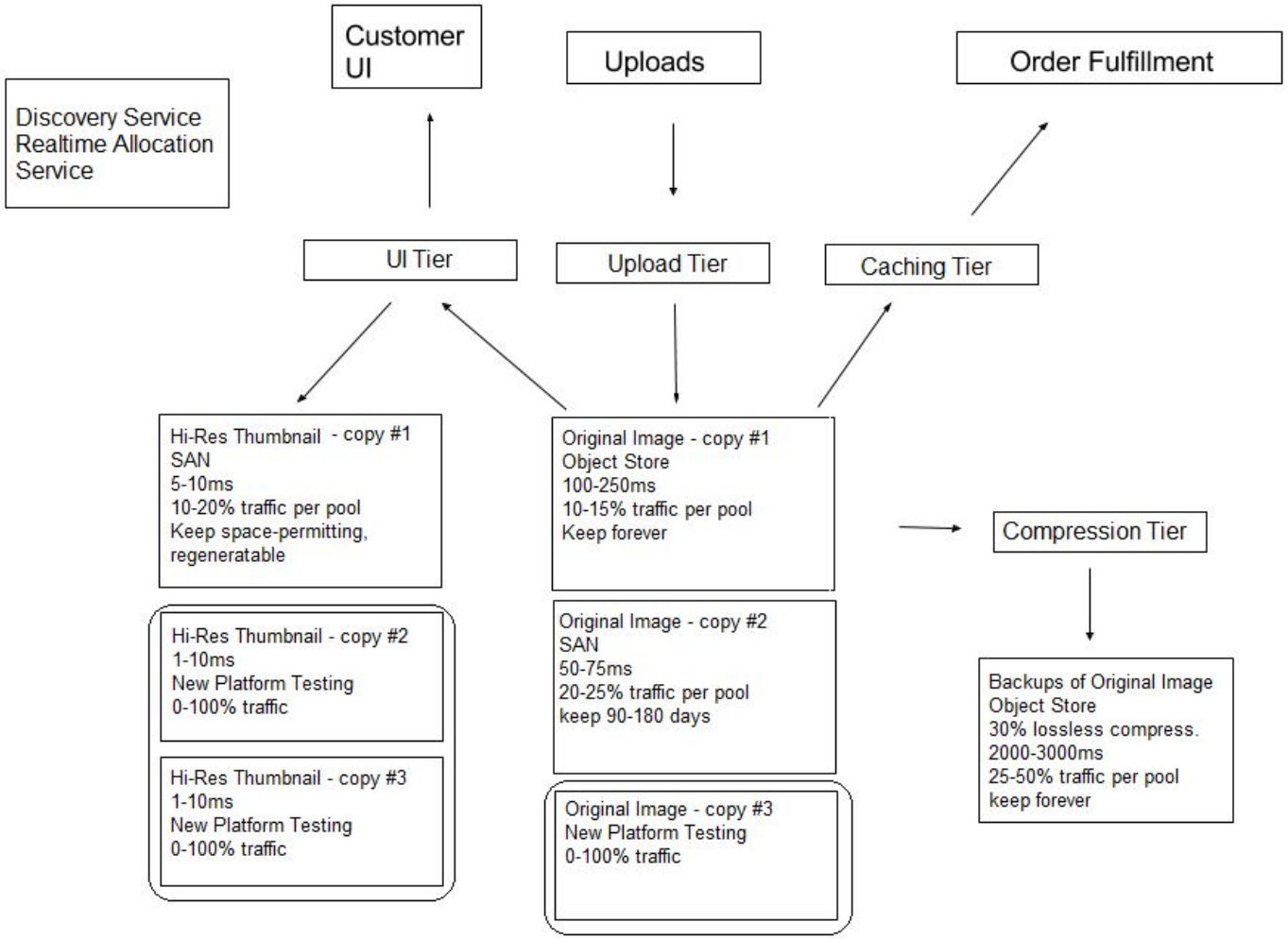
Architecture: Design Considerations

My predecessor and former manager at Shutterfly, Justin Stottlemyer, presented at MSST 2013 when we were about 2.5 years into the transition to object storage. His presentation, The 1000X Rule, is well worth the read: http://storageconference.us/2013/Presentations/Stottlemyer.pdf . The excellent scalability, resilience, and stability of the Image Archive are the results of his architectural decisions.

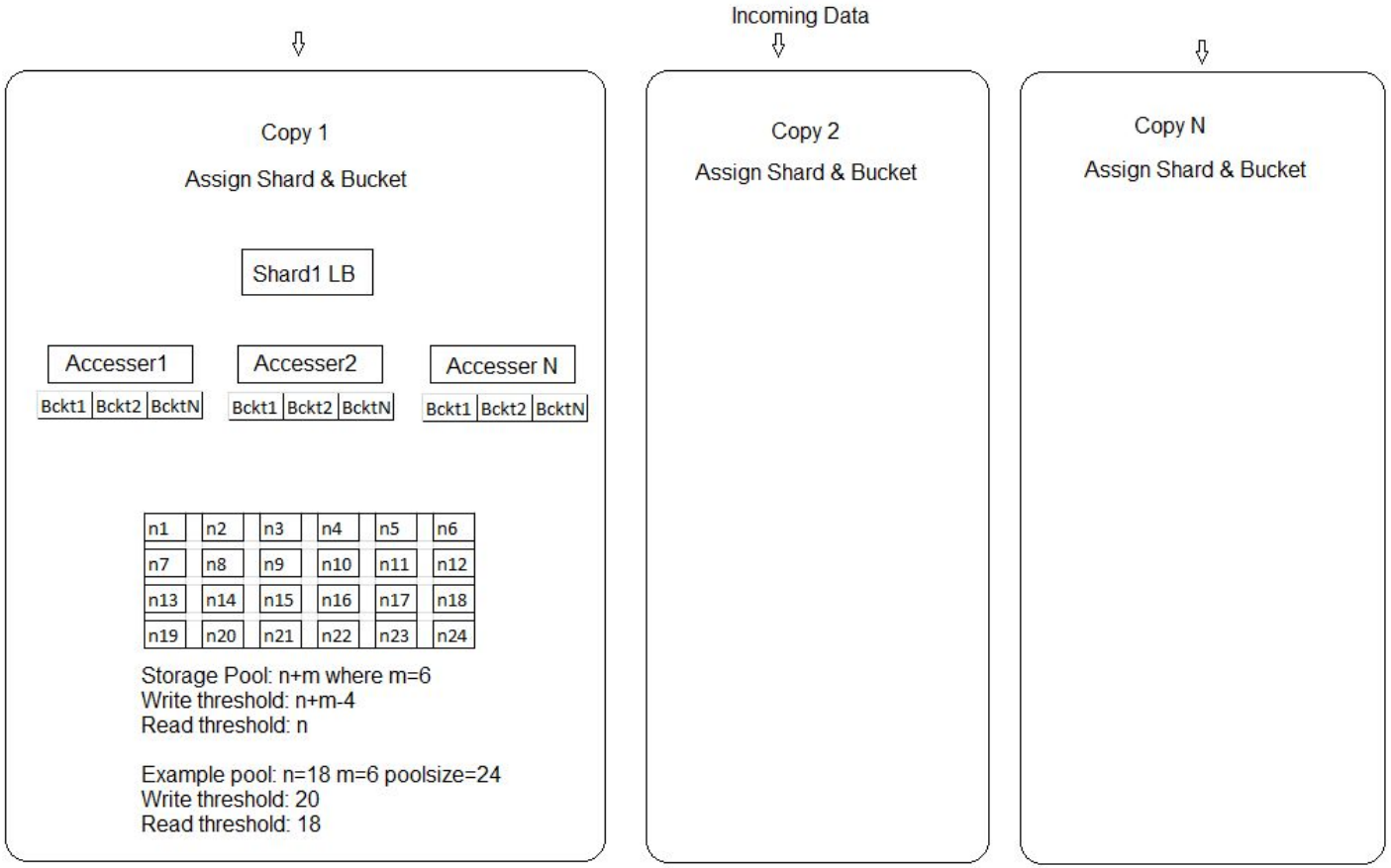
Important Design Features:

- Standard, repeatable building blocks of storage
- Simplified deployment and maintenance
- Fault tolerance orders of magnitude above standard RAID6
- Scalable access layers
- Multiple layers of redundancy
- Limited failure domains
- Upgrade in Place
- Fail in Place
- Shared-nothing architecture, no SPOFs
- Easy to introduce new technologies
- “Any m of n+m nodes can be down at any time”

Architecture: High level



Architecture: Data Ingest



Site applications only need to write to/read from 1 of N copies to function.
New technologies can be safely added as additional copies for testing.

Cleversafe specific: Any accesser can write to any bucket across multiple storpools.
Objects written to a bucket are striped across one disk in each node in the storpool.
Any m nodes in a storpool can be down and data can still be read.

Architecture: Erasure Coding

Cleversafe is a commercial implementation of Reed-Solomon erasure coding. We were an early adopter in 2010. The market has gotten considerably more crowded since then.

Very, very short version: given some data, divide it into n pieces, then calculate and add m additional pieces such that reconstruction of the original data requires only any n of the n+m pieces.

Cleversafe Entities:

- Storage Nodes: Linux hosts containing disk drives
- Storage Pools: grouping of a set of Storage Nodes
- Vaults (buckets): a logical data pool defined within Storage Pools with a specific EC scheme:
  - width (n+m): number of nodes within the storage pool that a given object will be striped across
  - read threshold: minimum number of chunks of an object needed to read and reassemble it
  - write threshold: minimum number of chunks of an object needed to be written to consider it persisted and acked back to the client.
- Accesser: http server
  - Writes: splits the object into n chunks, adds m parity chunks according to the destination vault definition; writes one chunk to one disk on each Storage Node
  - Reads: requests the object chunks from n+m Storage nodes; re-assembles object from first n chunks received; returns object to caller

Architecture: Data Availability

It is possible to come up with some crazy-large MTTDL and Data Availability numbers.

But no matter how many trillions of years you think you have, the fact remains that a given object is actually only striped across 20-30 disks, one per machine. Machines go down for various reasons, and disk rebuilds are not instantaneous, especially the larger they get, so while the math is great, sometimes the real world has other plans.

Though these are not data loss events, we've had a couple of interesting data availability events...

Case #1: Due to various resource constraints, the weekly disk replacements were deferred at one point for a couple of weeks. When the data center techs finally got sufficient time to catch up, they tried to do them all at once, and the storage admins didn't really think that through. In the course of replacing All The Disks, because they had no visibility into the logical structure of the storage pools they were working on, they took down too many nodes at once in several pools, temporarily breaking threshold for the subset of objects for whom a given node was holding that nth chunk.

Case #2: During the datacenter move, unbeknownst to the storage group, servers were placed in different physical cages depending on whether their names were odd or even numbered; this had not been the case in the original datacenter. The storage nodes communicate over IP, so it generally doesn't matter where on the network they live, subject to latency constraints and connectivity. Subsequent to the move, we had a network partition event that broke connectivity between the cages, which split each storage pool exactly in half, breaking threshold.

Lessons learned:

- Low maintenance does not mean no maintenance, and rebuilds take time, keep up on them.
- People doing work without context, have no context. The storage admins, or their tools, need to maintain and communicate that context to the people physically working on the machines.
- It does in fact matter where your machines live and how they talk to each other.
- Assumptions are super deadly.

Operations: Philosophy

A few of our guiding principles:

3 Rules of Operations (G-rated version)

- Know Your Stuff (Competence): Know your environment, tools, machines, processes. Do not let anyone play magical “you don’t need to know this” wavy hands with any piece of technology you are responsible for. If you’re responsible for it, you should know how it works.
- Get Stuff Done (Momentum): Don’t be afraid of change; find out what needs to be done and do it.
- Don’t Break Stuff (Diligence): Be careful in your work; know the right time to do (and not to do) things, have a plan and a rollback plan; document.

Bus/Island Rule: The correct level of documentation and skillset cross-pollination has been achieved when anyone on the team, manager included, can get hit by a bus and/or win the lottery and retire to his/her own private island, and the site and team can continue to function.

Corollaries of the Hockey Rule: (100% of shots you don’t take, won’t go in)

- 100% of systems you don’t make changes to, you are much less likely to break.
- 100% of systems you don’t understand, you are going to have a really hard time fixing.
- 100% of obscure yet important things you didn’t write down and share with the team when you worked on them, somebody else is going to waste days or weeks rediscovering.
- Add your own, the possibilities are endless..

Operations: Keep It Simple

1.It's so much easier to manage a lot of things if they are all essentially alike.

Do we have 190PB of object storage hardware deployed? And 46,000 disks? Sure.

But in smaller pieces that can also be looked at as:

- 5 administrative domains consisting of
  - 5-10 storage pools each consisting of
    - 20-30 machines each, containing
      - 45-84 of the same size disk in each of them

All running the same software, managed through the same UI.

2. Standardize as much as possible

- consistent, meaningful naming conventions
- write build checklists & enforce buddy checks so everyone on the team builds the same way
- think about how you would build another 50-100 of the same thing

Operations: Keep It Simple

3. Process efficiency

Time required to deploy a new 30-node 20PB storage pool:

- 15 minutes: Hostname/DNS/IP allocation
- 1 week: Racking/cabling/BMC setup
- 1 day: OS install (remote)
- Excluding all that:
  - 10 minutes: configure storage pool, configure vault, deploy to accessers, send test traffic, done.

Exactly the same for a 5PB, or 10PB, or 40PB pool … it's not the size of the pool, it's the simplicity of setting one up that makes the work scalable.

4. Automation/Monitoring

- The best system installation is the one you have your hands on the least.
- Systems should validate and keep their own configs in a known correct state
- Systems should tell the admins when something goes wrong
- Anything that bites you once, set up automated validation/alerting on
- Trending, trending, trending … if you don't know what's normal, how will you know when things change?

Interesting Events: 2012: Kodak Image Acquisition

In 2011, Kodak Gallery went into chapter 13. Shutterfly agreed to purchase the customer info, and their 5B pictures. They had to be copied out of Kodak's colo by Q3-2012 as the lease was expiring.

Considerations:

How much data do we need to copy? About 9PB

Move the data … how?

- Move their storage to our colo? Nope. We bought the data, not the equipment.
- Network? Also nope.
- Tapes? Super especially extra-nope.

04/2012 Shutterfly acquires Kodak Gallery:

(http://www.wsj.com/articles/SB10001424052702303592404577364271258571262)

Interesting Events: 2012: Kodak Image Acquisition

So how did we do it?

1. Bought 4 pools of our (standard at the time) 4U45 storage bricks, mix of 2T and 3T drives; 12PB raw capacity.
2. Sent them to Kodak's colo.
3. Set them up as 20+6 Cleversafe erasure-code storage pools (33% data expansion).
4. Sent 10 accessers - Cleversafe web layer, 1Gb/s nics.
5. Sent a 10TB flash-based mysql server.
6. Wrote a perl-based migration utility that copied images from Kodak's Isilon architecture into our Cleversafe pools, onsite at the Kodak colo.
7. Migration utility ran *on* the Cleversafe accesser nodes..
8. Ran multiple instances per accesser, sweet spot seemed to about 20-30 instances per accesser
   a. Don't crush the Isilons
   b. Don't crush the accessers
   c. Don't crush the database
   d. Don't crush the storage nodes
9. Eventually reached about 1GB/s consistent throughput, mainly limited by the Isilons and uneven sharding of the Kodak images.
10. Migration farm completely spun up by May 2012, completed October 2012.

Interesting Events: 2012: Kodak Image Acquisition

So that's great, but the data is still in the wrong physical location… now what?

1. We put all 104 storage bricks on a set of trucks
2. Shipped them across town to our colo
3. Brought them up in our colo
4. Replaced the (very few) drives that didn't survive the trip
    a. With a 20+6 encoding scheme, each image is striped across 26 drives, one per machine, and only 20 chunks are needed to recreate the image.
    b. We could have (theoretically) lost 24 entire machines and still not lost the data.
    c. Because we wrote to 4 pools, our failure domains were reduced even further … only 25% of the images were at risk on each pool of 26 machines.
    d. The Cleversafe software automatically rebuilds the missing data when drives are replaced, it's just a matter of waiting.
5. There was a lot more overhead in converting Kodak customers to be Shutterfly customers, but as far as moving the images, that was it. Great demonstration that 'storage' can be portable/mobile/flexible.
6. We beat the November deadline to shut down the Kodak colo, avoiding an extension lease payment.

Interesting Events: 2014: Data Center Move

In 2014, Shutterfly decided to move to a new, less expensive datacenter. In Las Vegas, 530 miles away from our then-current colo in Santa Clara, CA.

The Image Archive object store at the time was: 24 storpools, 622 storage nodes, 28,068 disks, 76.7PB raw, containing about 50PB of user data.

Similar conversation about how does one move 50PB of live user data, in 3 months, without taking down the site, nor buying another 76PB of hardware.

One answer is: 48 machines at a time, on a truck, one per week.

Interesting Events: 2014: Data Center Move

Here is how it went:

1. Build out new datacenter space: Fall 2013 - Winter 2014
2. Every week, starting in April 2014, designated 2 storage nodes of each storage pool (48 nodes total) for transport.
3. Powered them off, put them on a truck, they're off to Vegas!
   a. The site can write to the storage pools with as many at 4 nodes down, so 2 down is above write threshold. Extra housecleaning each week to make sure that was the case.
   b. Shipping was expedited so the nodes were only down for 18-24 hours at most.
   c. Remaining 20-28 nodes in each pool kept taking/serving live site traffic out of Santa Clara.
4. Nodes arrived in Vegas, were racked, cabled, re-IP'd, powered on.
5. Any broken disks replaced on arrival (fortunately, again very few).
6. Cleversafe rebuilding process had a full week to regenerate the data blocks missed by the nodes during the time they were in transit.
7. Approximately ¼ of the way through, we moved some of the accessers as well, in anticipation of having most of the data being served from Vegas as the balance of nodes shifted there.
8. Repeat week after week, until all storage nodes were moved.
9. At about the half-way mark, we shifted site traffic to the accessers in Vegas. There was some additional latency during the middle weeks of the move due to the nodes in the pools being split semi-equally across the WAN link.
10. All nodes moved by July 2014.
11. shutterfly.com was up and running the entire time.

Interesting Events: 2014: Data Center Move

Storage in Motion… as long as the pools are at threshold, they can be written/read no matter where they are.

Disk Failures & AFRs

In a fleet of 50,000 drives, some are going to fail, it's a fact of life. We typically lose 10-15 drives per week. Due to our node-level +6 erasure coding scheme, it is generally a non-event.

We have found some of the more important aspects to be:

1. Keep up on regular maintenance: make sure all broken disks get replaced within a week of failure.
2. Prioritize: for us, there's more potential for site impact with the 240 or so nodes that are actively taking write traffic at any given time, so we provide the datacenter techs with a prioritized list of what to replace first.
3. Keep enough spares: we generally try to keep 30-60 days worth of spares of each disk size deployed.
4. Plan for obsolescence: migrating data is a fact of life; we generally don't expect any data to remain where it is for more than 3 years.
5. Trust but not too much: the n+m schemes we run have astronomical MTTDL numbers, but it is in fact possible to have the exact wrong combination of drives and nodes go offline and therefore lose threshold. Keep your nodes healthy and your drives replaced.

Partial fleet stats through EOY 2015:

			Lifetime		
Year/Size/Model	Deployed	Failures	Fail %	YearsSvc	AvgFail%
2011 2TB HDS723020BLA642	7470	749	10.03%	5	2.01%
2012 3TB HDS723030ALA640	9360	436	4.66%	4	1.16%
2014 4TB ST4000DM000-1F21	8520	187	2.19%	2	1.10%
2014 5TB ST5000AS0011-1L5	1440	26	1.81%	2	0.90%
2014 3TB ST3000NC002-1DY1	1248	16	1.28%	2	0.64%
2015 4TB ST4000NC000-1FR1	7560	10	0.13%	1	0.13%
2015 8TB ST8000AS0002-1NA	2520	11	0.44%	1	0.44%

SMR

We love it, for the two of our four use cases that are essentially write once, read for a while, update never: the long term archive and the on-disk backups.

Long term archive:

1. Images are written until the storage pool is full
2. Images are never deleted, nor updated.
3. Images are read for the first 30-60 days, then seldom referenced again, except for Xmas each year.
4. Backups are essentially the same, except we re-verify them monthly.

Deployment to production process (now standard for all new devices)

1. First testing Summer 2014
2. Very close coordination with vendors (Seagate, Cleversafe)
3. Weekly disk log dumps for analysis by Seagate
4. Install 1 disk in 1 inactive (read-rarely) pool, monitor rebuild & read traffic
5. Populate 1 full node, 60 drives, in 1 inactive pool, monitor rebuild & read traffic
6. Deploy 1 full 18+6 (1440 drive) pool, copy older data into it for validation
7. Direct 1% of live uploads, 2nd parallel copy, to new pool
8. Increase to 5%, 10%, 15%

Lesson learned:

- do not ever delete data from these while the site is writing to them. Ouch.

Image Compression

JPEG images are already highly compressed. Using gzip or other compression on them doesn't have much effect.

There are lossy compressions available, but the Shutterfly public statement is we will always keep the original version of everything our customers upload to us. So lossless it is....

In 2014 we discovered packjpg (http://www.elektronik.htw-aalen.de/packjpg/) and found it provided about 30% compression on average. Which doesn't sound like a lot, until you're talking about 65PB of data … that's 20PB of hardware and software licensing costs that can be avoided, plus future savings.

2 main downsides:

1. Slow (relatively): can only compress at about 1MB/s per 2Ghz hardware thread.
2. Output is not a JPEG, so a compressed image has to be decompressed, at 1MB/s, before it can be used by the site. The images currently average just over 2MB each, so there's a 2-3 second latency being added.

Image Compression

The compression latency can be overcome by running it as an async job, outside of the upload data flow.

The decompression latency can't really be overcome in full at present; so we are mitigating the effect by only using the compression on our backups, and we are considering how best to identify the least likely to be used images in the primary archive and compress only those.

The not-a-JPEG issue is addressed by our caching tier, which already does various types of data translations so the upstream applications don't have to.

So: timing… if you need to compress 65PB at 1MB/s, how long will it take?

Turns out, just over 2,191 years.

Impractical, obviously. Here's a handy chart:

MB/s	GB/hr	TB/day	archiveTB	days	months	years
1	3.5	0.1	65000	788859.3	26295.3	2191.28
5	17.6	0.4	65000	157771.9	5259.1	438.26
50	175.8	4.1	65000	15777.2	525.9	43.83
250	878.9	20.6	65000	3155.4	105.2	8.77
500	1757.8	41.2	65000	1577.7	52.6	4.38
1000	3515.6	82.4	65000	788.9	26.3	2.19
2000	7031.3	164.8	65000	394.4	13.1	1.10
3000	10546.9	247.2	65000	263.0	8.8	0.73
4000	14062.5	329.6	65000	197.2	6.6	0.55
5000	17578.1	412.0	65000	157.8	5.3	0.44

Image Compression

Our daily ingest ranges from 30TB-100TB/day at present depending on the time of year, so deploying a compute capacity of 1000MB/s will cover us for daily work, but won't help with the backlog. The desired time for completing a re-backup of the entire archive to disk (the images were previously backed up to tape) was 'as soon as possible, while not being inordinately expensive'.

Deploying 3000-4000MB/s of compression should run about 6-8 months, without leaving us far over-provisioned.

Options we've explored:

1. AWS/GCP: lots and lots of cores available, and we could use spot pricing, and we could stuff the 65PB into either of them for free but … we unfortunately also need to pull the 40PB back out. The math on data egress pricing, about $.04/GB, works out to about $42K/PB x 40 =$1.6M. Plus compute time, and temporary storage. Ouch.
2. Dense compute chassis: We took a look at the SeaMicro 64-node chassis. Each E3-1265L v3 @ 2.50 GHz blade was good for about 8MB (8 packjpg instances in parallel), total for the chassis of 512MB/s. But then AMD bought SeaMicro and shut it down.
3. We explored having packjpg ported to one of the FPGA platforms. Unfortunately, we have no in-house FPGA development experience, and it would have been a one-off project for one of the contract development houses. We did an RFQ for a guaranteed 3000MB/s of compression capacity, and it came out higher than what we could do in-house with generic Xeons.

Image Compression

(cont'd)

4. The latest Dell R430 with dual E5-2650L v3 1.8GHz, comes out to 48 hardware threads at about $90 per thread (3/2016). 3000 of them (hw threads) would be about $270K.
5. This very cool article about using older generation Xeons, specifically the E5-2670v1, available secondhand, pushes the price down to about $33/hw thread (assuming custom built boxes at around $1K each) http://www.techspot.com/review/1155-affordable-dual-xeon-pc/
6. Our storage nodes are, in fact, just Linux boxes (running a Debian live-image), with lots of disks. As long as the systems are mostly idle, which is true most of the year for everything except the active upload pools, there's nothing wrong with getting some extra work out of them:

						Machine		
				Machine		usable		
Machine			Threads/	hwthread	Theor.	hwthread	CPU Act.	
Count	Sockets	Cores	Core	count	MB/s	count	Usbl MB/s	CPU Type
585	1	4	4	4	2340	1	585	X3430 @ 2.40GHz
35	2	8	16	32	1120	8	280	E5-2650 0 @ 2.00GHz
178	1	6	12	12	2136	4	712	E5-2620 v2 @ 2.10GHz
189	1	4	1	4	756	1	189	E5-2603 0 @ 1.80GHz
5	1	4	1	4	20	1	5	E5-2609 v2 @ 2.50GHz
9	2	4	2	16	144	16	144	E5620 @ 2.40GHz
4	2	8	2	32	128	32	128	E5-2690 0 @ 2.90GHz
20	2	14	2	56	1120	28	560	E5-2695 v3 @ 2.30GHz
62	1	4	2	8	496	8	496	E3-1265L v3 @ 2.50GHz
1087					8260		3099	
Machine					Theor.		Usable	
Count					MB/s		MB/s	

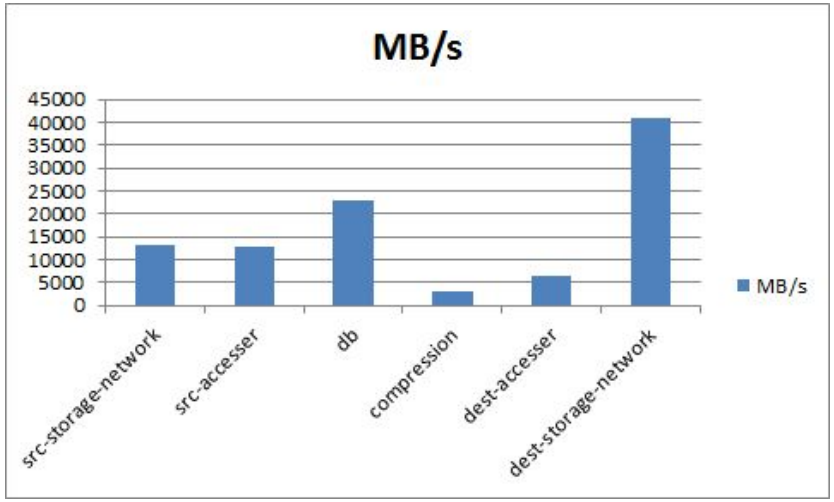
Image Compression

The actual hardware we will end up using for the bulk of the compressions is still somewhat in flux, and will consist of a hodgepodge of what we have already available supplemented by some amount of new machines.

The actual compressions are performed by a simple distributed worker farm running against a set of Redis queues, with some realtime (self)-tunables to limit the workload per hardware class, grouping, and source/dest rate limiting. The workers are distributed in custom .rpm and .deb packages. The compression farm itself is expected to be be the gating factor, rather than Redis, or the source or destination storage pools, followed by the accesser layer, which can be increased fairly easily.

Theoretical throughput of each step. Excluding the disk farms on each end as they are 3000-5000 disks @ 100MB/s per.

	MB/s
src-storage-network (104 nodes @ 1Gb)	13312
src-accesser (10 @ 10Gbit)	12800
db (10K updates/sec)	23000
compression	3000
dest-accesser (5 @ 10Gbit)	6400
dest-storage-network (32 @ 10Gbit)	40960



Future Plans

A few of the things in the pipeline:

Sub-millisecond high-resolution thumbnails: the bulk of the images displayed to the user while they are designing products using our site are not the actual full-size images they uploaded, rather, we generate at upload time a down-sample version that is about 10% the size of the original. We presently serve those with a median latency of 2-5ms from our FC SAN silos. We are testing an erasure-coded clustered filesystem based on Intel's DPDK low-latency toolkit to see if we can get the thumbnails down into the 500-900usec range.

On-demand storage allocation: at present our minimum shard size for directing customer traffic to any given storage destination is 1% (100 storage locations). It is also, for historical reasons, tied to midnight UTC, and we can write only 3 parallel copies of any given image. We are developing a real-time storage allocation service that will allow us to spread traffic across at least 10,000 storage locations, will be updateable in realtime, and will have some degree of automatic failure detection so no humans will be needed to do said realtime updates.

Conversion to named buckets: Amazon S3-like named buckets were not available in the early releases of Cleversafe; a REST API was provided but we had to maintain an external metadata linkage of the images to the object id. Given that our image names are already unique, that's extra overhead we don't need. Most of the 29B images will need to be converted, which means a lot more migrations.

Future Plans

(con’td)

Where do old disks go to die? When we finish the current round of tech refresh migrations, we’ll have about 16,000 5-year old 2T/3T drives that have not actually failed as of yet. We plan to put them into newer, denser chassis, bump up the erasure coding scheme to +10 nodes and continue to use them for non-critical data, tracking their long term AFRs along the way.

Virtual accessers: our present Cleversafe access layer is presently 10Gbit appliances. We plan to virtualize those up to 100-200Gbit per physical host.

Platform diversity: we have numerous initiatives in flight to compare and contrast how our workloads fare on different object platforms. Especially for the short term and long term copies of the fullsize images, we have found it useful at times to *not* be using the same hardware and software for everything. Different platforms have different failure modes and a corner case that bites one vendor may not affect others the same way.