Big Data Challenges in the Era of Data deluge: Practical considerations

Ilya Volvovski, Cleversafe Inc

Building production quality systems

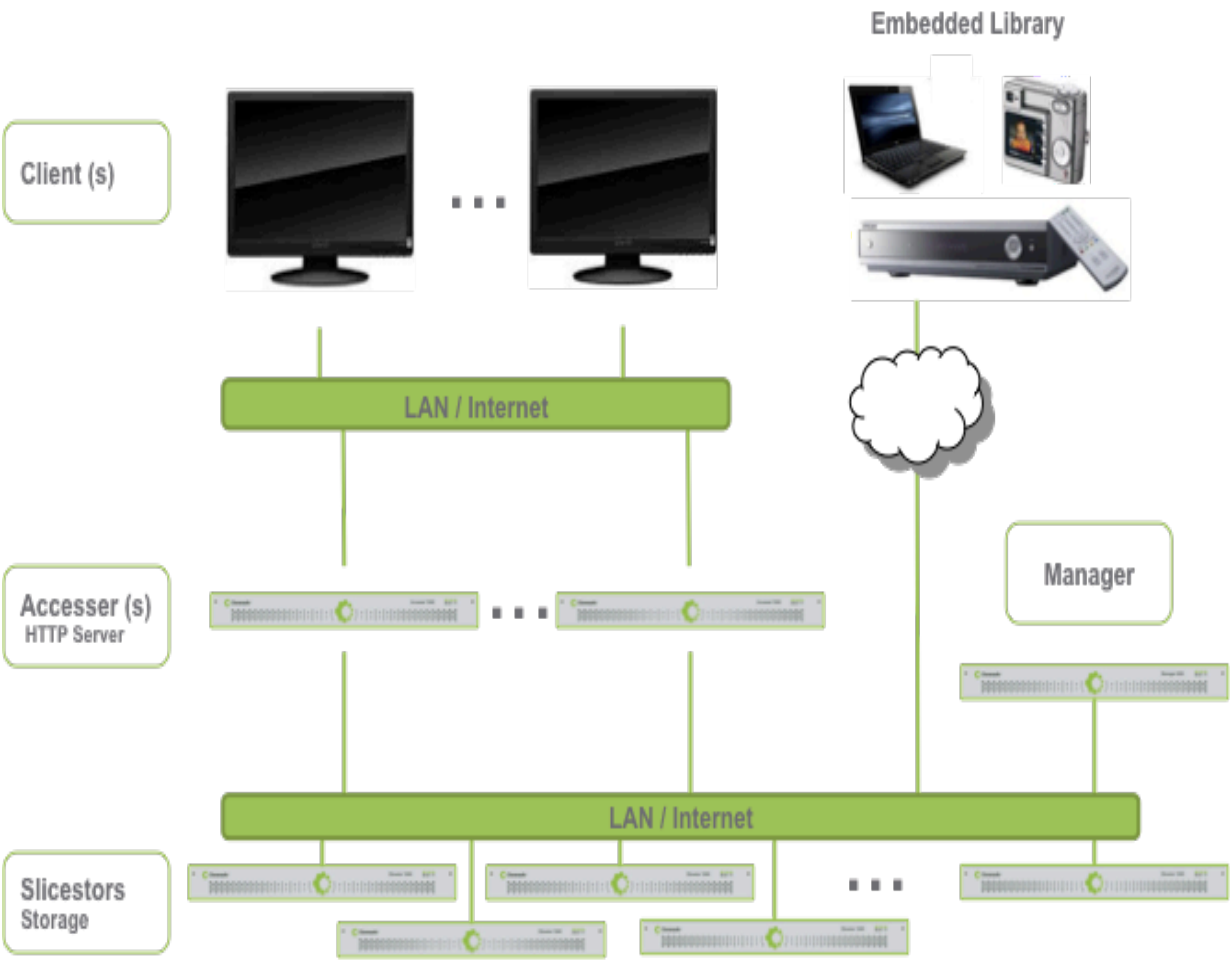
Defining and designing a system to meet system requirements is a long way from having a high quality, operational and usable production system.

Cleversafe/Scality/Amplidata/
EMC
Microsoft Azure

Illustrated with Cleversafe dsNet

Some project are successful even though a picture looks scary

System architecture



Accesser

- The **Accesser** device is used by the user's application to connect to dsNet over the network
  - Talks many protocols
  - Stateless
  - Performs IDA, encryption/decryption. High CPU use
  - 10Gig network
  - Typically utilizes a load balancer for workload distribution.
  - Software
    - Route IDA artifacts
    - Read/Write intelligence
    - Stateless
    - HTTP/REST API

Slice Store

The **Slice Store** device – actual storage (IDA byproducts)

- Stores data, ultra dense up to 336 TB in a single box and growing
- Hot swappable disks
- 10Gig network
- Software:
  - Store/Retrieve data
  - Manage disks
  - Rebuilding
  - Proprietary protocol

Manager

- System Configuration
- System Maintenance
- HTTP REST/API based
  - Anything that could be done through UI could be done through API for integration
- Has dedicated agents on each node
- Stores stat data locally (short term)
- Stores data in regular distributed containers (long term)

Architectural principles

- Efficient reliability based on IDA
  - Data/ metadata/index (no exceptions)
- Scalability on every level
  - Data
  - Configuration
- Threshold security
  - confidentiality based on threshold number of components, an intruder needs to compromise to break security
- Performance in every layer
  - Maximum concurrency based on asynchronous model
  - Efficient and flexible backend storage

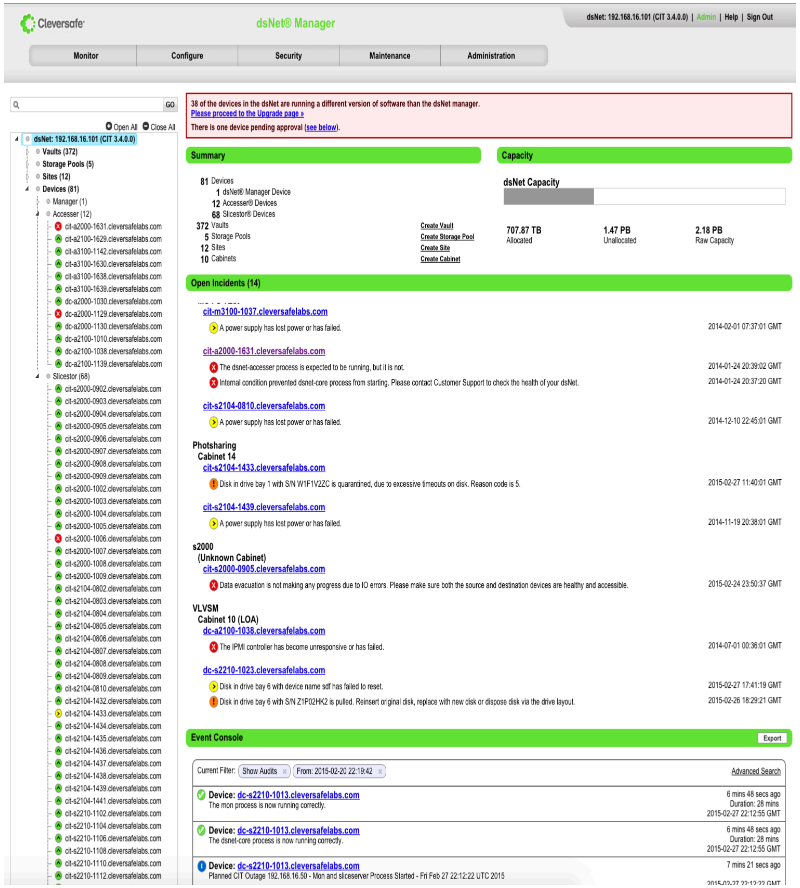
Authentication

Support different types:

- Internal Username/Password
- Externally managed Username/Password
- Public Key Infrastructure
- Active Directory or Open LDAP server
- Open for DIY
    - Implement
    - Integrate

Scalable UI

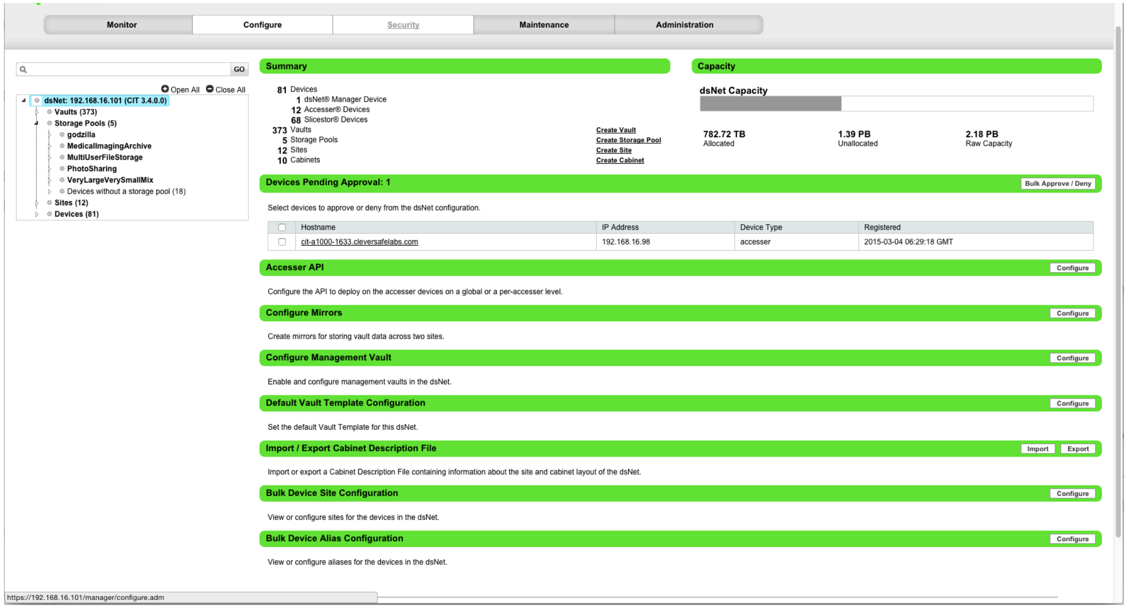
- Secure UI
- Role based restricted access
- Configurable
- Major Features
  - Container management
  - Storage management
  - System Customization
  - Monitoring
  - Administration



Templates, search, user defined view, persistent views, compare views, filtering

Provisioning

- Easy to understand and manipulate
  - Grouping
  - Pools
  - Templates
- Hardware provisioning
  - Bulk
  - Secure
- Logical
  - Storage container creation/deletion
  - Limits/quotas



Management API

Anything that could be done through UI, should be available through Manager API:

- To automate
- To integrate

dsNet Namespace

- Each object name consists of:
- 22 bytes routable name
  - Container UUID
  - Storage type (hint the storage implementation)
  - Generation ID (expansion factor)
- 24 bytes storage random internal
- Slice Name is an object name + IDA index(2 bytes)
- Total addressable name is 48 bytes
- Namespace is assigned by ranges and could be changed dynamically due to space reallocation.

Realistic IDA configurations

Immediate versus long term reliability

Guaranteed level (W/T/WT) - Write threshold

16/10/13	26/20/23	36/20/23

Functionality

Unsung hero!!!

Interfaces

- Simple Object (SO)
  - Accept data, return an opaque long ID
  - Client is responsible for maintenance
  - The most efficient and scalable
- Named object (NO
  - Accept both data and names
  - Client access by name
  - Listable or not listable
- Index support for all (in progress)
  - Ability to find data based on metadata
  - Scalable, collision resistant
- S3, Openstack
- CIFS,NFS, HDFS
- Growing list, open for extension

Internal dsNet I/O operations

- Write (with hidden revisions)
- Read (always the latest revision)
- Delete
- List

And this is all for I/O!!

Object Segments

- The original object is split of segments:
- Addressable
- Faster TTFB
- Random read
- Higher concurrency
- Ability to resume
- Segment size
  - Bigger → fewer addressable objects
  - Smaller → better response time

Slice Store IO operations

- Write operations is consist of:
  - Checked write
  - Commit or rollback
  - Finalize (after successful commit)
  - Undo (after failed commit or in case of delete)
- Read
  - Reads returns 0,1 or more revisions. Client needs to decide which revision is the latest restorable
- List_Range (used primarily for rebuild)

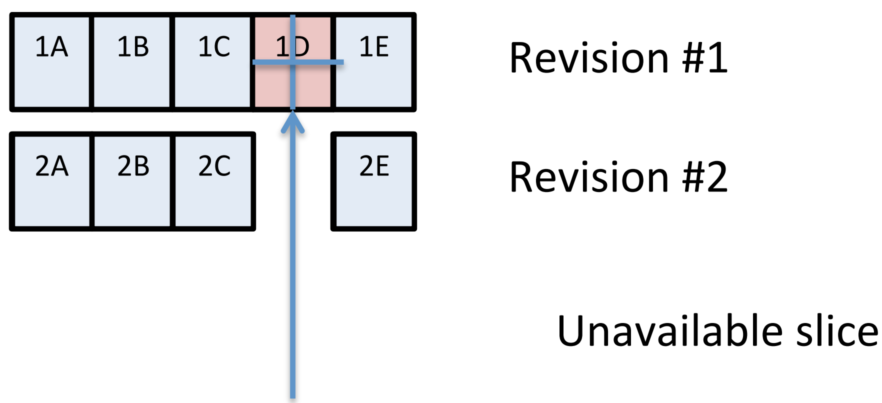
IO operations design principles

- No locking, optimistic concurrency on write
  - Read/update, success if object has not changed
  - Retry if the assumption was incorrect (object has changed)
- Consensus on write
  - The majority of servers have to confirm the accepted revision
  - If none achieved majority retry with back-off
- Error if consensus can't be achieved due to node's unavailability (temporal unavailability)
- Delete has the same semantics as a regular write on Slice Store (revision for consensus)
- List doesn't have to be precise, used for rebuilding only and resolved on rebuild attempt

Three phase commit

5/3 example

[illegible]



Will be rebuilt/removed during rebuilding

Animation, may look bad in PDF

Rebuilding

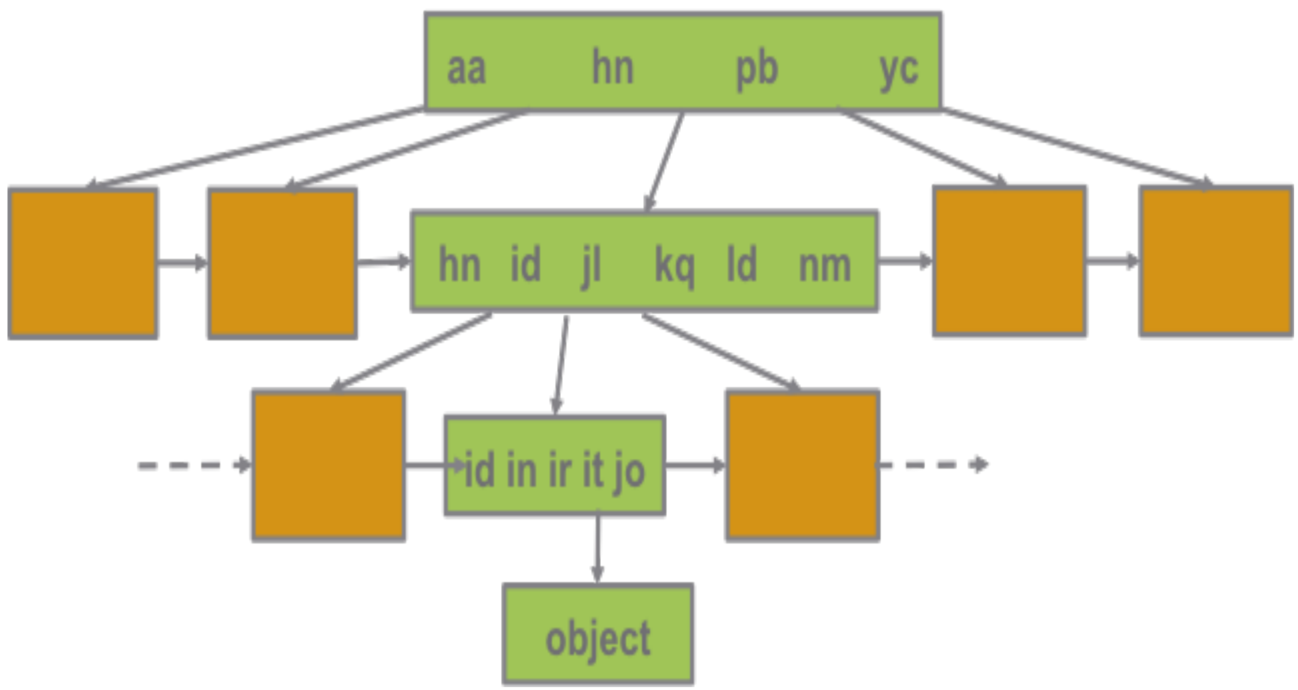
Ability to effectively repair lost system elements is one of the most important features of the storage system. Includes the following critical elements:

- Discovery (scanning)
  - Should be fast enough
  - Should have little impact on overall system performance
  - Scale with storage size
- Repair
  - Data recovery
  - How to make it secure in IDA settings

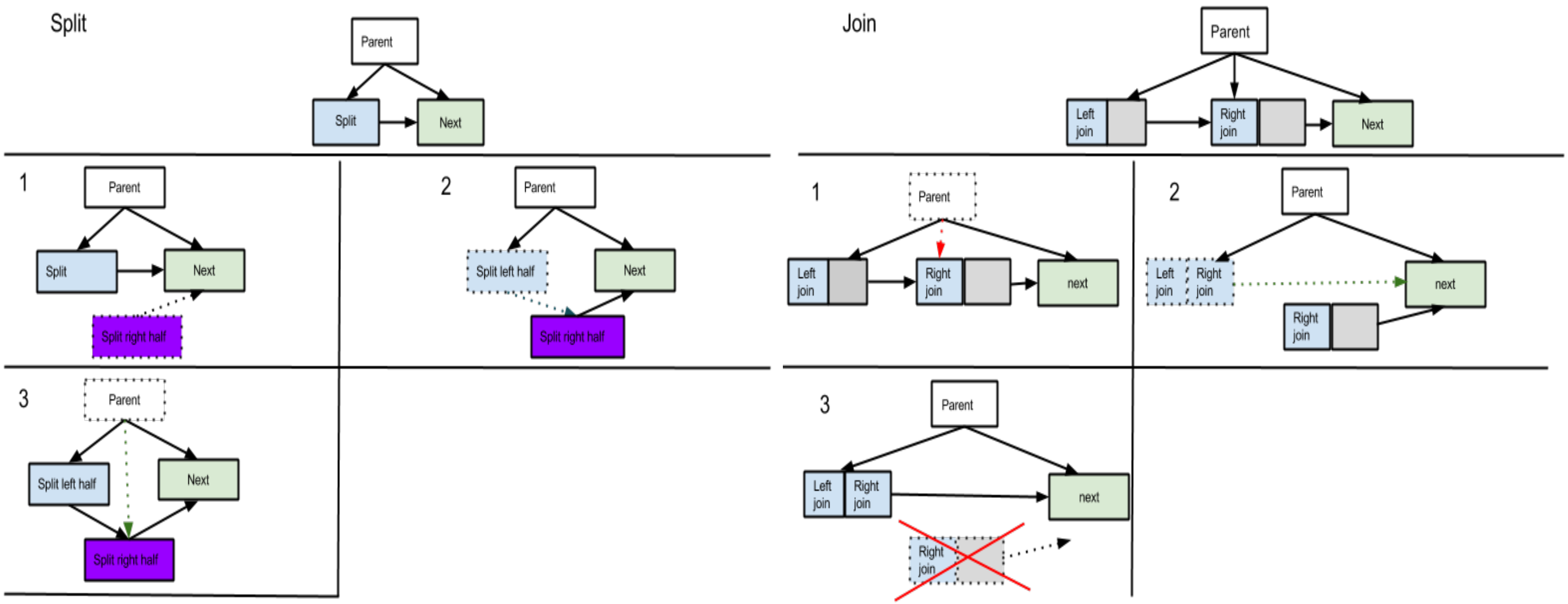
Rebuilding animated

Scalable Dispersed Index

- Optimistic concurrent index structure
  - like B-tree, but lockless
  - Similar to concurrent skip list but uses batching



Lockless index restructuring



Storage System Maintenance

Maintenance is a complicated multi-facet process:

- Facilitate proactive equipment replacement
- Perform system upgrades
- Help to understand system behavior
  - Identify bottlenecks
  - Identify and alert observed or potential components' failures
  - Auto correlate events

Limping components

- Components
  - Drives
  - Cables
  - Switches
  - Any hardware could misbehave
  - Software
    - Zombie process
- Remove permanently limping components from the critical path

Disk Management

- Detect
- Identify
- Tolerate
- Isolate
- Notify
- Remove/Replace

- Predict
- Migrate
- Remove/Replace

Danger of false positives

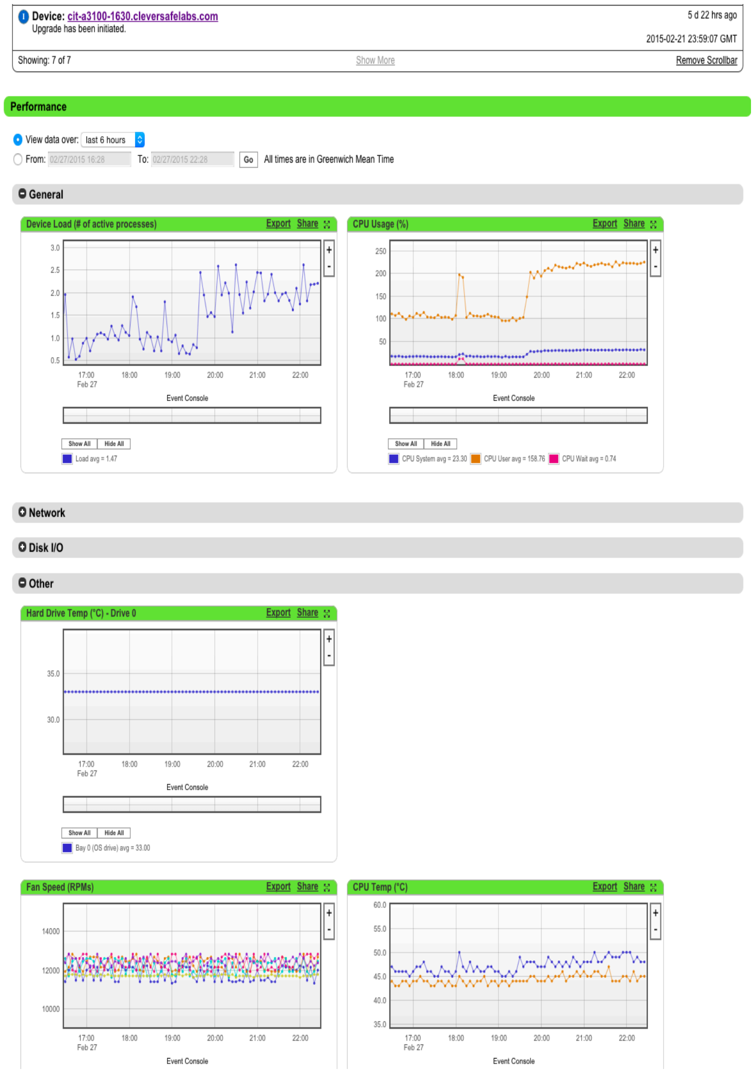
Failing disk discovery

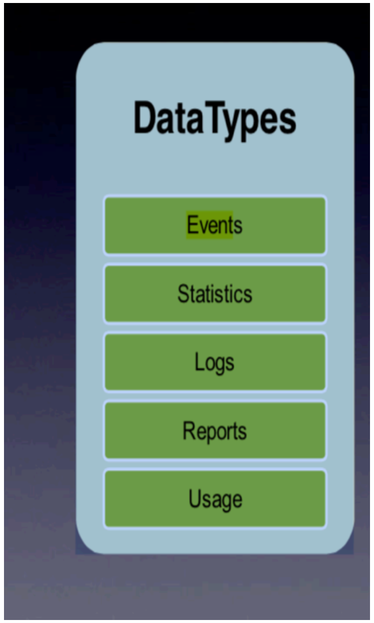
- S.M.A.R.T
  - Choose attributes which predict failures
    - Reallocated Sectors
    - Spin Retry Count (impending mechanical problems)
    - Pending Sector Count (unstable to be remapped)
    - … and others
  - Make sense of this information
    - Not easy, no standard interpretation, vendors differ
- Software heuristics
  - Kernel level (inability to mount file system)
  - Application level (abnormal operation execution time, too many errors)

Failing disk replacement

- Simple
  - Zero maintenance
    - Remove, replace and forget (little training)
  - Hot swappable. No need for coordination
- The least impactful
  - Salvage as much data as possible (rebuilding is expensive).
    - but don’t try too hard

System monitoring

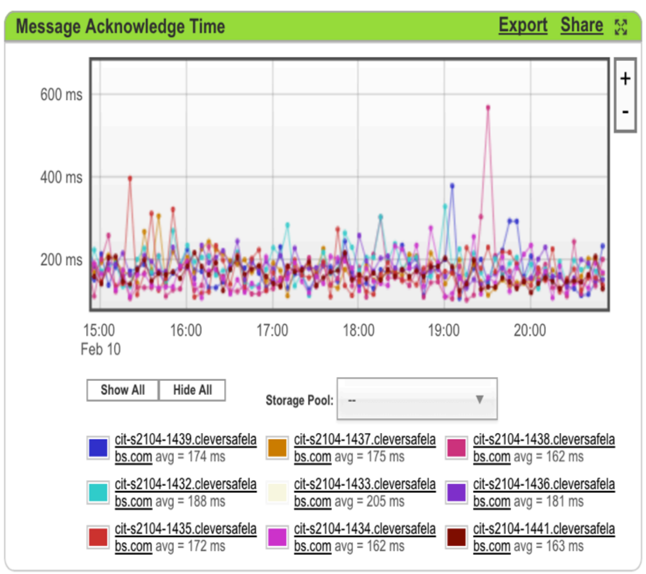




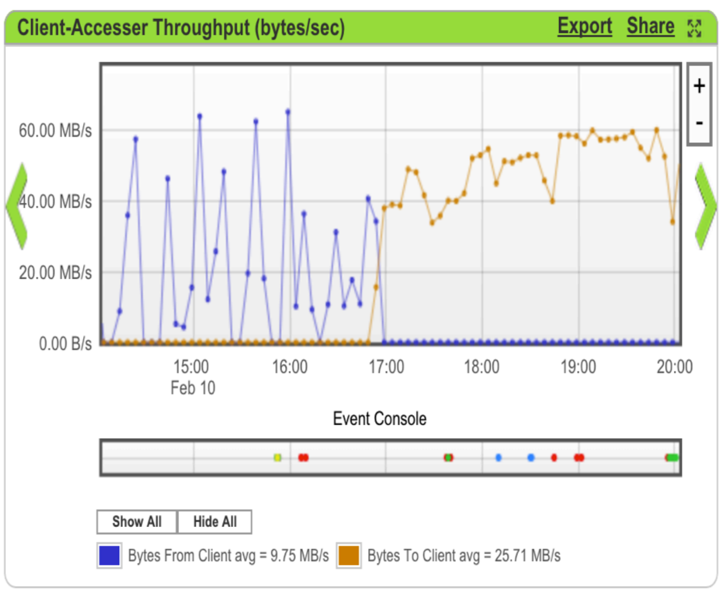
Storage system should be able to proactively convey operational details in consumable and actionable form.

Performance indicators

- Client throughput/latency
- Network throughput
- Message Ack times
- Disk IO



ETC.



Failing and Limping hardware performance impact

- A failing component is easy to detect and eliminate
- A limping component is a component that works but
  - May behave like a zombie
  - System component is under attack
  - Slow due to temporary condition (e.g. Java GC, disk housekeeping)
  - Could be transient or permanent
- System has built-in redundancy, use it wisely

Performance

- Do concurrently as many operations as possible
- Report success to a client as soon as contract is fulfilled
- Setup acceptable wait for component's operation completion based on historic averages
- Retry in order to reduce dependence on a slow component

Troubleshooting capabilities

- How easy
  - to find a faulty element
  - to isolate a faulty element
  - to replace a faulty element
- Detect bottlenecks
  - To change system configuration
  - To improve networking
  - To change hardware

System upgrade

Zero downtime upgrade is absolute must for a production quality storage system

- Systems evolve, features are added, bugs are fixed
- As much parallelism but not too much
  - Maintain availability and reliability during upgrade
  - Find most data independent elements to upgrade in parallel
- Support online format changes
  - Data migration: mostly metadata (easy at startup) but sometime data format (needs to be executed in background)

Defensive practices

- Never allow mass non-user initiated cleanups
- If something looks strange it is probably wrong
  - Keep reasonable caps on all assumptions
    - leftovers after crash could not exceed X
  - Never delete data in large chunks
    - How much data could be corrupted?
    - How much could be unreadable while disk is operational?
    - How many times you could experience a rare condition (such as across node checksum don't match)

Defensive practices

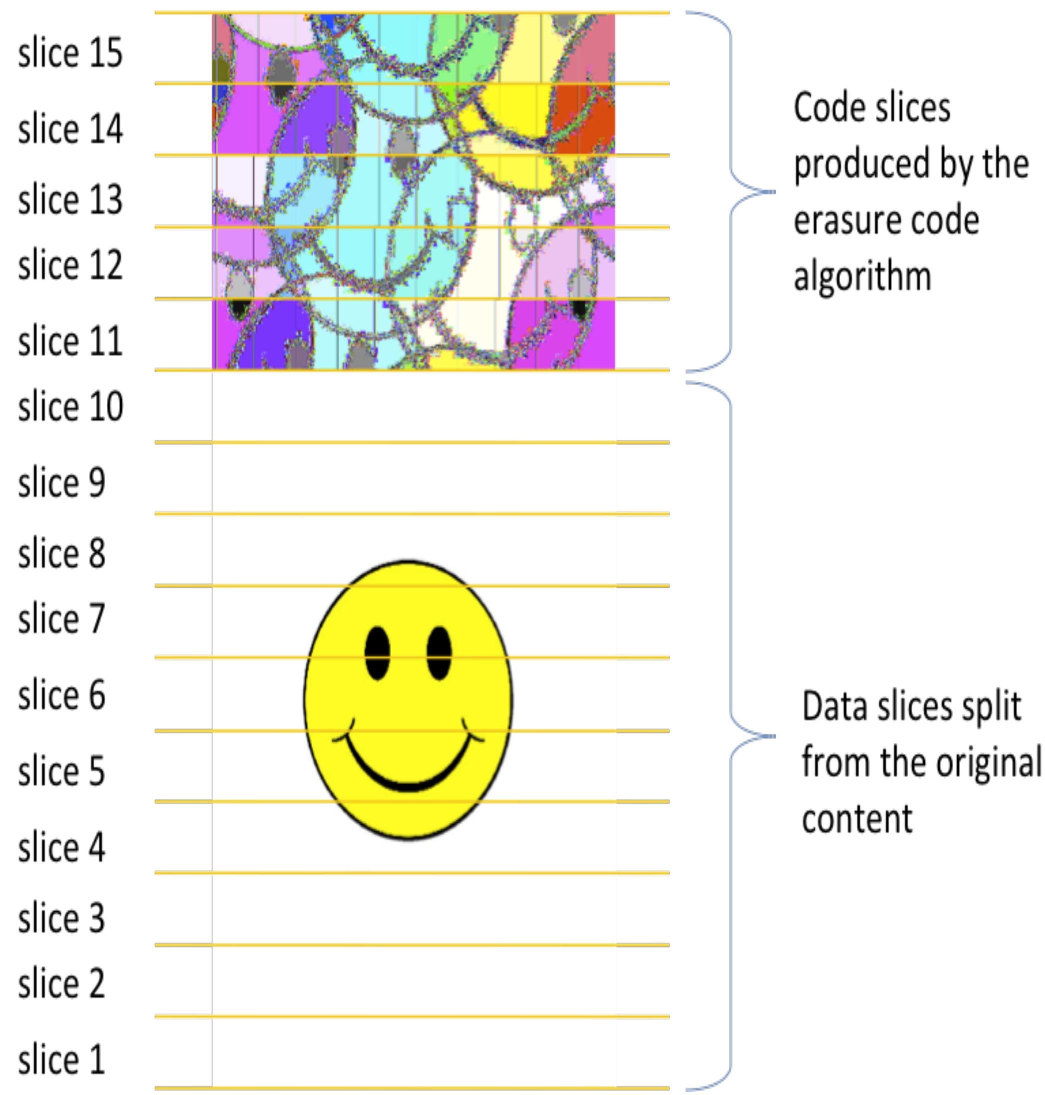
- Create recovery procedure
  - Especially after crash
  - Mechanism to determine crash condition
  - Always assume the worst
  - Checkpointing
- UPS (battery backup)
  - Power outages do happen
  - Sync mode is prohibitively performance expensive

Securing data without key management

IDA generated data is not secure

- Reveals information from the original data (unless it is encrypted)
- Make disks vulnerable to theft
- A significant expense for safe destruction

IDA leaks information



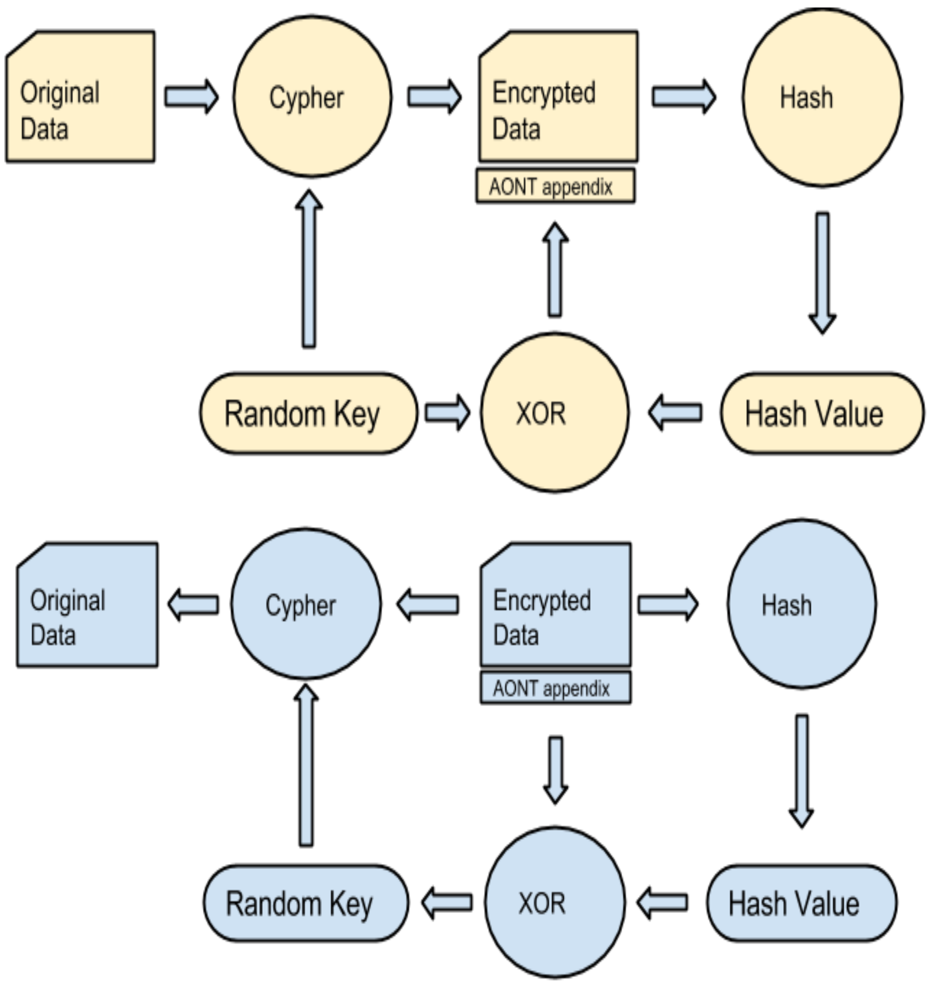
Examples courtesy of Jason Resch (Cleversafe)

A tree encoded with 15/10 IDA

Input to be encoded

After Error Coding

AONT with IDA



- Encrypt data with random key
- XOR with hash of encrypted data
- Add the result to data
- This is the final package and data could not be recognized

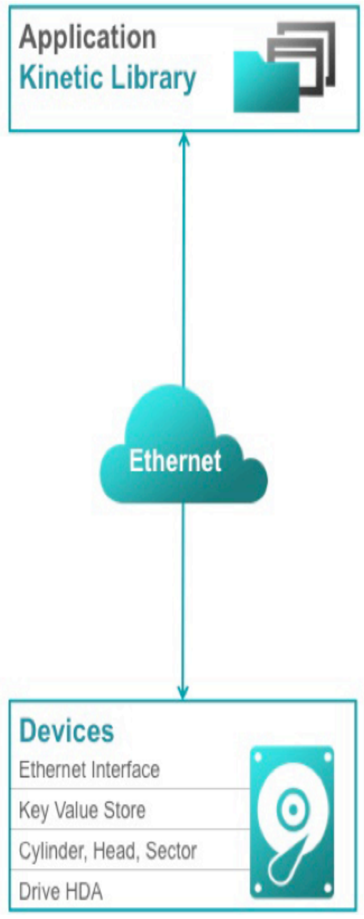
What else is going on

Storage hardware trends

- Rotating Perpendicular Magnetic Recording (PMR) drives do not become much faster, but become more dense
- SSD become cheaper and less write weary, could last many overwrites. Still more expensive
- New rotating Shinged Magnetic Recording (SMR)drives increase capacity of magnetic drives. Zone overlap, data can't be overwritten in place
    - Device managed mode (looks like a normal drive)
    - Host managed mode (host is responsible for correct access pattern to an SMR drive)
    - Emerging standards for cross vendor operability
    - Could be very efficient in specialized storage systems
    - 10TB SMR drives is today's reality, 16TB will be available very soon

Object Storage disks

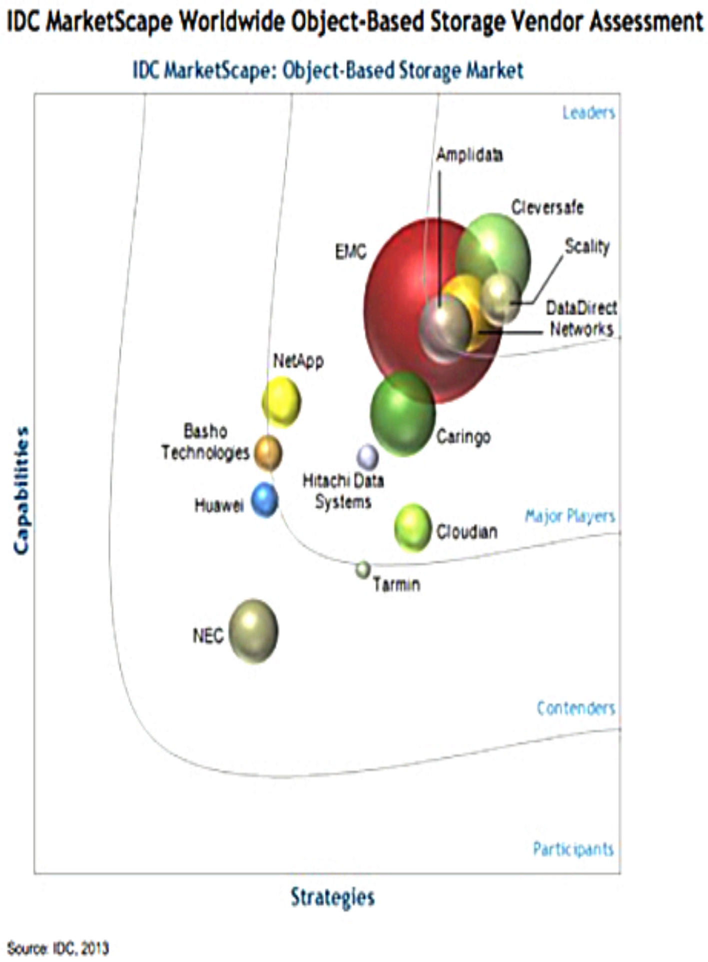
- Kinetic Open Storage Platform (Seagate)
  - Key/Value store with version support
  - Provides put/get/delete with version functionality
  - Network addressable
  - Provides data at rest security (PKI)
  - Removes artificial for Object Storage block/sector to object mapping (done inside a disk)
  - Drive-To-Drive Data transit (rebalancing between drives without other entities involvement)
  - Has full application stack inside (Kinetic is not open)
  - Improves performance

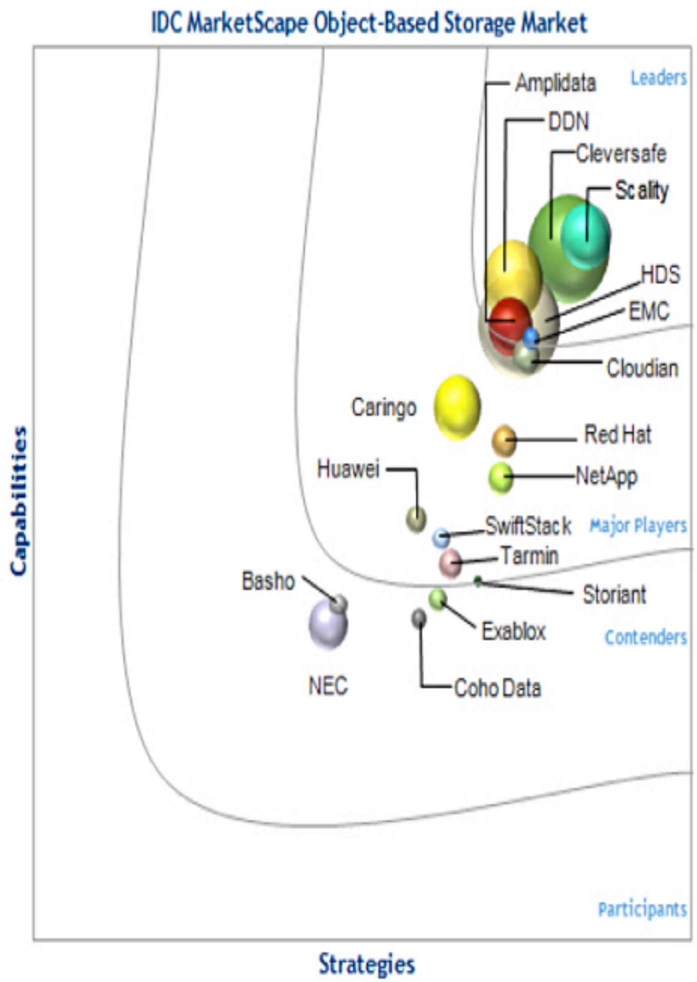


Object Storage disks (cont)

- Built-In Ethernet and Key/Value store
  - It is what a lot of object storage application need
    - Remove the necessity to compile one representation into another
  - Great promise
- However
  - May not be compatible or efficient with Object STorage required semantics
  - Much harder to overcome limitations compared with DYI approach

Who is who in Object Storage world





Sources

1. https://www.backblaze.com/blog/hard-drive-reliability-update-september-2014/
2. RABIN, M. O., Efficient dispersal of information for security, load balancing, and fault tolerance. Journal of the Association for Computing Machinery 36, 2 (April 1989), 335–348.
3. Shamir Adi, How to Share a Secret, 1979 http://cs.jhu.edu/~sdoshi/crypto/papers/shamirturing.pdf
4. Reed-Solomon coding in depth introduction: http://web.eecs.utk.edu/~plank/plank/papers/CS-96-332.pdf
3. Disk failures types http://queue.acm.org/detail.cfm?id=1317403
4. Resch J., Volvovski I. , Reliability Models for Highly Fault-tolerant Storage Systems
5. CAP Theorem: http://www.infoq.com/articles/cap-twelve-years-later-how-the-rules-have-changed
6. Super computers performance: http://www.computerweekly.com/news/2240148760/Supercomputers-will-reach-exascale-speeds-within-decade
7. Lehman B, Yao Bing, B-link trees: http://www.csd.uoc.gr/~hy460/pdf/p650-lehman.pdf
8. AONT-RS: Blending Security and Performance in Dispersed Storage Systems https://www.usenix.org/legacy/event/fast11/tech/full_papers/Resch.pdf
9. IDS report: http://www.theregister.co.uk/2013/11/27/idcs_objectscape_pretty_as_a_picture/

Thank you !