SFD10

Hedvig

*Avinash Lakshman - Founder & CEO - https://www.linkedin.com/in/avinashlakshman

Co-authored Amazon Dynamo
Then did Apache Cassandra while at Facebook

“the fate of glass is to break”
believes the fate of traditional storage is to disappear

*Rob Whiteley - VP, Marketing

business innovation requires flexibility
business innovation - business executives
time-to-market - developers
flexible infrastructure - IT Ops / DevOps

Today’s storage challenges
- doesn’t scale out and back predictably
- can’t easily leverage hardware and cloud innovations
- can’t change or adapt quickly
- can’t inherently survive DC failures
- doesn’t deliver performance at low cost
- doesn’t simplify and automate IT

Hedvig is SDS
- standard servers + hedvig software = hedvig distributed storage platform
[SFD10_Hedvig_WhatIs.jpg]
- software completely decoupled from commodity hardware
- application-specific storage policies
- automated and API-driven

Hedvig’s 7 core capabilities
- seamless scaling with x86 or ARM (haven’t seen an ARM-64 deployment yet)
- hyperconverged and hyperscale architectures (can mix and match in the same cluster)
- support for any hypervisor, container or OS (Xen, KVM, HyperV, ESX, containers, OpenStack, bare-metal Windows or Linux)
- block (iSCSI), file (NFS) and object (S3, SWIFT) protocols in one platform
- enterprise features: dedupe, compression, tiering, caching, snaps/clones
- granular feature provisioning per virtual disk
- multi-DC and cloud replication

Hedvig components
- storage service - forms elastic cluster using commodity servers and/or cloud infrastructure
- storage proxy - presents block and file storage; runs as VM, container, or bare

metal
- RESTful APIs - provides object via S3 or Swift, instruments control and data plane

How it works
- create and present virtual disks to the application tier
- hedvig storage proxy captures and directs I/O to storage cluster
- hedvig Storage Service distributes and replicates data across nodes
- the cluster caches and balances across nodes and racks
- the cluster replicates for DR across DCs and/or clouds

Ideal use cases
Traditional
- server virtualisation
- backup and bc/dr
- VDI
New workloads
- production clouds
- test/dev
- big data/IoT

*Bharat
Hedvig architecture whiteboard

Storage proxy responsibility redirects requests from VM on compute node to backend storage
Virtual disk is the primary mode of abstraction
Proxy masquerades as iSCSI or NFS
Proxy talks to backend via programmable APIs

Proxy is logically 1, but is physically an HA pair
Tracks performance of server nodes in terms of latency, can intelligently route to closes node based on latency

1. Replication Factor (RF) - how many copies of virtual disk on backend (supports 1 - 6, default 3)
Quorum: [RF/2] + 1
2. Replication Policy
- Agnostic - anywhere you see fit
- rack-aware
- DC-aware
3. Dedupe, compression, client-caching (cache the data blocks for this virtual disk at the proxy)

Containers and storage pools
Virtual disk is chunked up into 16GB

Every 3 disks grouped into a storage pool (logical entity)

You can scale out and scale in

nodes in the cluster - demarcation can be racks or DCs

HBLOCK (Data)
- replication
- locally storing data

Pages (metadata)

Writes
1. Assign Replicas to container
2. write to Hblock
3. Update pages

Reads
Metacache
Block cache (deduped)

DedupeCache
Global dedupe
inline

*Abhijith Shenoy - https://www.linkedin.com/in/abhijith-shenoy-59385b6
Demo - proviisoining, snaps & clones, automation demo

[comment - UI needs some work still]

*Gaurav - Docker container deployment demo and DC outage demo

*Chris Kranz - customer field deployment whiteboard
@ckranz

latency far is 10ms in this example