SFD8 - Day 2 - INFINIDAT

*Randy Arseneau - CMO - Company Intro and Background
“Moshe Version 3.0” - Moshe Yanai (Founder and CEO - https://en.wikipedia.org/wiki/Moshe_Yanai)

“our modest goal: store humanity’s knowledge, forever”

250+PB deployed
Cloud Providers, Financial Sector, Academic, Media, Internet, Retail

A few use cases
Better, Faster, cheaper
Consolidation ratios
High Resiliency

3 weeks of burn-in prior to shipping to customers (minimum)
Over 100PB in the Test lab in Israel

Design principles
- provide the right infrastructure for current and future data explosion
- add onboard capabilities to leverage and optimise use of data
- provide superior mechanism for data protection
- overthrow existing paradigms of storage-application integration
- support unlimited use cases by eliminating cost constraints

Competitive Posture
Focus on array consolidation
- 2:1 to 4:1 array consolidation ratios
- 6:1 floor space
- equal / lower latency
- 65 - 80% cost reduction

Primary workload
- DB / Analytics
- VMware
- KVM/OpenStack
- Big Data
- Backup - replacing a bit of Data Domain in the marketplace

*Brian Carmody - CTO - Solution Deep Dive

Infinibox Architecture

Why another storage company?

Cost per Genome example - “Genomics” has become a data storage problem

- emerging economies need to be able to afford this as well as first world

"The finest storage system that has ever been created"

Virtual User Address Space
- VUAs comprise the entire address space visible to users
- any volume - standard, thin or snapshot - is equally represented in the system as a sequence of VUAs
- VUA space is, in practice, unlimited

Sharing the address space
- the VUA space is partitioned across a number of Virtual Units (VUs)
- A VU is an application container that manages a subset of the VUA
- Volumes are constructed as a stripe across all VUs
- VUA to VU mapping via modulo function: (LBA / 64KB) % NUMBER_OF_VUs
- VUs are our first level of abstraction away from the hardware, and are the basis of our node resiliency mechanism

Hardware
(Dell R730s)
- triple-redundant power (N+2 redundant on everything) and data path
- 3 active-active-active nodes (linux servers clustered together with InfiniBand point-to-point connections)
- 24 x 8Gbps FC ports
- 6/12 x 10 Gbps Ethernet ports
- Up to 3.2TB RAM
- Up to 48TB SSD
- Up to 480 disk drives
- support 3, 4 and 6TB NL-SAS drives

disk enclosures connected to DC power, not internal system UPS

[power distribution slide]

[SAS and IB Interconnect slide]

Disk enclosure (2 PSUs, 2 I/O modules - SAS expander, 60 top-loading drives)

Every node can see every disk

Virtual Disk Address (VDA) Space - the raw storage capacity - storage for parity (2 sections of parity per 14 data sections)
5% in each disk for spare and self-healing

You can lose any 12 drives in the system

Mapping between VUA and VDA addresses is stored in a “trie”
- compact
- fast
- cache friendly

infinitely variable block/section size between the VUA and VDA mapping

- range of VDAs is much smaller than the range of VUAs
- VUA are unallocated to VDA until actually written (thin provisioning)
- more than one VUA may be mapped to a VDA

support FC, iSCSI, NFS currently supported in Production via Port drivers. FICON and REST are coming

for each 64K section, additional 4K used for metadata

The concatenation of all the stripes is the VDS

RAID groups are a bucket that stores stripes

Randomness

Destage
- destage stripes are not just randomly created as they come
- data sections are classified by “activity patterns”: sequential, cold, hot, newly written, etc
- “multi-modal” destage stripes are built of sections with similar activity patterns
- consecutive stripes, taken with their parity blocks, form the RAID groups

Cache Manager
- DRAM and NAND flash are a transparent cache in front of the disk store
- 4KB cache granularity ~= 12B slots for storing the working set
- cache is unaware of high-level structures like LUNs, file systems, etc (only aware of sections and their associated metadata)
- I/Os are classified by sequence detector into Activity Vectors, which are sued to predict future I/Os
- Predictions are transmitted to sequence handler which pre-fetches sections with elevated probability

Snapshots
“Snapshots are not interesting / commoditized / table stakes / an RFP checkbox”
There’s room for improvement with both CoW and RoW
Most modern file systems and storage systems have strange snapshot performance behaviour at scale - and lock management is the culprit (hard on scale-up, very hard on scale-out)

Re-imaging snapshots
- snapshots are metadata updates and zero-time operations
- no locking of metadata nor pausing IO
- every written section includes a timestamp in its metadata
- every write is unambiguously in or out of a snap based on the timestamp
- 100K snaps / system with no performance impact
- a volume with zero snaps and many snaps has indistinguishable performance
- a snap group with 25 incoming snaps /sec has razor flat latency

Replication
- built on Infinisnap
- Ethernet transport
- Replicate one volume to the entire system
- < 4 second RPO
- Zero performance impact

File Systems
- provided to al customers as a free SW upgrade
- directly integrated with the backend (no additional translation layer, common features are shared between NAS and SAN)
- 3-way active cluster with load balancing
- designed for scale (high file count, deep / shallow file systems, 1000s of file systems scaling with each release)
- balancing performance and cost using a hybrid storage model

[Unified Architecture Slide]

InfiniNAS - architecture peak

- based on a scalable Btree data structure
- all nodes are always 1 indirection away (regardless of depth / width of the file system, guarantees persistent performance over time)
- Btree is tightly integrated with core data structures (small memory footprint)
- Dedicated performance optimisation for SAN / NAS (InfiniNAS takes more educated prefetching decisions, increases cache hit / lower latency)