Silk All-Flash Array Architecture

September 2021

TABLE OF CONTENTS

Executive Summary

As more organizations move their data to the cloud, Silk is invested in ensuring that its Data Platform makes it easy to migrate and maintain data on both the public and private cloud. In this paper, we will take a look at how Silk offers companies who choose to keep applications in the private cloud traditional SAN to achieve new levels of performance at a better cost by delivering high speed and low latency all-flash technology to host devices over a network.

Silk's Platform is a combination of tested and packaged software and services. The software stack is built on the operating environment components of VisionOS which offers a rich set of data services to reduce, protect, manage, and connect your data.

- **DataShrink** – Data reduction features and capabilities are mandatory for economics of flash storage. Silk supports selective inline global deduplication, inline byte aligned compression, thin provisioning, pattern removal and zero detection. These data efficiency functions have established Silk as the cost-efficiency leader of flash storage.
- **DataProtect** – Silk values its customers' data more than anything. Easy to use instantaneous space optimized (zero footprint) snapshots and asynchronous replication allow for returning to any point in time in any site. Data-at-rest AES 256-bit encryption ensures that data is kept private and safe. A highly available and scalable architecture with of no single points of failure, that supports non-disruptive upgrades (NDU) and a robust data protection heuristic enables 99.999% of platform data availability.
- **DataManage** – Silk can be managed by various interfaces. Administration of individual arrays is done via an intuitive HTML5 based GUI, a scriptable CLI and a full function RESTful API. Administration of groups of arrays globally is done using the Silk Clarity cloud-based tool using HTML5, and assets are ordered based on client defined organizational schema—by geo, business unit, datacenter, etc.
- **DataConnect** – Silk's RESTful API allows for external ITSM applications for easy integration and seamlessly management of the platform. This eco-system is constantly growing and includes VMware vSphere, Microsoft VSS, OpenStack, Docker (containers) and Cisco UCS director. Full Microsoft PowerShell support and an SDK, a Terraform provider, Ansible playbook examples and Kubernetes cluster support via CSI compliant plugin are also included.

This white paper describes the Silk architecture and VisionOS operating environment, detailing its core features and functionalities and how they are deployed in the data platform stack.

Introduction

Flash as data persistence media has been a game changer. It finally allows data services and IO performance to match (and exceed) the evolution, progress, and performance of x86 CPUs and network technology. However, using flash for data services requires creating a new architecture, since legacy architectures are tightly designed to match the characteristics of spinning disks and caching engines, which do not work well with all flash data persistence and performance mechanics.

VisionOS was designed from the ground up to utilize flash in the most efficient manner possible, maximizing performance, capacity, and data services while providing for a fully scalable and expandable architecture that allows for granular addition of performance or capacity on demand and non-disruptively.

Silk's scalable architecture and operating environment are the primary pillars to drive all-flash for critical line of business workflows and the most important customer experience applications, providing the following tangible business benefits:

- **Cost efficiency** – Rich data services such as global inline selective deduplication, inline compression, thin-provisioning, zero footprint snapshots and replication, and Silk's ultra-efficient proprietary heuristic for triple data protection provide excellent capacity and performance density at extremely low cost. The Silk solution also offers a highly efficient power and cooling footprint to keep operating expenses low and a rapid ROI. Additional efficiencies are gained through the zero-touch scalability, consistent performance, and automated self-configuration and self-tuning the platform has built in. Very little administration and no classroom training are required to operate and integrate the platform into existing datacenters and workflows.
- **Scalability** – The Silk platform's unique scale-up and scale-out architecture covers both dimensions of scale – capacity and performance. Silk can linearly grow the number of CPU cores by adding c. nodes or independently grow capacity by adding media nodes – removing the limits of rigid architectures that are unable to scale-out and cannot benefit from true shared metadata pool. Silk's Gen6 can scale to 4PB of shared capacity with data services that span across the entire namespace and offers HTML5, SSH, or RESTful management through a single pane of glass.
- **Performance** – The Silk Platform accelerates production transactional, BI, OLAP, and DevOps environments with consistently high performance that serves any kind of mixed workload through our famous proprietary global variable block size algorithm. This feature delivers consistent <1ms latency with millions of IOPS and over 20GB/s of throughput for any mix of workload—small, large, random, or sequential. It delivers consistently better user experiences in transactional environments, removes the killer IO Blender effect from virtual servers, and significantly reduces workflow runtimes for analytics environments and DevOps iterations. Silk's platform has been consistently recognized by Gartner as being a top performance player every year since the all-flash report's inception in 2015.

System Overview

General

VisionOS is the secret sauce that binds best-of-breed enterprise hardware components to create the high-performance data services and persistence engine. Vision OS is software-defined, meaning it does not use any proprietary hardware technology.

Silk Data Pod

The Silk Data Pod (SDP) is the building block of the Silk Platform and includes the following hardware components for on-premise deployment:

- Two c.nodes, each a 1U controller with 512GB of RAM and 40 CPU cores.
- Each c.node provides full redundancy for its components:
  - Two or Four 16Gbps or Two 32Gbps FC ports and two 10 or 25GbE Ethernet ports for iSCSI host connectivity
  - Two 40Gbps InfiniBand ports for cnode East/West backend interconnectivity
  - Four 12Gbps SAS channels for connectivity with the media shelves
- Base media shelf, 2U in size.
  - 24 hot-swap SSDs; datacenter grade with 3D TLC flash
- Expansion media shelves, 2U in size.
  - Same as the base media shelf

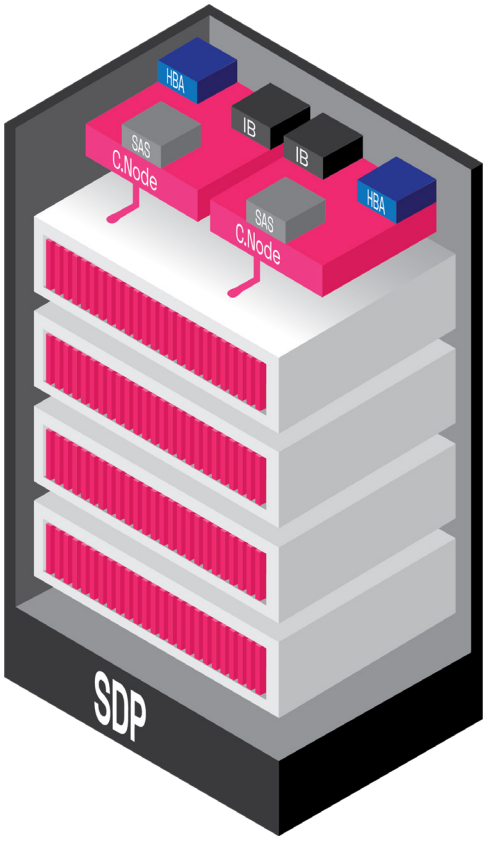
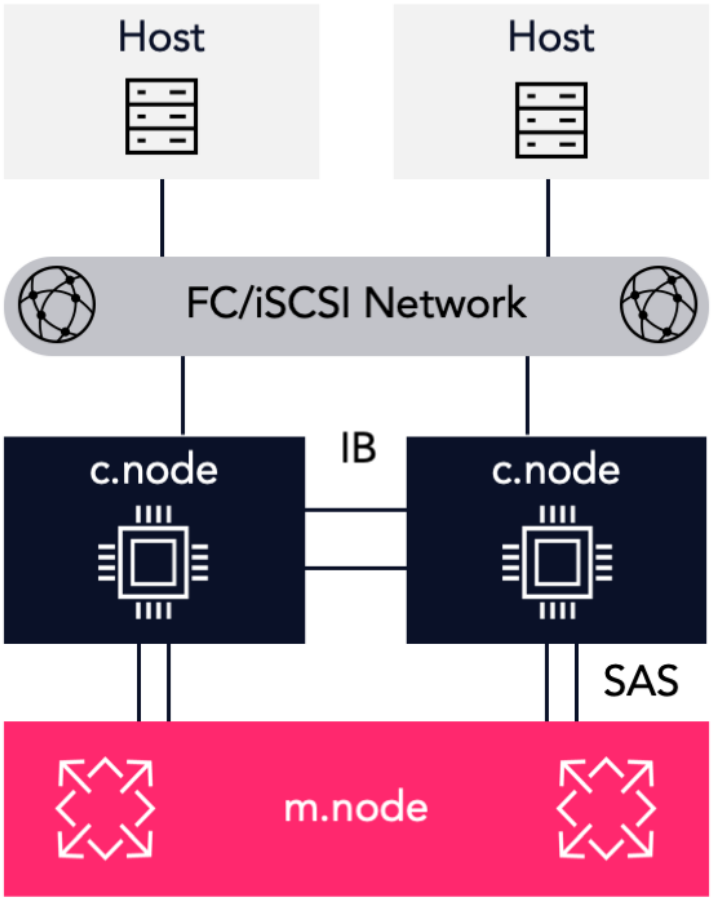


Figure 1: Basic SDP Components

SCALABLE ARCHITECTURE

The Silk Platform is designed to facilitate linear growth in both capacity and performance while maintaining consistent low latency. This is achieved through implementing a true symmetric, active-active, scale-out architecture where all controller elements of the platform are working together simultaneously. In addition, Silk also provides for simple capacity expansion with zero impact on performance for true scale-up architecture. The combination of both dynamic non-disruptive scale-out and simple scale-up architectures in a single platform is the key element for building a data infrastructure that can scale in the most cost-effective way. All while ensuring efficient granular deployment of hardware to meet application requirements for capacity and performance, and significantly reducing wasteful and costly stranded capacity or performance. Any increase in performance or capacity is handled through an automated rebalancing of data and IO access, with no human intervention or management required.

The starting point for any Silk configuration is a single data pod, made up of two c.nodes and an m.node:



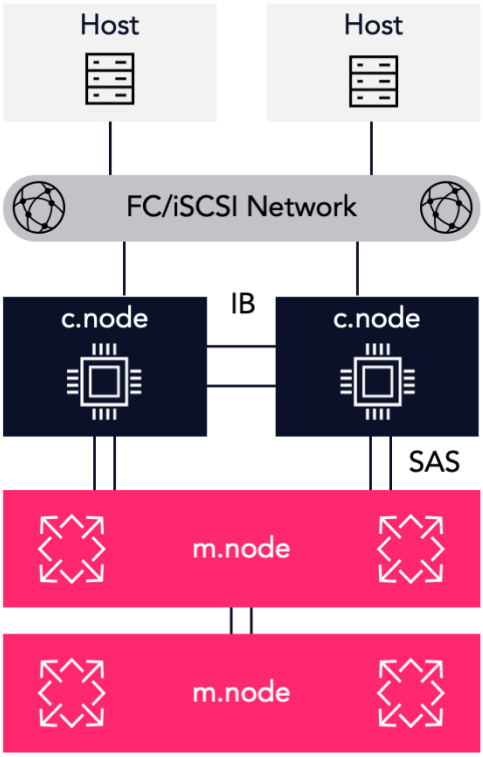
- The c.nodes are connected redundantly using point-to-point IB connectivity.
- Each c.node is connected redundantly to the media shelf using SAS connectivity.
- The c.nodes have Active/Active connectivity either directly to the host(s) or through a switch, via FC or iSCSI.
- Host data and metadata are automatically evenly distributed between all media in the data pod and are accessible from every c.node cluster.

From this basic building block the platform can be expanded according to application needs using a scale-up and/or a scale-out approach.

SCALE-UP

Scaling up means adding more capacity by adding expansion media shelves without adding c.nodes.

Scaling up a single data pod with an expansion shelf is shown below:



- The expansion increases the capacity density and reduces the cost per GB.
- The shelf is connected redundantly using SAS connectivity.
- The expansion is done online with no downtime.
- The new configuration has the same performance as before.
- Existing volumes are automatically redistributed between all the media in the pod.
- There is no change required to any of the host connections or definitions.
- Scale up to a total of four (4) shelves or 240 physical usable TB (1PB logical) capacity in a single pod.
- Mix and match different shelf capacity sizes as needed.
- The expansion process can be seen in the GUI inventory screen, as seen below:

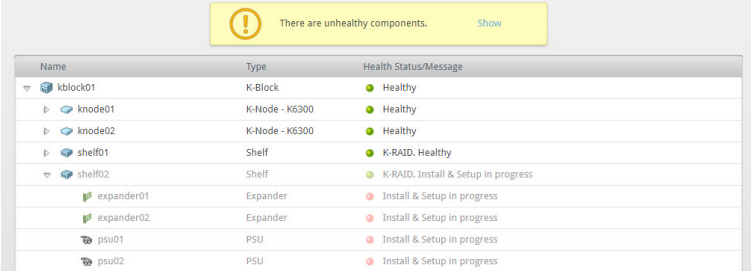


Figure 4: Scale-up expansion in-progress

SCALE-OUT

Scaling out means increasing the number of cnodes, thus adding more performance and capacity headroom.

Scaling out from a single data pod to a dual data pod is shown in Figure 5:

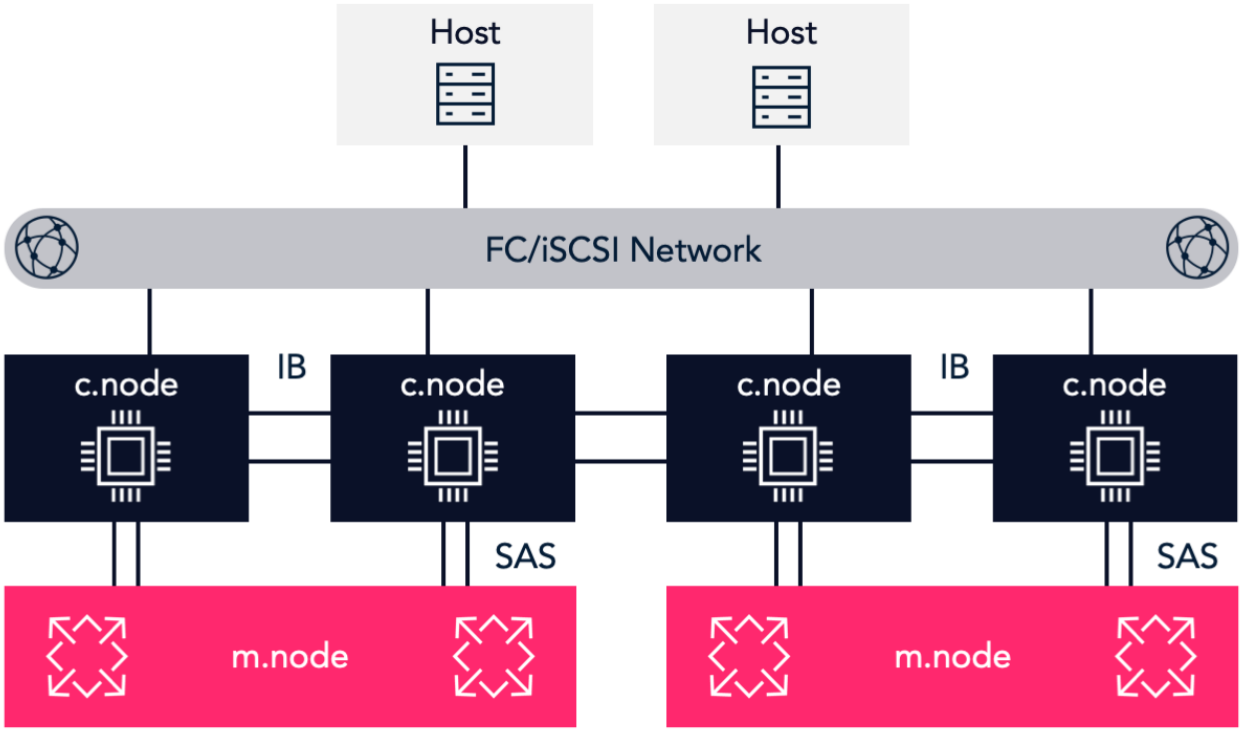
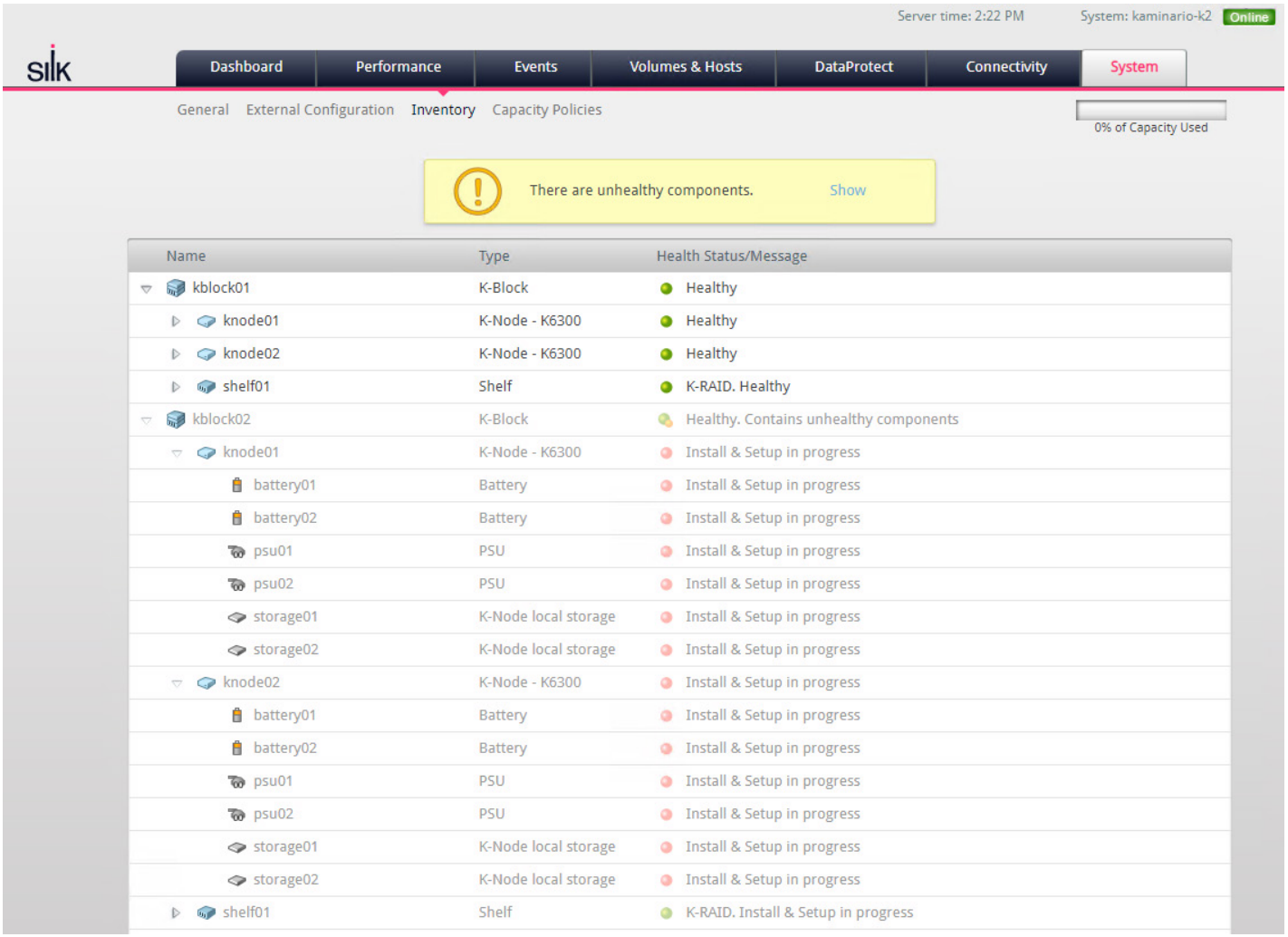


Figure 5: Scaling Out from a Single Data Pod to a Dual Data Pod

- The expansion linearly increases the maximum capacity, IOPS, and throughput capability of the data pod. Latency is always kept consistently low (<1ms) and is generally not affected by expansions.
- Two InfiniBand switches are required to support the interconnect between all the SDPs in the array. Scaling beyond two SDPs will not require any additional networking hardware.
- The expansion is performed online with no downtime or decrease in performance.
- Existing volumes are automatically redistributed between all the media in the cluster and can be accessed from every
- c.node in the data pod.
- Metadata is also redistributed between all c.nodes, enabling unified global data services across all c.nodes and media.
- New hosts can be connected to the c.node front end IO ports, and the new and existing hosts can access all volumes via standard mappings.
- The expansion process can be seen in the GUI inventory screen, as seen on the next page.



SCALE-OUT AND SCALE-UP

There is no particular order in which the SDP has to scale. Silk can first scale up, then scale out and vice versa. This flexibility removes any compromises of using a data platform that is not adaptable to the customer's needs and application requirements.

Like a single data pod scale up operation:

- The expansion increases the capacity and performance density and reduces the $/GB cost.
- The expansion shelves are connected redundantly using SAS connectivity.
- The expansion is performed online with no downtime or decrease in performance.
- Existing volumes are automatically redistributed between all the media in the array.

The system can continue to scale out with c.nodes and additional expansion shelves and/or scale up. Continuing the example above, scaling from a dual data pod to a quad data pod and increasing the number of media shelves to four (4) shelves per data pod.

SUMMARY TABLE

The ability to accommodate various capacity sizes of SSDs and the flexibility of scalability allows a tailored-fit configuration to meet the requirements of the individual customer. The table below captures the most common configurations and capacities.

Silk Gen6 Data Platform				
Scaled Configuration	1 data pod	2 data pods	3 data pods	4 data pods
Physical Capacity*	30TB-240TB	60TB-480TB	90TB-720TB	120TB-960TB
IOPS	Up to 425,000	Up to 850,000	Up to 1.2M	Up to 1.7M
Throughput	Up to 6GB/s	Up to 12GB/s	Up to 18GB/s	Up to 25GB/s
Latency	From 160 microseconds to 1 ms			

Physical Capacity is subject to drive size and represents a 1:1 data reduction ratio. Logical Capacity is 1PB per data pod and depends on data reduction ratios.

DataShrink

The ability to build a cost-effective data platform relies almost entirely on the efficiency of the architecture in delivering usable physical capacity from RAW capacity, and then determining how much effective logical capacity can be generated from that physical capacity through the use of data reduction services.

VisionOS is built on the central concept of being highly efficient without compromising on other features such as enterprise resiliency and consistent performance. These data reduction and efficiency features play a central role in IO processing.

DEDUPLICATION

VisionOS's global inline selective deduplication meets the demanding requirements of eliminating redundant data and stores unique data only once. Deduplication is performed globally across all of the media shelves and the hash comparison processing is distributed across all the c.nodes in the pod, which enables generally higher deduplication ratios because of the large comparison pool, while maintaining high IO and throughput performance with consistently low latency (1ms or better) gained from the distributed processing architecture. As the data pod scales out, so does the deduplication pool and hash processing power. VisionOS offers the option of selective inline deduplication. It allows storing data without deduplication for applications whose data redundancy is negligible and additional performance is preferred (such as OLTP database applications using SQL), as well as for security-sensitive applications where deduplication is prohibited. Any volume can be moved from a deduplication group to a non-deduplication group and vice versa at any time.

COMPRESSION

VisionOS uses inline real-time compression that is optimized for low latency performance. Compression is done after deduplication, which makes compression the standard always-on method of data reduction for non-dedupable data sets that are common in database environments such as Oracle and SQL Server.

VisionOS software uses lightweight and quick LZ4 based compression algorithms with the ability to compress data to byte-aligned granularity, which achieves high compression ratios, but completely avoids any padding or fragmentation. The compression is performed on 4KB sequential segments rather than on larger segments, thus ensuring that small block reads do not result in retrieval and rehydration of unnecessary data. Because the entire block is tracked in metadata, any size read request across the block is done as a single IO operation. This means that a 128K block can be retrieved in whole or part from 4K up to 128K in a single IO operation. This dramatically speeds up read access latency and also greatly reduces back end CPU processor load. The output of a compressed 4KB block is rounded to the nearest byte. This byte-aligned compression prevents internal fragmentation and facilitates better compression ratios. Blocks smaller than 4K are compressed and then combined together to fill out the entire 4K segment. These segments are combined together to fill out an entire "stripe" of logical data that is persisted down to flash all at the same time. Many GB of full stripes of data are committed at the same time down to the flash layer, dramatically reducing write operation amplification and eliminating any possible fragmentation during the write process.

THIN PROVISIONING

Thin provisioning enables maximizing efficient utilization of the physical storage capacity and gives admins the ability to plan capacity provisioning for the long term. All volumes in the Silk Data Pod are always thin provisioned, with a fine grained on-demand growth of 4KB. Unmap and Trim operations are also supported in the same granularity and are done as background operations over time. VisionOS delivers the required management tools that bring the thin provisioning feature to the next level, where the capacity management of volume provisioning is easy and hassle-free. Volumes can easily be expanded on demand to the hosts, but capacity is never physically allocated until it is actually used—as unique compressed bits.

DataProtect

Silk is architected and built to meet the requirements of the most latency and availability-sensitive enterprise applications. Enterprise resiliency starts by deploying only enterprise grade hardware components and continues with High Availability (HA) N+1 scalable fault domains throughout the Silk Data Pod. In addition, data can be protected from errors at the application level using native features such as snapshots and replication.

SNAPSHOTS

Silk's patented crash-consistent snapshot architecture follows VisionOS's guidelines for storage efficiency, performance, and scalability. Snapshots are created instantly, with no performance impact and are zero footprint at creation. Snapshots track only the deltas from the volume in a 4KB granularity, using a redirect-on-write (RoW) approach. This storage-efficient design also keeps the impact on flash endurance to a minimum by keeping writes to a minimum. Snapshots can be mounted for read/write purposes for additional working environments such as QA, Test/Dev, analytics, backup, or other DevOps functions, all at a very low cost of storage capacity. Read/write snapshots deliver the same performance as the production volumes, without any performance impact on the production volumes.

Using the snapshots restore functionality for recovery purposes is done without losing any of the snapshot history and can be done at any time. The snapshots can be accessed from any of the controllers of the Silk Data Pod and restores/mounts are instantaneous. Snapshots can also be scheduled with the built-in scheduler. Application consistent snapshots can be scheduled using the Silk PowerShell toolkit or the native REST API calls to quiesce volume groups before snapshot and enables them after.

REPLICATION

Silk replication provides Data Pod resiliency to support enterprise datacenter BC/DR resiliency requirements and data mobility/migration services. Silk leverages its snapshot architecture to facilitate asynchronous snapshot-based replication between Silk Data Pods. Since replication is based on Silk's pointer-based snapshot architecture, there is no impact on production performance, and only deltas of the replicated copies are captured with no dependencies on the link speed.

VisionOS storage efficiency features change block tracking and compression are also used to dramatically reduce the amount of data sent between arrays. Failover is push-button simple. Failback works the same, and only the deltas are sent back with no need to fully rebuild the original source Data Pod.

All essential disaster recovery capabilities such as short Recovery Point Objective (RPO) and Recovery Time Objective (RTO) are gained natively without any third-party software components. RPO can be set as low as 1 minute, with intervals of 5 minutes, with up to 24 hours available between updates.

ENCRYPTION

Silk provides SED based 256-bit AES encryption features to ensure that data is not accessible except through the normal IO path of the Data Pod. Any retrieval of data from the media after removal from the shelf will be encrypted and useless. Once encryption is enabled, all the data in the Data Pod is protected. If an SSD needs to be replaced due to failure, the keys can be easily changed after replacement, resulting in full cryptographic erasure of the failed SSD. Support options are available to keep or destroy on-site replaced drives as needed. Keys are managed by the Silk engine internally; no key manager is required. The keys are also spread out across the entire capacity space of the Data Pod and cannot be compromised by theft of individual drives. Multiple key copies are maintained in this fashion to ensure that any flash data corruption cannot impact the integrity of the decryption functions.

K-RAID™

Silk developed the K-RAID - a proprietary data protection schema that delivers the equivalent of triple parity protection but is also highly efficient with an 87.5% utilization rate from raw—a parity overhead of only 12.5% per shelf (3 out of 24 SSDs). This high rate is gained without compromising on either the protection level or the performance of the shelf. This rate is achieved by deploying an efficient erasure coding heuristic on each flash shelf. This erasure coding algorithm consists of two logical RAID groups, each one with a parity (P1 and P2), and an additional logical parity device for the two groups (Q), as show in Figure 9, below:

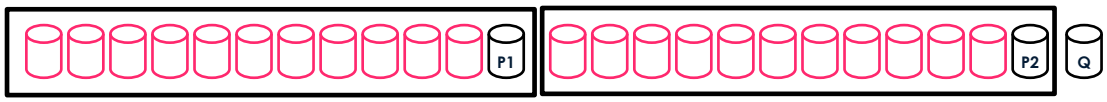


Figure 9: K-RAID logical layout of two RAID groups and a Q parity for both groups

This is a logical representation--there are no dedicated parity drives – all data and parity roles rotate between all the drives in a cyclic manner as data stripes are persisted to flash.

Aside from being highly efficient, K-RAID is extremely robust. It can sustain two concurrent SSD failures and up to three SSD failures within each separate shelf without loss of any data. As the Data Pod scales up in capacity, the number of sustainable concurrent system-wide SSD failures also increases, as the fault domain boundary for SSDs is the individual shelf. This serves to significantly limit the blast radius of possible failures to a single shelf as opposed to the entire capacity pool. K-RAID employs a triple parity protection schema that adapts according to the failure at hand. Single media failure is quickly recovered thanks to efficient metadata and real-time system health monitoring. There is minimal performance impact during the rebuild and no performance impact once the rebuild is completed.

The K-RAID layout is pseudo-randomly distributed throughout each SSD shelf, meaning that there is no fixed Parity or Data disk role per SSD. There is no hot spare, meaning that all the SSDs are utilized for protection and performance at all times.

K-RAID is fully automated and does not require any human configuration, tuning, or tweaking, which means another traditionally tedious IT task is avoided with Silk.

K-RAID FLOW

1. The flow of how K-RAID works in the scenario of media failure is shown logically below:
2. **D5 = XOR (P1, D1…D11)**

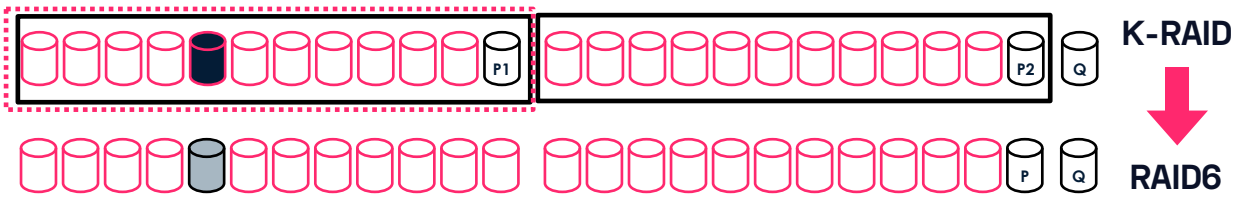


Figure 11: K-RAID flow during media failure

- Media failure is shown in the left logical RAID group. However, since the K-RAID layout changes within the SSD Shelf every 64KB, all the different K-RAID layouts experience a different failure, i.e., D5 is not necessarily affected in each layout. To be precise, it will only affect D5 in 1/24 of all permutations of the K-RAID layouts

- Due to the different K-RAID layouts, the rebuild process of the failed media is performed from all the remaining 23 healthy drives, which results in considerably faster rebuild times.
- When reading from a RAID group that has lost or corrupted a data segment, the XOR calculations are performed only from that affected RAID group, which is built roughly from half the drives in the shelf, delivering superior read performance during rebuild. In that RAID group, the Parity segment will take over the role of the lost Data segment and the Parity of the unaffected RAID group (P2 in the figure) will become P, a product of XOR (P1, P2).
- When reading from a RAID group that lost a Parity segment (P1, P2, or Q), there is no performance hit at all.
- Once the rebuild is completed, the K-RAID falls back to a dual-parity RAID6 scheme, meaning it is still capable of handling two more concurrent SSD failures without losing data.
- When the failed media is replaced and restored, the K-RAID will be restored to its original triple parity layout.

3. The flow of how K-RAID works in the scenario of a second media failure is shown below:

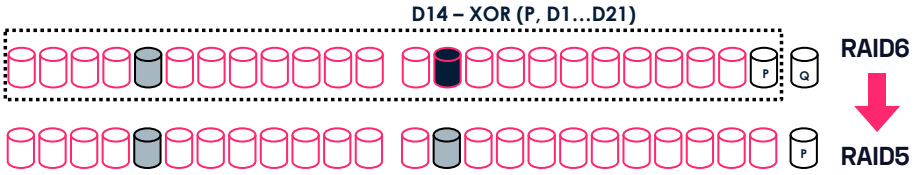


Figure 12: K-RAID flow during second media failure

- This is a standard RAID6 type recovery.
- When reading from a K-RAID layout that lost a Parity segment (P or Q), there is no performance hit at all.
- The Parity segment (P) will take over the role of the lost data segment, and the second parity Q will become the new (P).
- Once the rebuild is completed, the K-RAID falls back to a single parity RAID5 scheme, meaning it is still capable of handling another media failure without losing any data.
- When the failed media are replaced and restored, the K-RAID will be restored to its original triple parity layout.

2. The Q parity – handling concurrent failures. When two drives fail concurrently, the Q parity comes into play for those K-RAID layouts that lost two Data segments or a single Data segment and its Parity, as shown below:

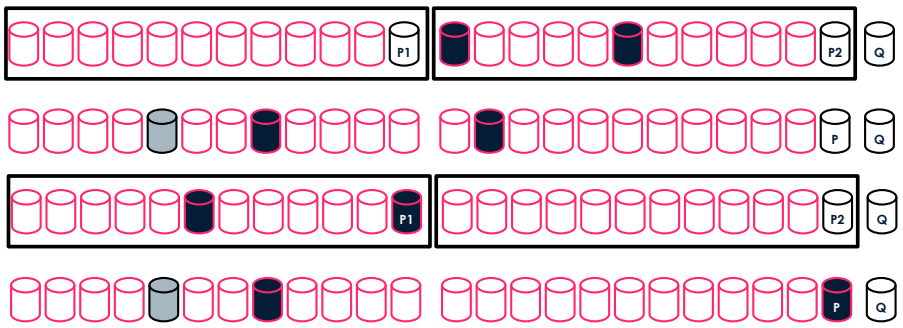


Figure 13: Using the Q Parity for two concurrent failures

- VisionOS uses a proprietary highly efficient algorithm for rebuilding the data out of the healthy data segments, P and Q. In some of the K-RAID layouts, two concurrent failures will be treated as two single, unrelated failures. In other K-RAID layouts, one or two of the lost segments can be P1, P2 or Q.

NO SINGLE POINT OF FAILURE

VisionOS supports a double-everything approach for Silk hardware components and all of the data and metadata at rest are protected by K-RAID™. However, Silk employs a symmetric active-active architecture that does not have any passive or idle components; all of its resources are being utilized at all times. There is full redundancy of every component in the system and there is not any one single component that can fail and cause unplanned downtime or data loss. Each Silk Data Pod element is a standalone failure domain, which means that the entire Data Pod, in the case of extremely unfortunate events, can sustain more concurrent failures as the Data Pod scales out without losing data and can still maintain data availability.

POWER LOSS

Silk possesses enterprise capabilities to sustain a full power outage in the facility and still keep the data cleanly intact and available when power returns. Any metadata and data that were already acknowledged by the c.nodes before being persisted to the media are protected in two separate c.nodes for redundancy. Silk c.nodes immediately recognize across the management plane when a full power loss event has occurred and has impacted the entire Data Pod (i.e. every c.node in the Data Pod loses power to both power supplies). Each c.node in the Data Pod is equipped with BBUs (battery backup) that will allow enough time for the c.nodes to fully de-stage any inflight data that has not been persisted to the media. Data that is already stored to the K-RAID is kept persistent and is sustainable through power cycles. Full state information is de-staged to redundant, mirrored local SSDs on each controller to allow for clean recovery when power returns. The SDP will initiate automated shutdown procedures as soon as the memory and state are flushed.

NON-DISRUPTIVE UPGRADES (NDU)

Silk can upgrade any of its hardware and software components with no impact on data availability or performance. Silk's architecture is software-defined, so new features/enhancements/bug fixes/firmware updates to VisionOS can be deployed with no dependencies on maintenance windows or scheduling around running workloads. Hardware can be replaced/upgraded/added in similar fashion. With NDU combined with a scalable architecture, Silk provides the best TCO of storage: No monolithic fork-lift upgrades to add capacity or performance through granular scale out or scale up, and no need to plan down-time with NDU. All of these operations are performed non-disruptively and can be done remotely or with simple helping hands for hardware adds and moves.

DataManage

VisionOS provides a rich set of tools to monitor and manage the Silk cluster. These management services do not require additional hardware since they are built into Silk's software and are accessed using redundant secured 1GbE connections via HTML, SSH, or RESTful commands. With DataManage, there is no need to manually configure RAID groups, tune the array to a specific application, or create affinities between volumes and controllers.

UI

The Silk UI is simple and clean, keeping all the necessary information in a tiled dashboard. Each tile presents the essential details of a Silk management component: System Health, Volume information, Replication, Performance Statistics, Capacity and Events.

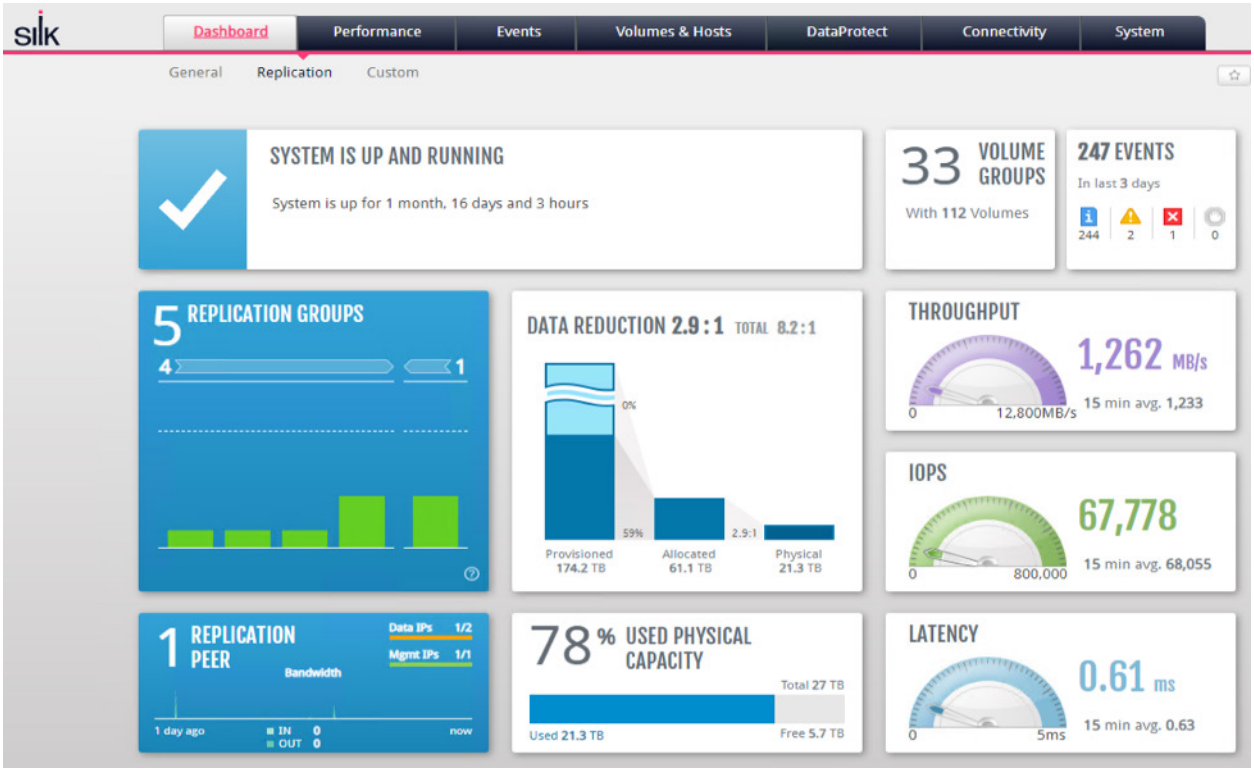


Figure 14: Silk GUI Dashboard

Clicking a tile will automatically open a more detailed and actionable view of that management component: Create volumes or snapshots, set replication policies, create capacity policies, assign hosts, manage connections, and so on. The tiles are dynamic and can be rearranged according to personal taste, as can be seen in Figure 15:

The GUI is accessible via any web browser and does not require any software installation or separate licensing

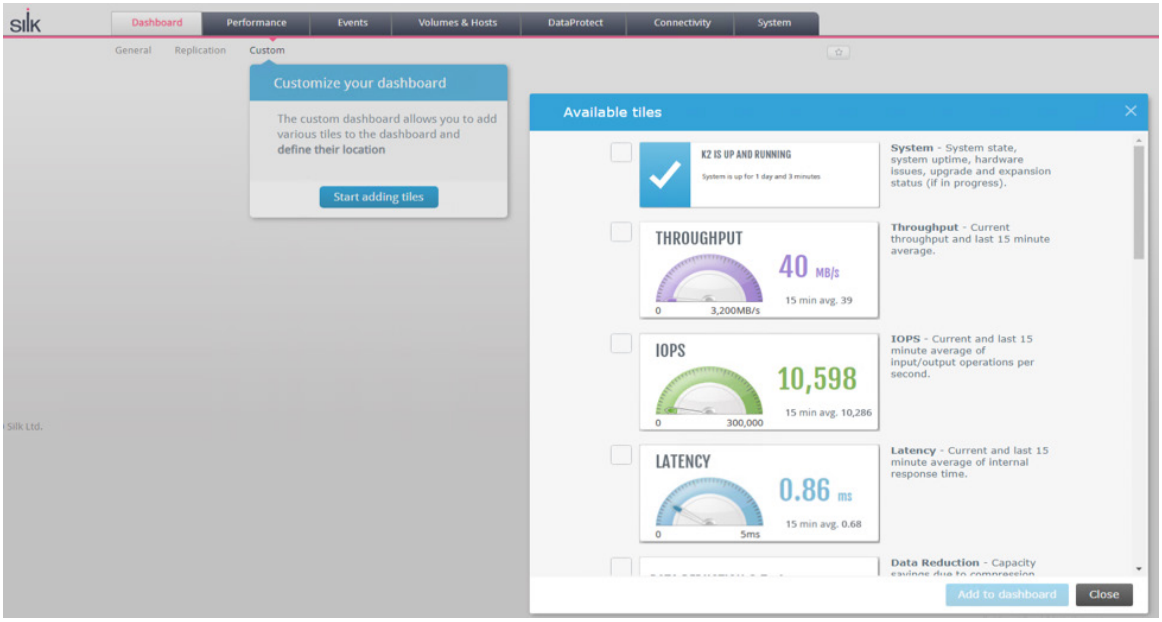


Figure 15: Silk GUI Dynamic Tiles

CLI

The CLI is fully scriptable and allows full control of all system features and functionality. The CLI is accessible via an SSH connection using a Linux/Unix shell or Putty-like utility.

RESTful API

Silk exposes a full set of RESTful APIs that allows it to be controlled, managed, and monitored via automatic third-party software and in-house management platforms. The Silk REST API is publicly available on GitHub with examples on how to develop automatic flows. It can be deployed into any framework with JSON, Python, or IAC/Configuration management tool (like Puppet or Chef).

PowerShell SDK

Silk has built a full SDK for PowerShell with cmdlets that support all system functionality. The SDK includes full documentation and examples of how to integrate Silk functions into PowerShell based workflows. The PowerShell SDK is available on Silk's GitHub site.

Terraform Provider

Silk has written a Terraform provider that supports the main user functions of the system for common workflow integration. The provider can easily be extended as needed to support any functionality needed. The Terraform provider is available on Silk's GitHub site.

Kubernetes

Silk has written a CSI-compliant Kubernetes plugin that supports the full range of common operations of CNCF-certified Kubernetes clusters. The plugin is available on Silk's GitHub site.

DataConnect

Using the RESTful API platform, VisionOS allows third party components and independent software vendors (ISV) to monitor and manage the platform. In addition, other protocols such as SNMP, SMI-S, and Syslog can also be used.

VMWARE INTEGRATION

VisionOS incorporates full support for VMware's vStorage API for Array Integration (VAAI), allowing for offloading host storage operations to the Silk c.nodes. By using the Silk Storage Replication Adapter (SRA) in VMware's Site Recovery Manager (SRM), it is possible to benefit from Silk's native array-based replication in a VMware environment. Daily storage administration operations can be carried out via the Silk vSphere plug-in running in VMware's vCenter environment, while VMware's LogInsight can be used to aggregate Syslog events for additional event monitoring capabilities.

OPENSTACK

Leveraging Silk's RESTful API, Silk provides a Cinder driver that allows OpenStack environments to provision storage for private and public clouds. OpenStack is supported from Juno to current release.

CISCO UCS

Cisco UCS director enables the management of Silk as part of the entire stack of networking and compute.

Silk Clarity™

Silk Clarity is a cloud-based analytics platform that includes a comprehensive set of management and monitoring functionalities, including an ML engine with the capability to leverage application-level intelligence and big data analytics into truly intelligent actionable results – all of which enable customers to get more productivity out of their storage environment and deliver higher performance for business-critical applications.

VisionOS continuously monitors all components and is able, through a unique algorithm, to detect in real-time any errors that have a probability of occurring. Any error or change in the system triggers an automatic real-time event that is sent to Silk Support. It is also possible for the storage admin to subscribe to the events of interest by category such as LUN configuration, connectivity, hardware errors, etc. for proactive notification of potential and occurring events.

On a periodical basis, VisionOS collects system-wide information that is sent to Clarity, where it is analyzed to uncover performance, configuration, or capacity management issues and more. This information is reported back to Support, which uses Clarity to provide tailored support for each customer.

Figure 18: Silk Clarity Dashboard

IO Flow

The core function of the VisionOS operating environment is the IO flow of data being written to and retrieved from the Silk Platform. Silk provides Tier 1 high performance data persistence, data services, and data availability. VisionOS is on generation 8 at the time of this writing in 2020.

When taking a closer look at the IO flow, it encapsulates advanced data reduction and protection technologies such as global inline selective deduplication, global adaptive block size, inline compression, distributed metadata, K-RAID, and more – all developed in a true symmetric active-active scale-out architecture without the use of any proprietary hardware. This section details these technologies.

GLOBAL ADAPTIVE BLOCK SIZE

Workloads generated by real applications are variable in their block size. Storage arrays – specifically arrays that deploy deduplication – tend to use a fixed block size (usually 4KB-16KB), with very large boundaries (1KB-2KB) which cause fragmentation of the application's data blocks into small chunks when the original block size is larger than the fixed block boundary of the array. This method can dramatically limit the available throughput for the application and generates multiple backend operations for each IO that is bigger than the pre-defined fixed block size. A single front-end IO can cause dozens of backend operations depending on the size of the block, which significantly degrades performance and can dramatically increase IO latencies. What VisionOS does that is unique is to automatically adapt to the application's incoming block size and handle the IO in its native size without breaking it up or padding it out. This in turn keeps operational CPU overhead consistently low as possible. This patented and unique technology generates the best possible consistent performance for the application's real workload without increasing latency, reducing IOPS, or limiting throughput. The global adaptive block size algorithm allows the Silk Platform to support the real-world performance requirements of a multitude of application types all running concurrently, which is the core essence of a Tier 1 data services platform. This patented algorithm is crucial for deploying a true high performance symmetric active-active scale-out data platform.

METADATA MANAGEMENT

Metadata schema and management is essential and critical in any storage system that claims data services. Its importance grows tenfold when deploying a platform that supports NDU scalability combined with data services such as zero footprint pointer-based snapshots, snapshot-based replication, inline deduplication, inline compression, zero-detect, pattern removal, and perpetual thin-provisioning. The way VisionOS manages its metadata provides for more efficient and faster deduplication and compression and allows for real scale-out active-active access to all volumes and snapshots from every c.node simultaneously. It also facilitates fast recovery in the event of hardware failures and utilizes an optimized and extremely unobtrusive garbage collection process that does not impact performance until the platform is written to an almost completely full status.

Metadata is kept both in DRAM and on flash media, using a unique caching algorithm for consistent performance. This extremely advanced and efficient metadata schema removes restrictions on the capacity size of the media that are used in the platform, and any restrictions on the maximum achievable data reduction ratios. Because metadata is accessed from DRAM, which is the fastest possible medium for access, Silk's metadata performance is an order of magnitude superior to competitors who stage their MD on slower SCM or SSD media. Because of the efficiency of Silk's patented metadata schema, only 1% of allocated capacity is used for metadata persistence, a number unmatched in the industry and incredibly efficient. There is essentially no "metadata tax" with a Silk system, unlike many other systems who have to use up to 25% of their total available capacity just to store the metadata database and have hard caps on efficiency ratios and logical addressable space due to metadata schema limitations. Silk's scalable metadata solution is the most elegant in the industry today.

FLASH ENDURANCE

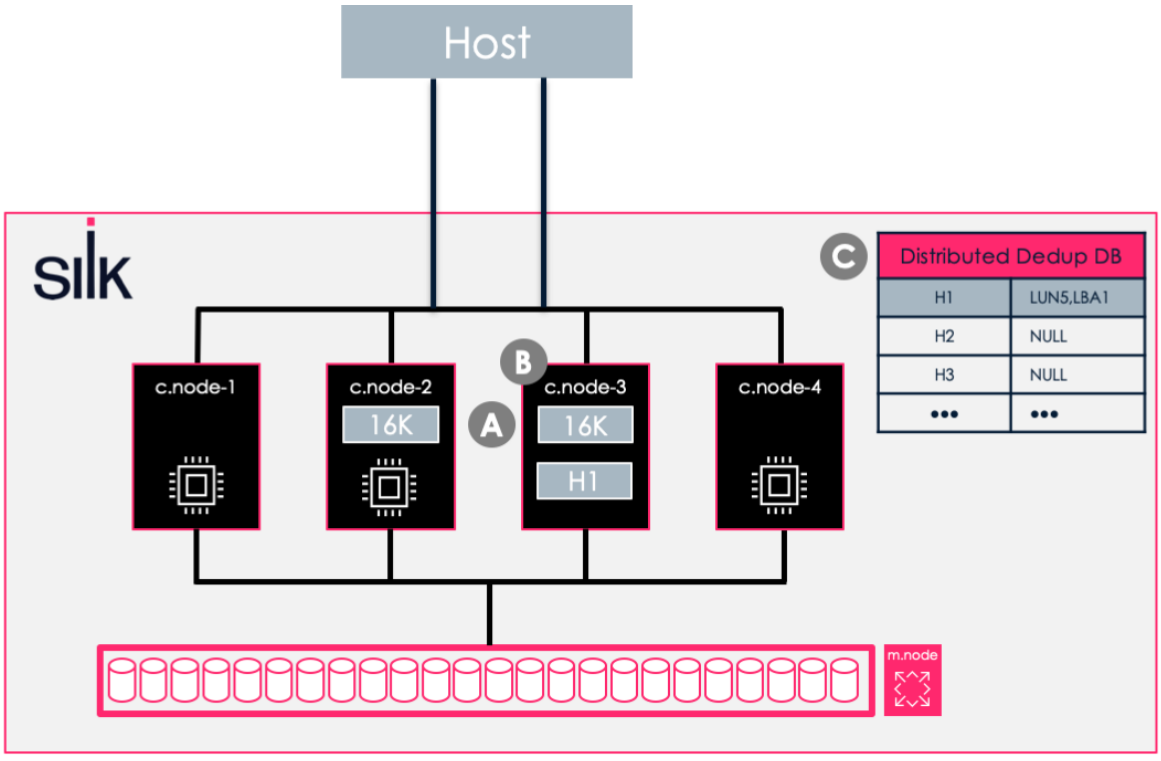
Silk utilizes TLC flash media as of today. VisionOS is designed not only to take full advantage of this media's characteristics but also limits its weaknesses around write endurance. VisionOS's powerful data reduction algorithms reduce the number of actual bytes that are written to the media; and unique data that is written to the K-RAID is persisted using coalesced Log Structured Array (LSA) full stripes so that there are no hot spots or redundant writes when data is persisted. In addition, writes are fully distributed across the entire array and a scalable distributed write cache eliminates hotspots. Stripe maintenance is a background process that continuously maintains high validity of existing stripes and provides immediate consistent availability of free space for new writes to be persisted without wasting media P/E cycles

FLOW OF OPERATIONS

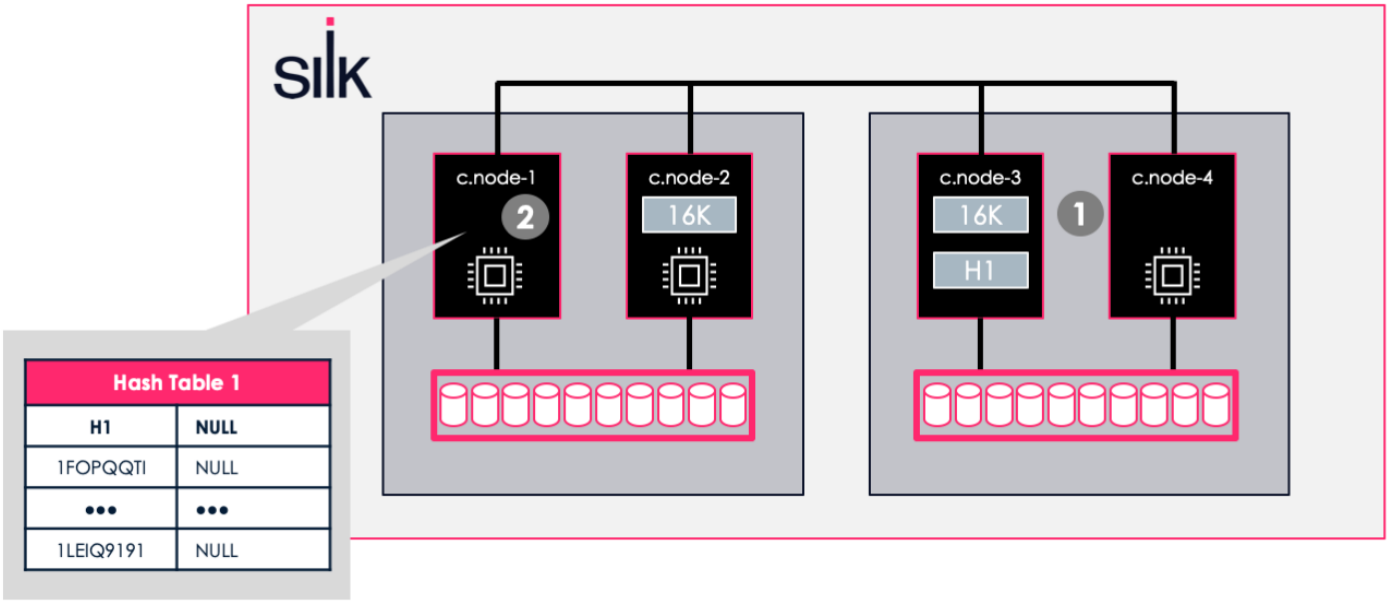
The diagrams below demonstrate a simplified IO flow of writes and reads. In the example flow, a 16KB block is used. It shows how the global adaptive block size algorithm works in a scale-out system with a real application load. This is also true for larger block sizes. When hundreds of thousands of operations are running concurrently through the active-active cluster, additional mechanisms such as batching and queuing take place to provide further optimizations.

Unique Write

1. An application writes a block of 16KB to a logical block address (LBA1) within a block device (LUN5) on the array (Diagram 1). This block can arrive at any of the c.nodes because of MPIO, since the SDP uses a scale-out Active/Active scheme. In this case, the block arrived at c.node2 (A). Once the block is stored in the c.node's DRAM, it must be mirrored to another c.node before returning an ACK to the application to satisfy data-in-flight protection requirements. The second c.node is selected according to a mapping table that maps the (LUN, LBA) of the incoming write to a specific c.node. In this case, cnode3 is selected (B). This mapping table is global and is identical in all the c.nodes (C). The block is then copied over the high-speed low latency InfiniBand fabric to the second c.node, c.node3, which then also stores the block in its DRAM. At this point, the block is stored in two different physical servers which have BBUs – the first c.node can now return an ACK to the application's host. This means that all subsequent data service work performed while storing the new block is asynchronous to host operations. This results in exceptionally low host-side latencies and allows for write latencies as low as 160us on the Silk system.

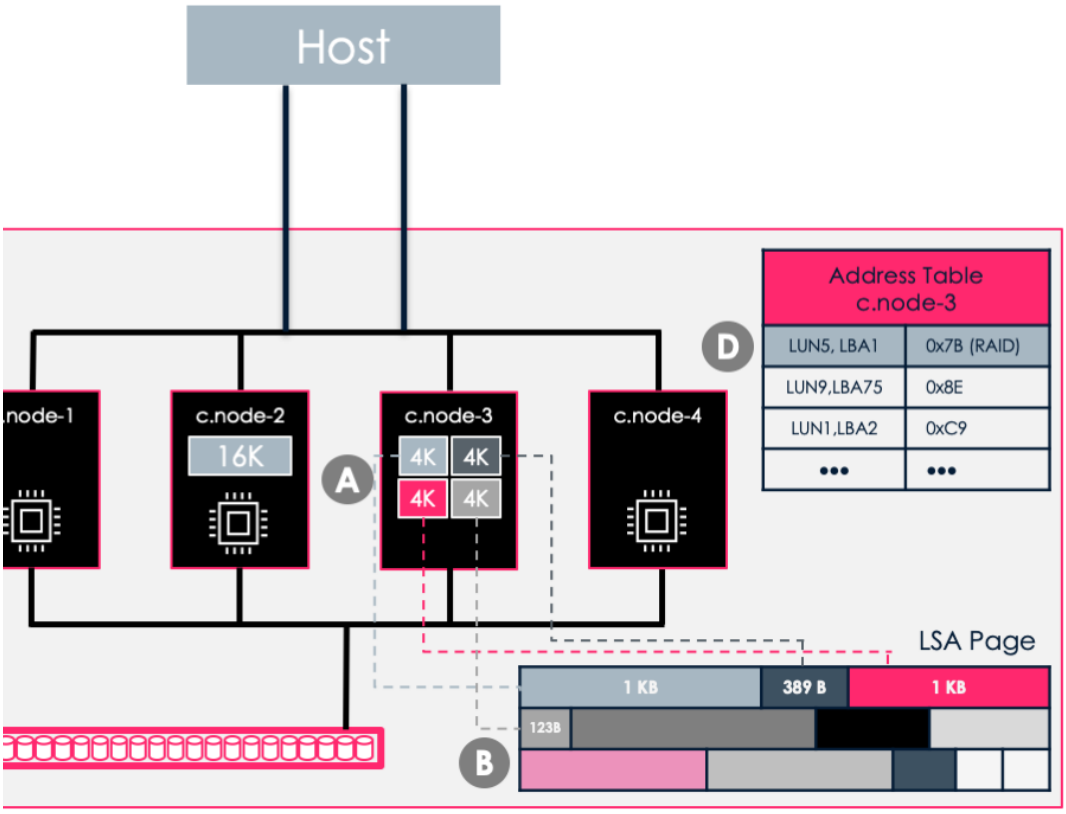


2. Next, on c.node3, the deduplication process starts. Deduplication is variable and adaptive, meaning the array will search for a duplicate weak hash match in the original block size of the 16KB and will not break the 16KB block into smaller fragments when looking for duplicate data, as happens in fixed block size deduplication. A hash value (H1) of the 16KB is created (1). All possible hash values are divided into several hash tables, according to the number of c.nodes in the active cluster. The hash value is sent over the InfiniBand fabric to the appropriate c.node and a lookup is done for hash H1 (2). This will give a first indication of whether that 16KB block already exists in the system. Since this example is a unique write to the cluster, the hash lookup will return negative. The same flow would have been true for a different block size, down to 4KB. For example, if it were an 8KB block or 4KB, a hash value of the 8KB or 4KB would have been created and so forth.



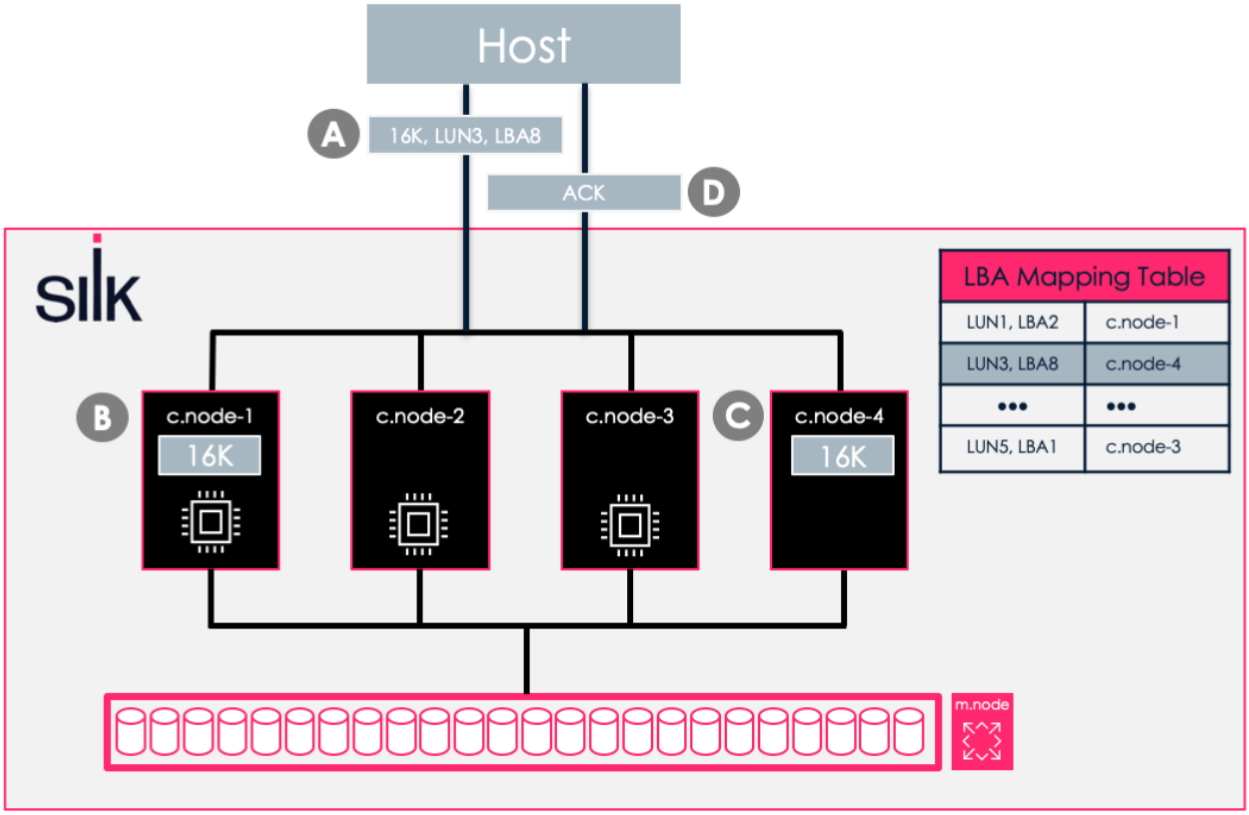
3. C.node3, the c.node that originated the hash lookups, is going to take ownership of the 16KB block. The c.node that was queried before updated its hash table (at the same time of the query that returned NULL) with the owner of the 16KB data: LUN5, LBA1.

4. The 16KB is now compressed: It is divided logically into 4KB segments and each 4KB is compressed separately - to the nearest byte (byte-aligned compression) (A). The compressed 16KB is now placed contiguously in a log-structured array (LSA) page. This page is about 3MB in size (B). Once the LSA page has been filled, c.node3 prepares a full stripe calculating the stripe parity in DRAM and writes it (both data and parity) to the K-RAID (C). C.node3 updates the location of the compressed 16KB in its metadata Address Table, which translates logical (LUN, LBA) pairs to physical addresses on the K-RAID media (D). Once the 16KB block is persisted, both c.node2 and c.node3 drop the 16KB block from their DRAM and the unique write process is complete.

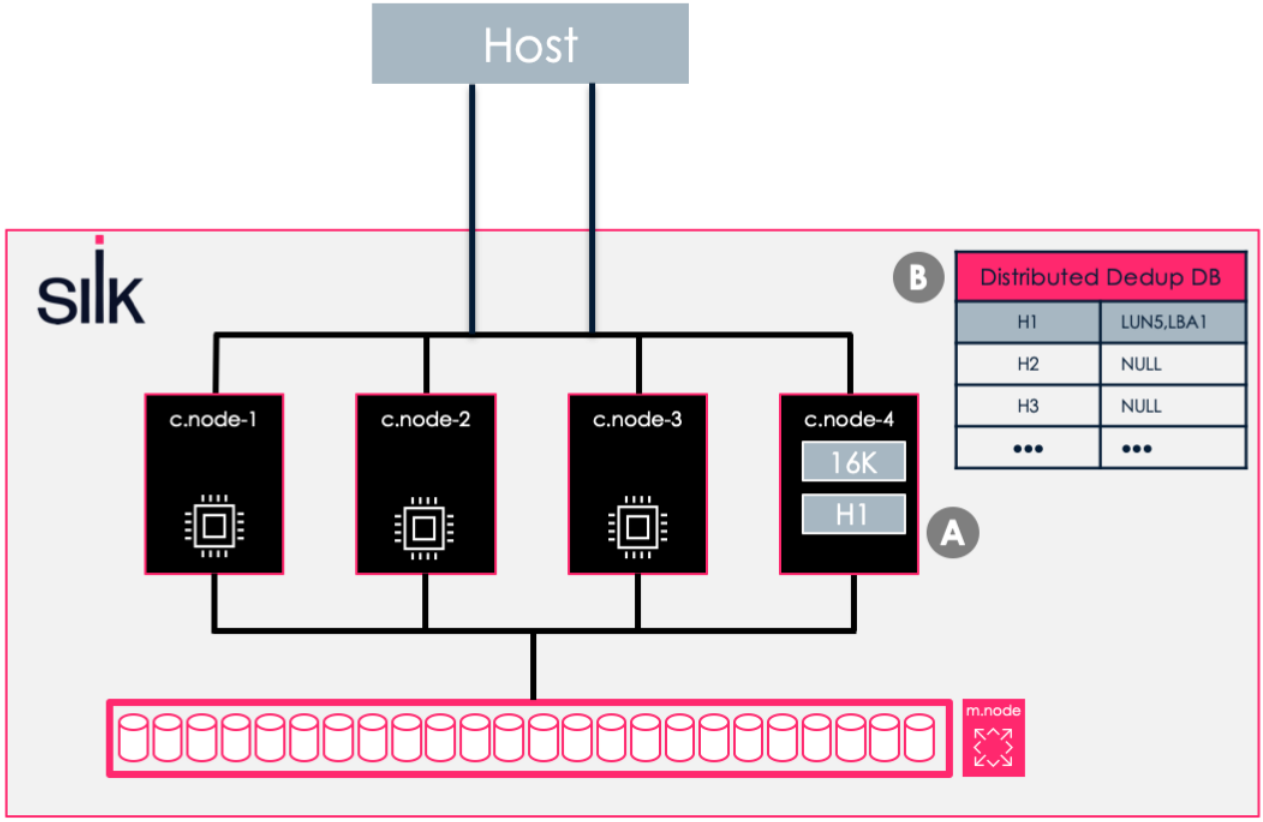


Dedupe Write

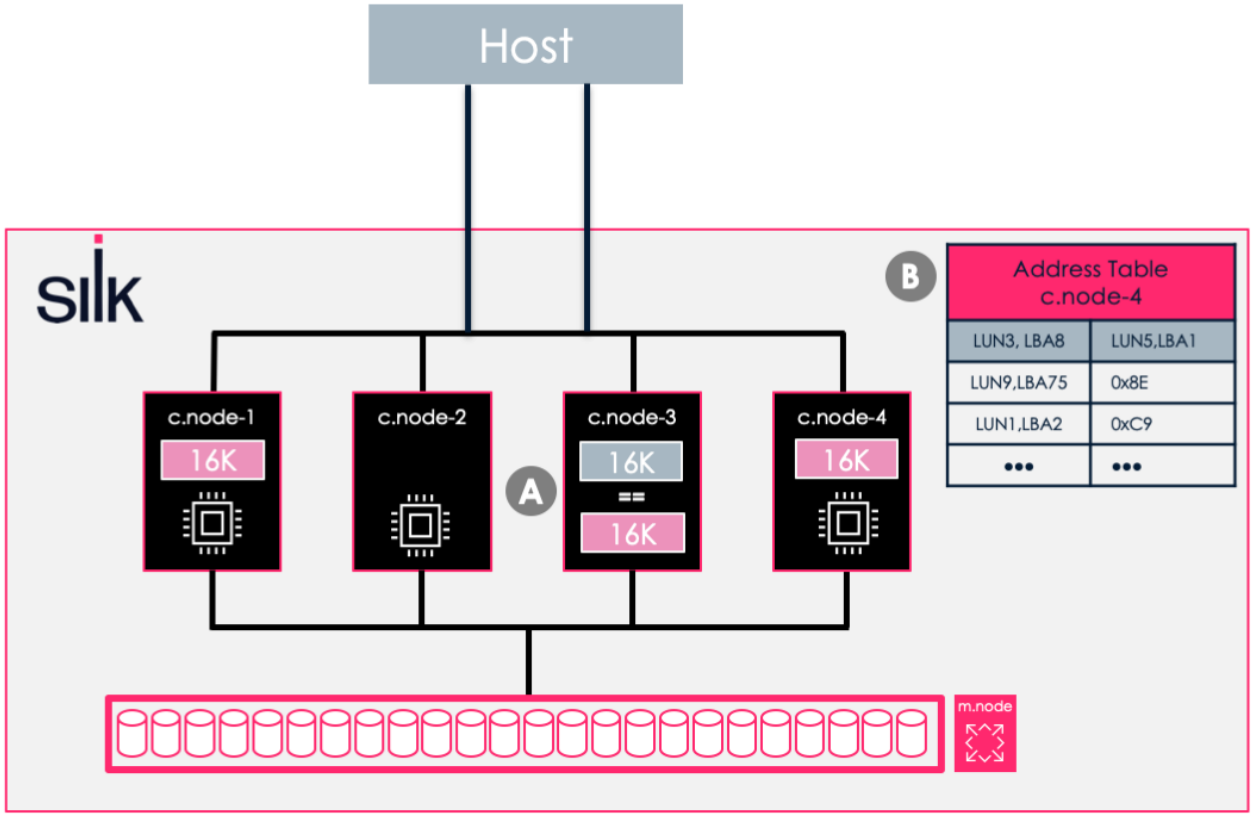
1. The same 16KB block is written again, with the following differences:
   - The write is designated for LUN3, LBA8 (A). It arrives at c.node1 and is stored in its DRAM (B).
   - The mapping table indicates that the write should be mirrored to c.node4. The block is mirrored over the InfiniBand fabric to c.node4 (C).
   - At this point, the block is stored in two different physical servers which have BBUs – c.node1 can now return an ACK to the application host (D).



2. On c.node4, a hash value (H1) is created for the 16KB block (A). The hash value is sent to the appropriate c.node, which is c.node1, and a lookup is performed in the hash table. C.node1 reports back that for the specified hash value, the address resolves to LUN5, LBA1 (B). Remember that previously the 16KB block that was written to LUN5, LBA1 was stored to the K-RAID by c.node3.

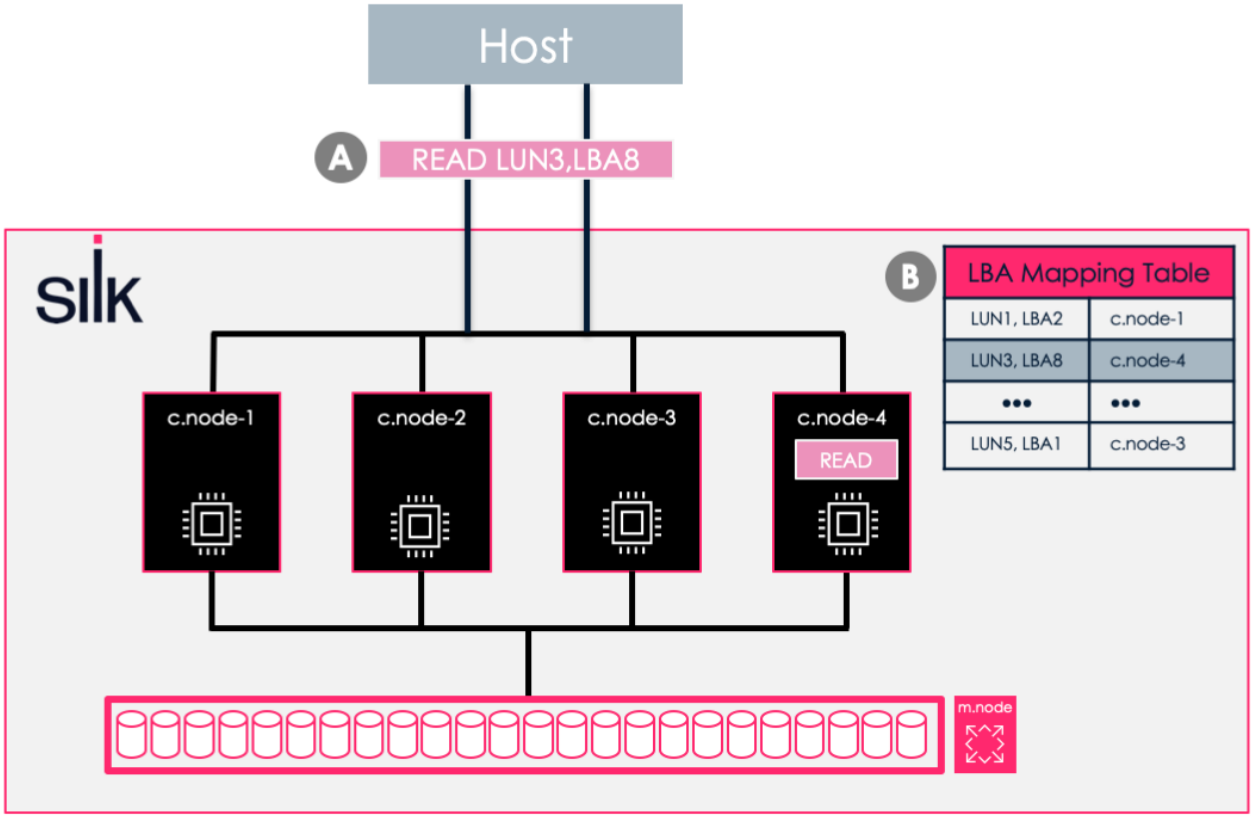


3. To avoid any chance of a hash collision, it is necessary to compare bit for bit the actual data. C.node4 sends the rehydrated 16KB block to c.node3 for a full compare (A). This requires a single read operation by c.node3 from the K-RAID, since it was stored as a single 16KB block. Once it checks out as an exact bit for bit match, c.node4 updates the Address table entry at LUN3, LBA8 to point to LUN5, LBA1 (B). These metadata updates are mirrored between c.nodes and eventually are persisted to the K-RAID to ensure that high availability of metadata is maintained at all times. At this point c.node1 and c.node4 drop the 16KB block from their DRAM and the dedupe write operation is complete.

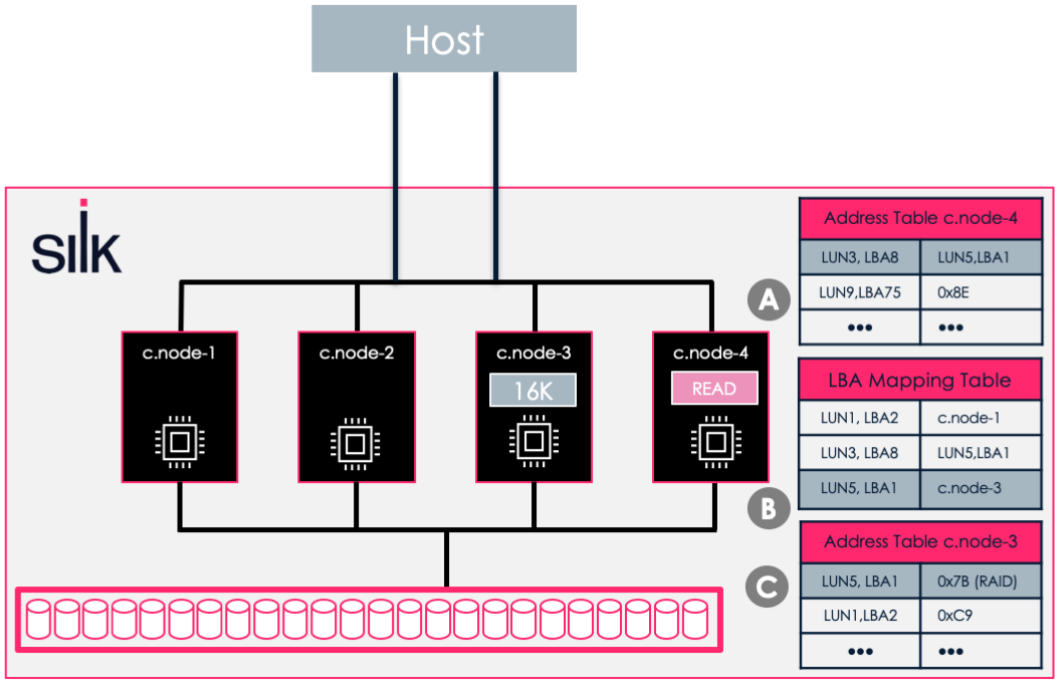


Read Operation

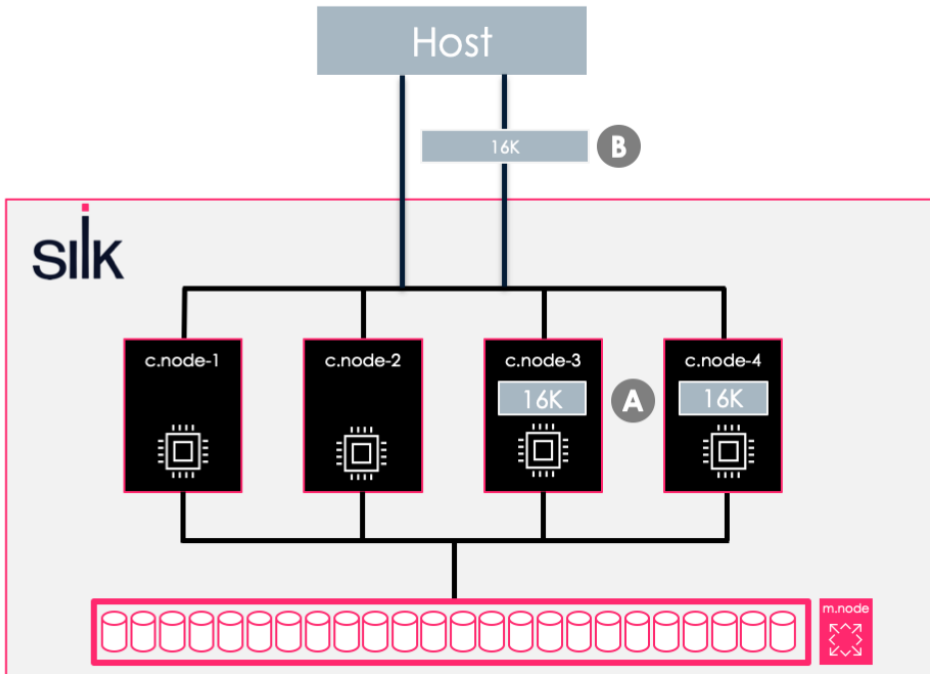
4. The application now reads 16KB from LUN3, LBA8 (A). This read request can arrive at any of the c.nodes due to MPIO. In this case, it is received by c.node4, which performs a lookup in its metadata Mapping table for the owner of LUN3, LBA8 (B). According to the Mapping table lookup, c.node4 happens to be owner of LUN3, LBA8.



5. C.node4 looks up LUN3, LBA8 in its Address table and finds a reference to a logical address of LUN5, LBA1 rather than a physical address (A). Using the Mapping table, it requests the data from c.node3 which is the owner of LUN5, LBA1 (B). C.node3 then looks up the physical address (C), retrieves the data, and rehydrates it from the physical SSD (D).



6. C.node3 sends the requested 16KB over the InfiniBand fabric to c.node4 (A). C.node4 sends the requested 16KB back to the application's host and the read operation is complete (B). Read responses go back out the same port they came in on. Data is moved east-west along the InfiniBand network as needed. The backend IB network is private and managed entirely by the Silk Platform. No end user management or maintenance is required.



Summary

The Silk high performance data platform, with next-gen VisionOS and Clarity software, offers users the performance and capabilities they need through a fully scalable and expandable architecture that fits your budget.

With a comprehensive set of data services features such as global inline selective deduplication, inline compression, thin-provisioning and the K-RAID™, Silk challenges the costs of other flash-based arrays and public cloud options on the market.

VisionOS provides the Silk Platform with a complete software stack of enterprise resiliency features such as high availability (HA), non-disruptive upgrades (NDU), snapshots, replication, and Silk Clarity for machine learning analytics and system optimization.

About Silk

Silk is the database supercharger – the smart platform that delivers game-changing database performance without changing a thing about your underlying apps or database infrastructure, whether you're running real-time transactional workloads or analytical workloads – so your entire stack runs 10x faster. And with always-on availability across regions, zones, and clouds, your databases keep going strong no matter what the cloud throws at you. Industry leaders like Priceline, Cisco, and Telefonica rely on Silk for unlimited cloud flexibility, unbreakable data resiliency, and the greatest database performance of their lives.

Visit www.silk.us to learn more.