WHITE PAPER

Silk Cloud Platform Architecture

January 2022

TABLE OF CONTENTS

Executive Summary

As more organizations move their data to the cloud, Silk is invested in ensuring that its Platform makes it easy to migrate and maintain data on the public cloud.

For organizations looking to leverage the public cloud, Silk offers the flexibility and high performance that users have come to expect, through its cloud agnostic platform.

With Silk, customers can dynamically compose data resources based on application requirements: allocating just the right amount of both performance and capacity while minimizing overprovisioning and waste. Furthermore, Silk offers a disaggregated, dynamic cloud architecture. Customers can scale capacity or performance resources independent of one another, thus delivering further flexibility to the business.

Silks Cloud Platform is a combination of tested and packaged software and services. The software stack is built on the software components of VisionOS™, Clarity, and Flex which provide a rich set of data services, machine learning, analytics, and policy-based automation and orchestration.

This white paper describes the Silk Cloud Platform architecture, detailing its core features and functionalities and how they come into play.

Introduction

The public cloud is a gamechanger. It offers enterprises a flexible platform and promises cost-efficient pricing. Yet it has its pitfalls. Applications may need to be refactored or modernized in order to be migrated to a specific cloud platform. For users of the public cloud, this essentially means that applications are limited to the platform they have been written for. Silk's Platform offers a shared framework where all data resources can be easily moved between the public and private cloud environments. This framework separates data from the infrastructure that it lives on, realizing the promise of true disaggregation.

Silk's scalable architecture and operating system are facilitators for driving cloud adoption to the next level with the following tangible business benefits:

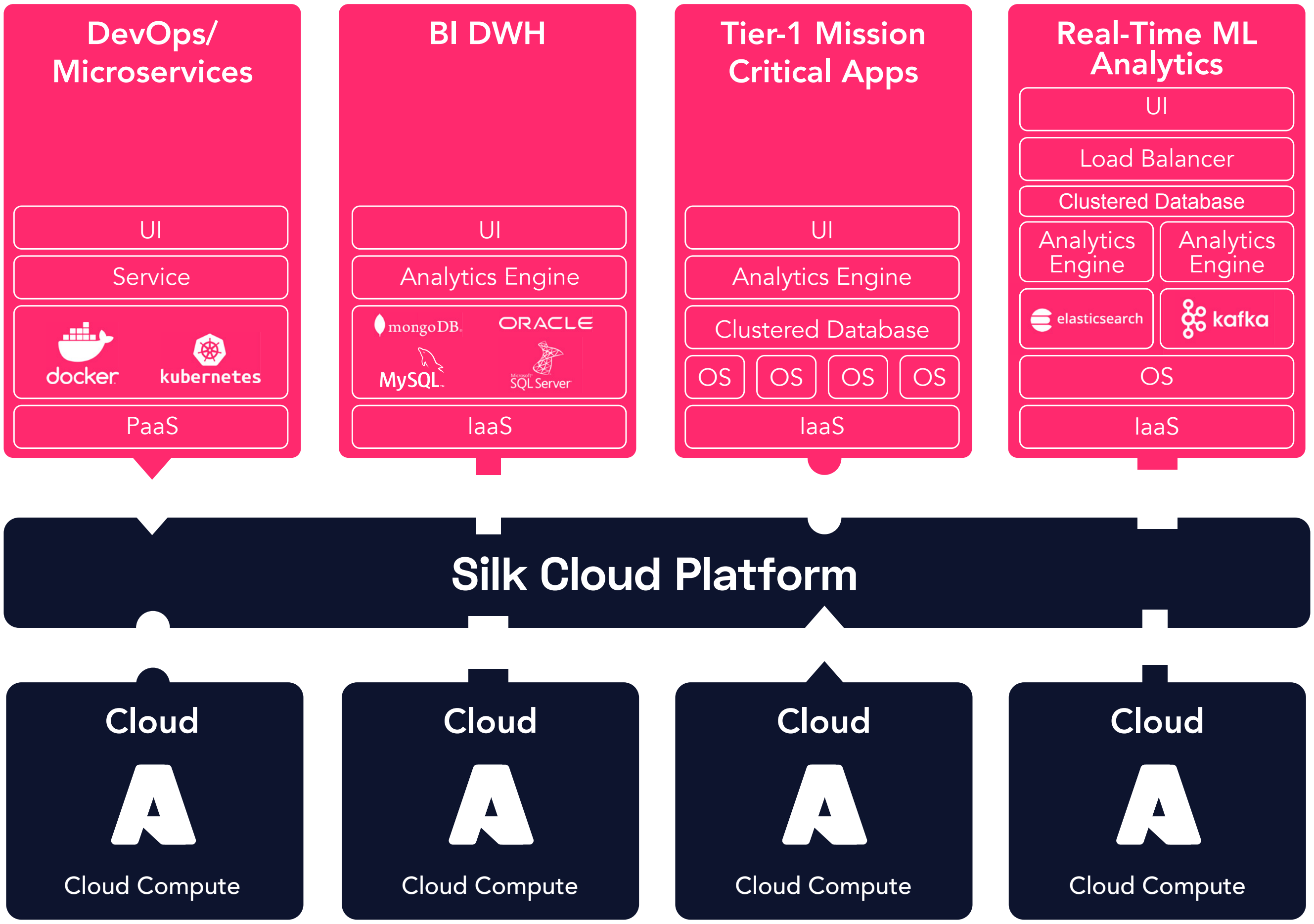
- **Performance** – Silk delivers unmatched consistent performance levels. Silk has a legacy as the top-rated solution for On-line Transaction Processing (OLTP), Analytics (OLAP), and High-Performance Compute workloads. The Silk Cloud Platform excels at serving mixed workloads via a patented global variable block size algorithm, delivers better user experience in VDI environments, removes the Input/Output (IO) blender effect from virtual servers, and allows customers to receive real-time reports from OLAP environments and faster database queries in OLTP environments -- all in a single platform. In addition, the flexibility delivered by Silk's platform enables enterprises to choose how much performance and capacity they need, while leveraging it on any cloud platform of choice.
- **Cost Efficiency** – Silk's goal is to make cloud adoption as cost-efficient and inexpensive as possible. Silk achieves this by disaggregating data from the infrastructure that it lives on. With Silk, customers can allocate the exact amount of performance or capacity needed by an application, thus minimizing waste. Capacity and performance can be dynamically provisioned and decommissioned or reallocated on any cloud to match changing application needs, further minimizing waste.
- **Scalability and Flexibility** – Silk's unique scale-up, down, in, and out architecture covers both dimensions of scale: capacity and performance. Silk can linearly scale the number of CPU cores by adding c.nodes (compute nodes) and independently scale capacity by adding m.nodes (media nodes) – thus breaking the limits of rigid architectures that are unable to scale-out or benefit from true shared metadata schema. Silk delivers the flexibility to add or remove underlying cloud resources on the fly, based on application requirements. Silk delivers multi-petabyte level scalability of shared capacity with data services and data reduction that span globally across all Silk resources and can be managed through a single pane of glass.
- **Resiliency** – Silk data volumes are packaged up into data pods for easy migration to different infrastructures. Each data pod can support a significant amount of performance and capacity requirements. If one compute engine goes down, the performance metric only decreases linearly. With Silk Flex, a failing resource can be identified before it fails and a new one can preemptively be provisioned from the available cloud resources.

System Overview

General

The best-in-breed combination of VisionOS, Flex, and Clarity are key components of The Silk Cloud Platform. This platform enables customers to quickly scale data infrastructure to achieve the performance and capacity that they need, along with the flexibility to meet these requirements on the underlying cloud infrastructure of their choice.

The platform efficiently manages data for any application type and environment offering a single platform that virutalizes multiple infrastructures.



High Level Architecture Overview

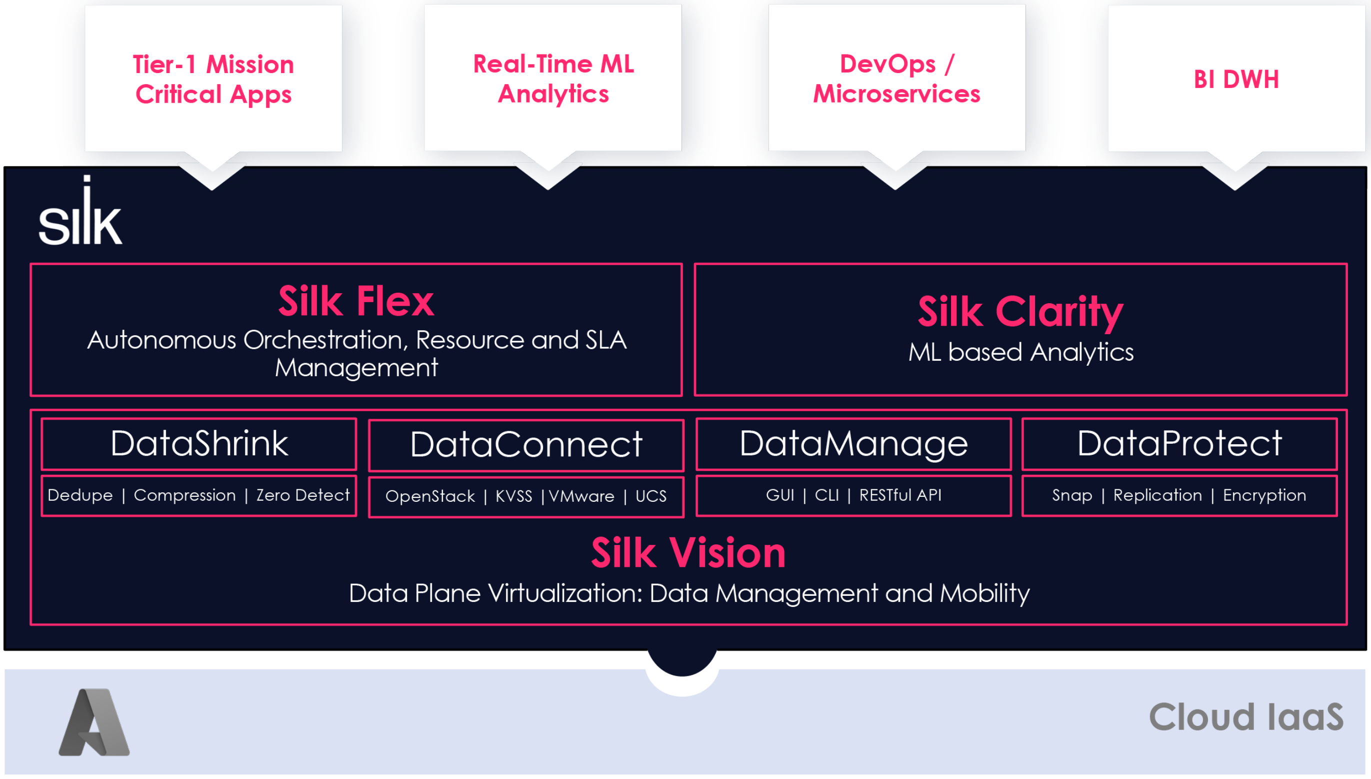
General

- **VisionOS** is a data hypervisor for resource virtualization and is built to help customers accelerate their cloud adoption journey. Running on any cloud, VisionOS delivers enterprise-class data services with a highly flexible symmetric active-active scale-out shared data framework. VisionOS turns any underlying cloud infrastructure into the world's most capable high performance scale-out data virtualization and mobility platform.

  VisionOS architecture is comprised of the following software elements:

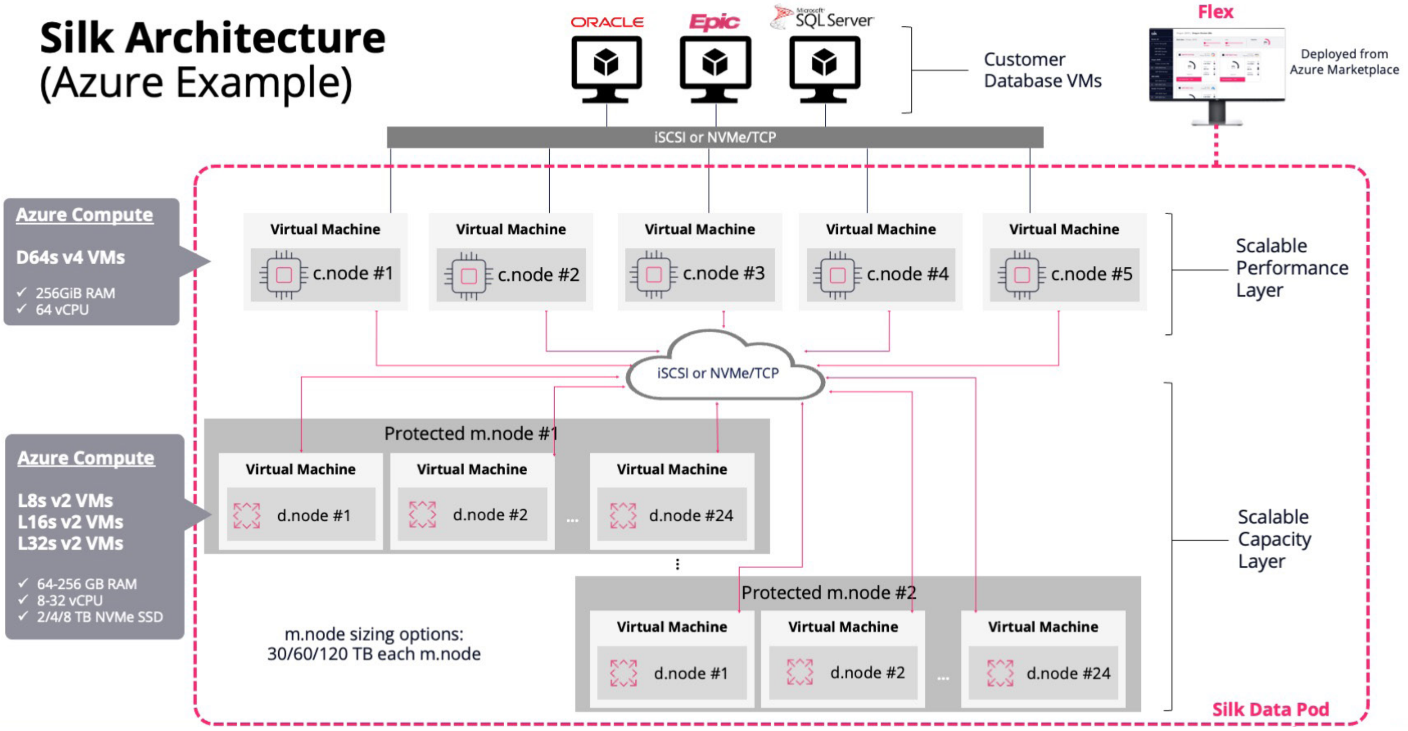
  - c.node (compute/controller node) - virtualization for data management and optimization
  - d.node (data node) - virtualization of a data endpoint
  - m.node (media node) - virtualization of data nodes management protected under K-RAID

- **Flex** is a data orchestration platform for managing resources in the public cloud. Flex delivers an on-demand ability to compose, optimize, manage, and decommission resources as needed to support application SLAs from within the Silk Cloud Platform. Flex orchestrates on-premises and cloud resources as new workloads emerge, move, and evolve over time.
- **Clarity** is a cloud-based Artificial Intelligence for IT Operations (AIOps) engine that delivers predictive analytics through a comprehensive set of management and monitoring functionality which analyze millions of data points on a daily basis from our install base. This includes the unique capability to leverage application-level intelligence, machine learning, and big data predictive analytics. Clarity is built to maximize business agility while minimizing risk. Clarity's advanced analytics provide recommendations on self-healing policies and preemptive resource optimization while enabling a whole new automated approach to managing business and application SLAs.


Tier-1 Mission Critical Apps
Real-Time ML Analytics
DevOps / Microservices
BI DWH
silk
Silk Flex
Autonomous Orchestration, Resource and SLA Management
Silk Clarity
ML based Analytics
DataShrink
Dedupe | Compression | Zero Detect
DataConnect
OpenStack | KVSS | VMware | UCS
DataManage
GUI | CLI | RESTful API
DataProtect
Snap | Replication | Encryption
Silk Vision
Data Plane Virtualization: Data Management and Mobility
Cloud IaaS

Public Cloud Architecture

The virtualized c.nodes and d.nodes are deployed on virtual machines utilizing compute engines in the different public clouds.



Public Cloud Resources

Silk resources for the public cloud encapsulate the following components:

- At least two c.nodes, with the ability to scale out to as many c.nodes as needed
- Each c.node is comprised of:
  - Up to 64 VCPUs
  - Up to 512GB DDR4 RAM
  - Up to 75Gbps connectivity
- Base m.node, with the ability to scale up to as many m.nodes as needed.
- Each m.node provides:
  - 24 logical data nodes with up to 8TB of physical capacity each, for serving reads and writes, parity protected by an erasure coding heuristic.
  - Each data node can be mirrored to persistent HDD, capable of sustaining the sequential log structure write pattern implemented by VisionOS for additional redundancy.

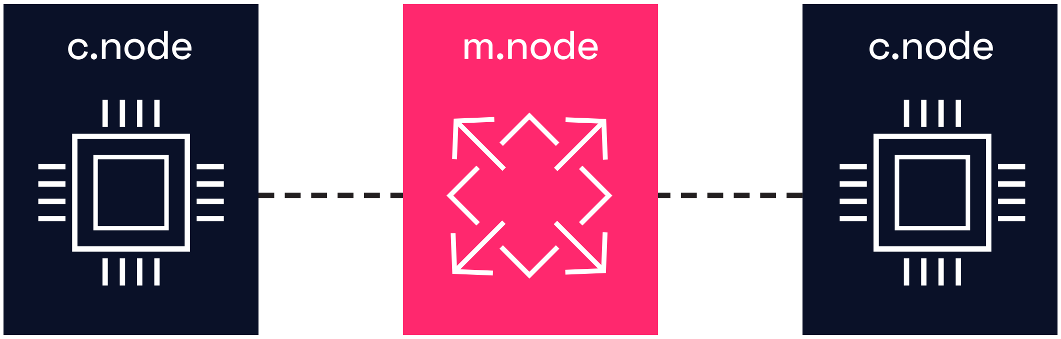
SCALABLE ARCHITECTURE

Silk's unique architecture is designed to facilitate online linear expansion in both capacity and performance separately while maintaining consistently low latency -- a scale-out and scale-up architecture. This online expansion can be additional controllers for performance or additional disks of capacity. The combination of both scale-out and scale-up architectures is the key for building an infrastructure that can scale in the most cost-effective manner, delivering the ability to meet the exact requirements for new and existing applications. Any increase in capacity results in an automated rebalancing of data within the data pod, with no intervention or human management.

The starting configuration for any Silk data pod configuration is two c.nodes and one m.node.

- Each c.node is connected to the m.nodes via one of many available networking protocols
- Volumes and metadata are automatically distributed between all m.nodes and can be accessed from every c.node in the data pod.

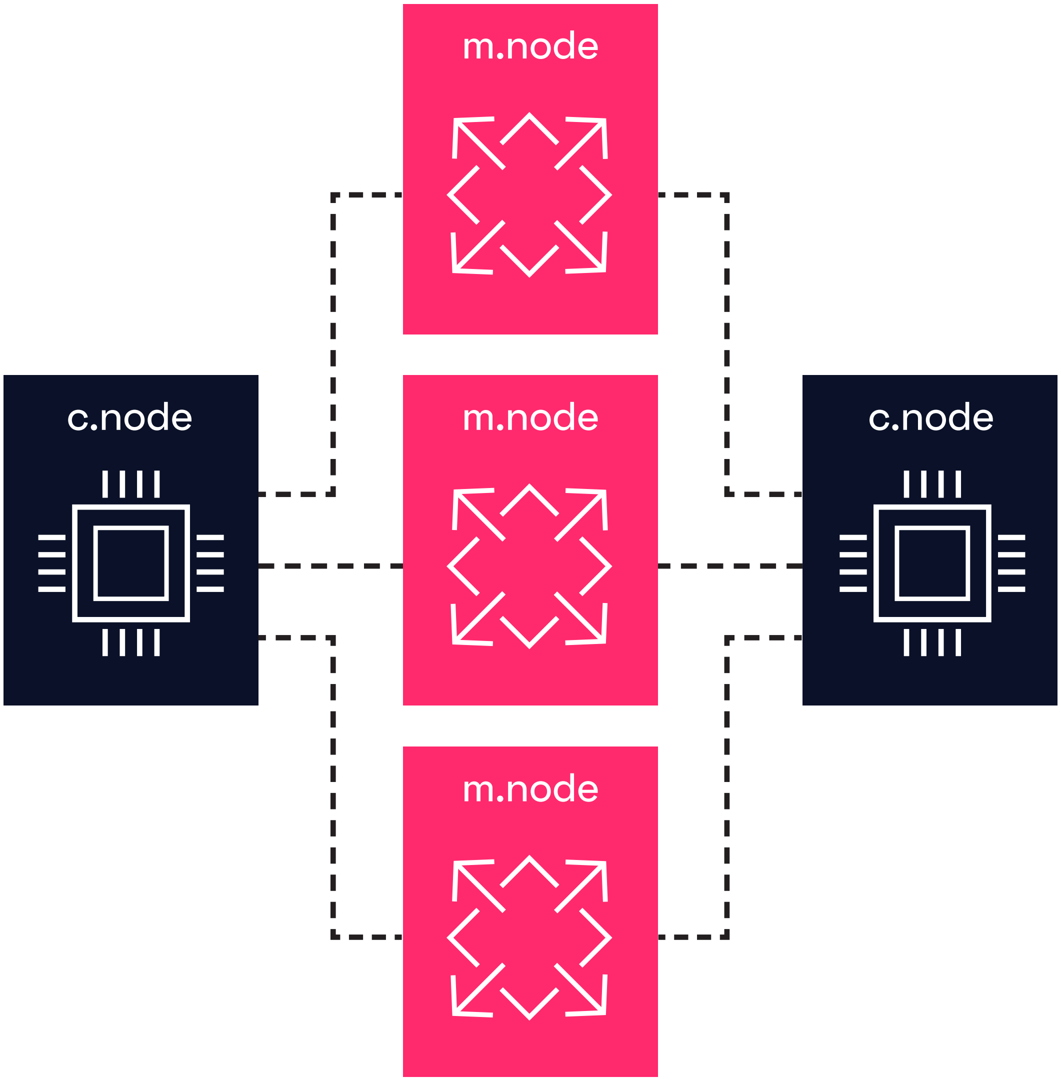
From this basic configuration, customers have the flexibility to scale up/down and out/in according to application needs.



SCALE-UP OR SCALE DOWN

Scaling up means simply adding more capacity by adding m.nodes without adding any other resources. Conversely, capacity can be scaled down easily when it is not needed.

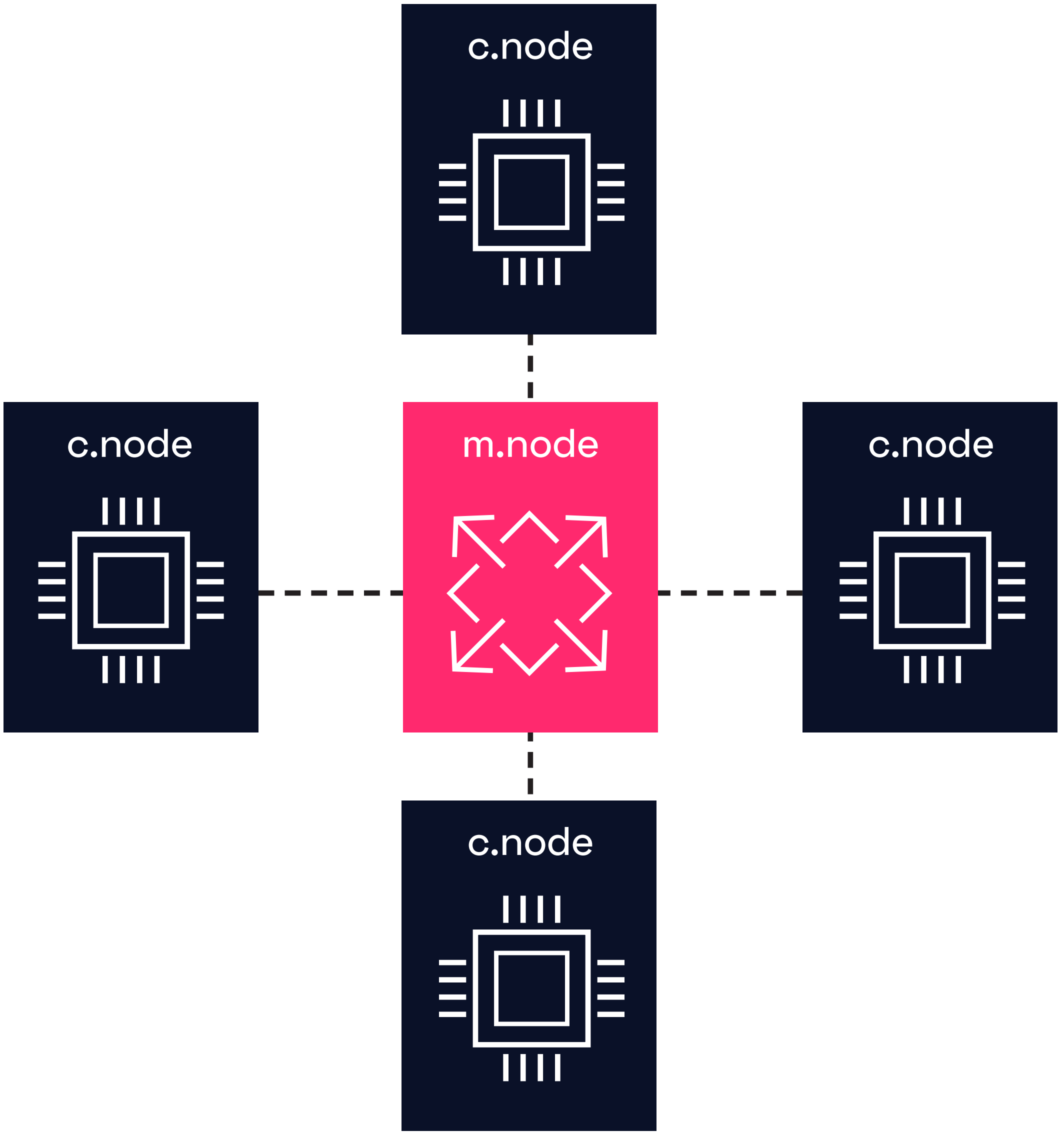
- The expansion increases the capacity density and reduces the cost per GB
- The expansion is done online with no downtime or decrease in performance
- The new configuration has the same performance as before
- Existing volumes, data and metadata are automatically redistributed between all the d.nodes in the data pod as part of the standard write handling
- There is no change required to any of the host connections or definitions
- Mix and match with the most up-to-date and cost-efficient cloud infrastructure technology
- Mix and match different capacity m.nodes



SCALE-OUT AND SCALE-IN

Scaling out means increasing the number of c.nodes within a data pod, without adding more m.nodes. Silk disaggregates performance and capacity, thus delivering the ability to add or remove c.nodes linearly in relation to the compute power you require.

- The number of c.nodes is disaggregated from the number of m.nodes
- Scale-out linearly increases the performance measures such as IOPS and throughput. The latency is kept consistently low and is indifferent to the expansion
- Scale-in provides the ability to reallocate or decommission resources
- Both scale-out and scale-in are performed online with no downtime.
- As all c.nodes are symmetrically active-active, global deduplication is consistent and supported across all resources within the data pod
- New hosts can be connected, and the new and existing hosts can access all new and existing volumes
- As c.nodes are added and subtracted, host connectivity is automatically adjusted to maximize performance



SCALE-UP AND SCALE-DOWN

There is no particular order in which the Silk Cloud Platform needs to scale. The platform can first scale up then scale out, and vice versa. This flexibility removes any compromises of using resources that are not tailored to the customer's needs or the cloud vendor's requirements.

- Scale-up increases the capacity density and reduces the $/GB factor
- Scale-down allows the customer to reallocate the capacity or retire that resource altogether
- Scale-up and scale-down are performed online with no downtime or decrease in performance
- Existing volumes are automatically redistributed between all d.nodes
- There is no change required to any of the host connections or definitions
- Scale with the latest and most cost-efficient cloud infrastructure technology

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 tables below capture the most common configurations and capacities for the public cloud:

Silk IN THE PUBLIC CLOUD

c.nodes	2 c.nodes	3 c.nodes	4 c.nodes	5 c.nodes	6 c.nodes	7 c.nodes	8 c.nodes
Usable Capacity*	12TB-0.5PB	12TB-1PB	12TB-1.5PB	12TB-2PB	12TB-2.5PB	12TB-3PB	12TB-3.5PB
IOPS	Up to 220K	Up to 330K	Up to 440K	Up to 550K	Up to 660K	Up to 770K	Up to 880K
Throughput	Up to 3.5GB/s	Up to 5.25GB/s	Up to 7GB/s	Up to 8.75GB/s	Up to 10.5GB/s	Up to 12.25GB/s	Up to 14GB/s
Latency	0.25ms						

Capacity is subject to drive size and the application's data reduction ratio.
For some datasets such as VDI the range will be higher. Latency based on 4K block size.

DataShrink

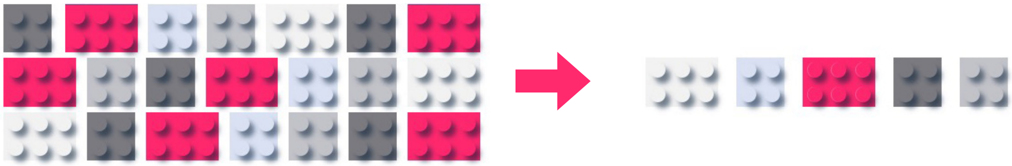
The ability to build a cost-efficient cloud platform relies almost entirely on the efficiency of the architecture or, in other words, how much effective capacity can be generated from the raw underlying physical SSD capacity.

VisionOS is focused on being highly efficient, but without compromising on features such as enterprise resiliency and consistent performance.

These features play a major role in the IO processing, as described in the IO Flow section of this paper.

DEDUPLICATION

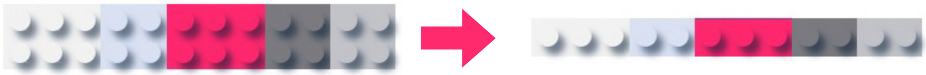
VisionOS's global inline selective deduplication meets the demanding requirements of eliminating redundant data so that it is stored only once. The deduplication is performed globally, and processing is distributed across all the c.nodes, enabling higher deduplication ratios, high performance, and consistent low latency. As the c.nodes scale out, so does the deduplication. In addition, VisionOS offers the option of selective deduplication. This allows data to be stored without deduplication for applications where data redundancy is negligible and additional performance is preferred (i.e. database applications like Oracle or SQL Server), or for security-sensitive applications where deduplication is prohibited.



COMPRESSION

VisionOS uses inline real-time data compression that is optimized for low latency performance. The data reduction is gained regardless of whether the data can be deduplicated or not, making compression the de-facto method of data reduction for non-dedupable data sets that are common in database environments, such as Oracle and SQL Server.

VisionOS uses byte-aligned compression algorithms with the ability to compress data in the granularity of a single byte which achieves greater compression ratios and eliminates any segment padding. The compression routine places a metadata marker every 4KB block rather than on bigger data segments, ensuring that small reads do not result in hydration of unnecessary data. 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.

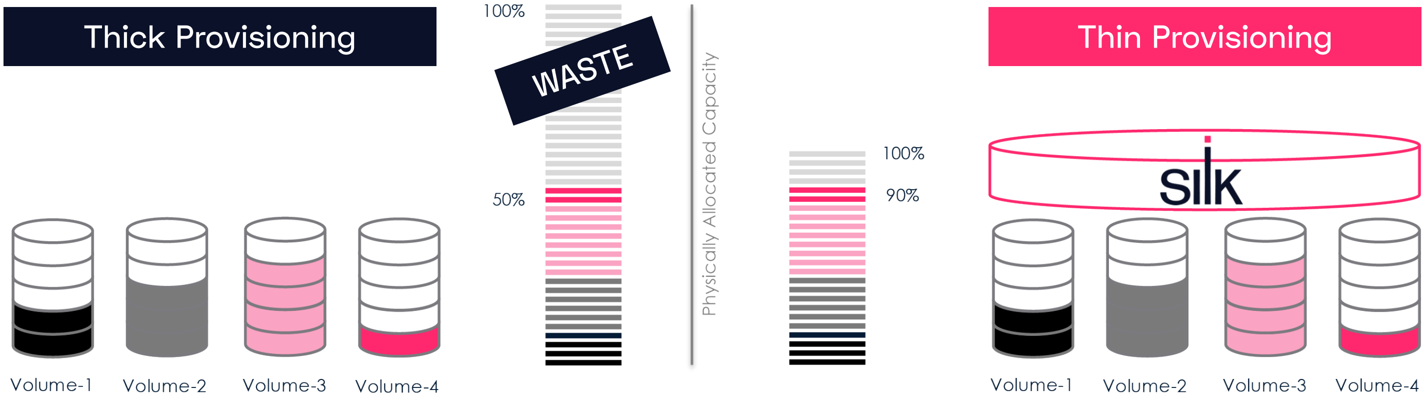


ZERO ELIMINATION

VisionOS uses inline zero elimination to avoid storing data that consists of zeros. Instead, VisionOS stores a metadata flag to indicate that a certain data address is logical zero. Zero elimination is done in a fine granularity of 4KB. Many applications either format the data upon initialization or store large sections of zeroes as a part of the data. Using zero detection, such applications will not consume valuable space within the data pod. Zero elimination also provides fast completion of format and similar types of operations commonly performed by applications.

THIN-PROVISIONING

Thin-provisioning allows for maximum utilization of the resources with the ability to plan capacity provisioning for the long term. However, thin-provisioning can be truly utilized only with a scalable architecture that can facilitate capacity growth within the data pod, with no limitations. All the volumes in the data pod are thinly-provisioned, regardless of the underlying public cloud platform (public cloud by default is thick provisioned, and cannot support native thin-provisioning); with fine granular on-demand growth of 4KB. Unmap (trim) operations are also supported in the same granularity. 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.



DataProtect

The Silk Cloud Platform is architected and built to meet the requirements of the most sensitive enterprise applications. With High Availability (HA) throughout the design, Silk offers scalability of fault domains and provides the right features for building an enterprise product. In addition, data is protected from misuse or human error at the application level using native features such as snapshots and replication.

RESILIENCY

The Silk Cloud Platform offers excellent performance resiliency. Since a data pod can have any number of c.nodes, if one c.node goes down, the performance metric only decreases linearly as a percentage of the total number of c.nodes active – reflecting the capabilities of one c.node. The beauty of dynamic scalability combined with ML-based monitoring is that Flex makes it possible to identify a c.node before it fails and allocate a new c.node from the underlying platform, or from other Silk data pods which can handle their workload with fewer c.nodes, proactively and automatically.

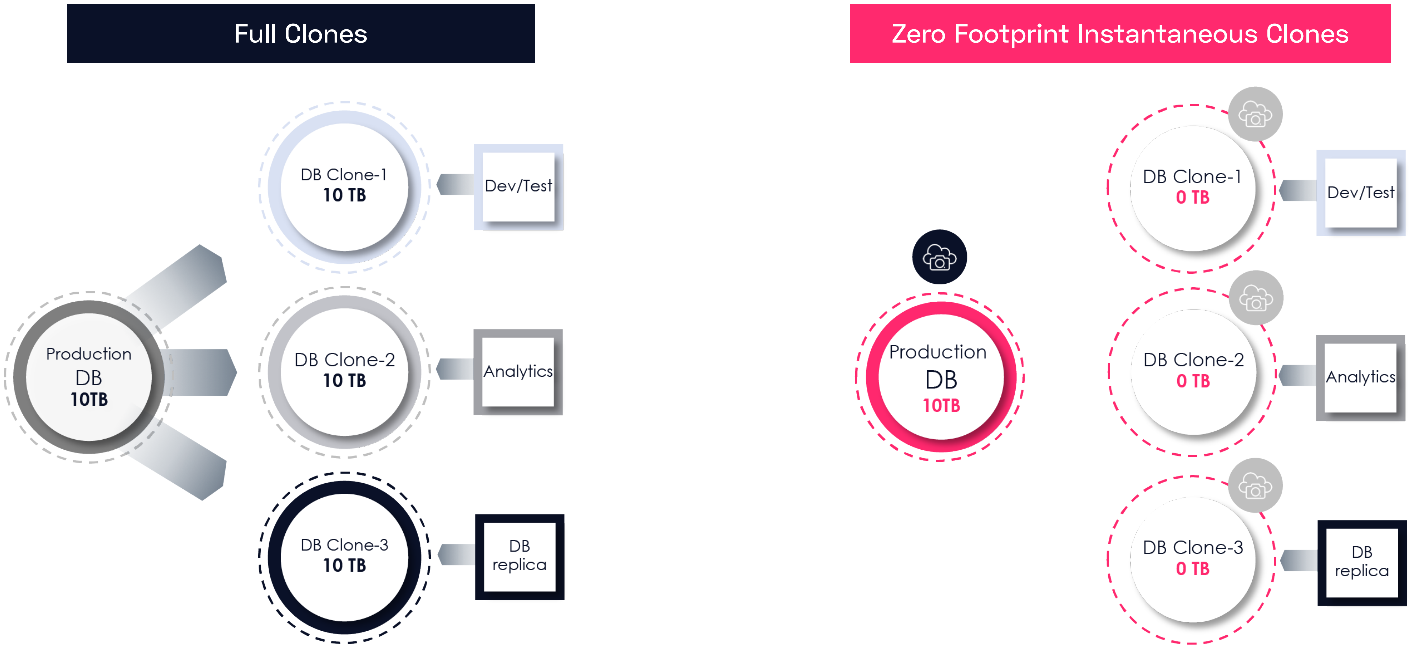
The time it takes to replace or rebuild a failed d.node is also shortened dramatically as the rebuild work is distributed across all c.nodes. As a result, rebuild time reduces linearly to the number of c.nodes.

SNAPSHOTS

Silk's patented snapshot architecture follows VisionOS guidelines of efficiency, performance, and scalability. Snapshots are created instantly, with no performance impact, and they do not take up any capacity. Snapshots track only the deltas from the source volume in a 4KB granularity, using a redirect-on-write (RoW) approach. This efficient design also keeps the impact on SSD endurance to a minimum. Snapshots can be mounted for read/write purposes, which serve to create additional working environments such as QA, Test & Dev, analytics, backup, and more… all at a very low cost of capacity. Read/write snapshots deliver the same performance of the production volumes, without any impact on the actual production volumes.

The duration of creating a snapshot has no dependencies on the number or size of the volumes being snapped or on the size of the data pod. Using the snapshots to restore functionality for recovery purposes is done without losing any of the snapshot's history and is allowed at any time. The snapshots can be accessed from any of the c.nodes, without bottlenecks or load balancing of affinity to a specific controller.

Snapshots can be used to create Read/Write zero-footprint instantaneous clones so that the same dataset can be used for multiple purposes without the need to create full copies of the same data.



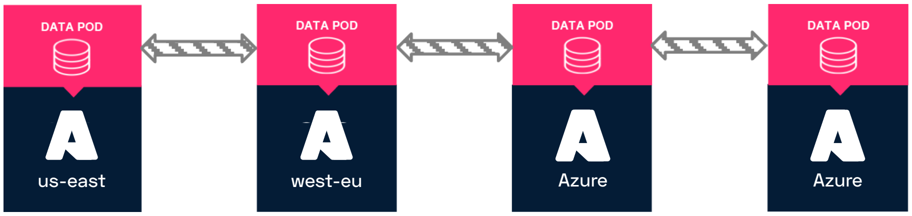
REPLICATION

Replication provides site resiliency that meets enterprise datacenter resiliency requirements. Silk leverages its snapshot architecture to facilitate asynchronous snapshot-based replication between data pods. Since the replication is based on the snapshot architecture, there is no impact on the data set's consistent high performance, while deltas of the replicated copies are captured with no dependencies on the link speed.

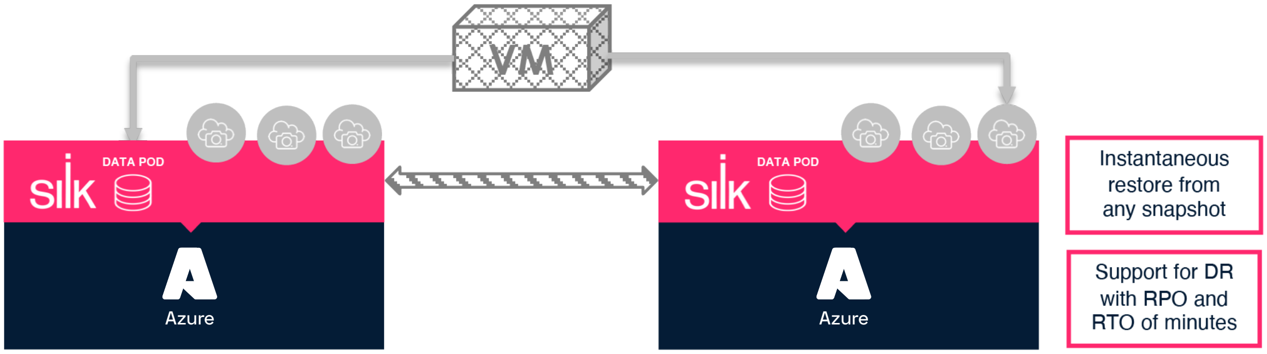
The VisionOS features of deduplication and compression are also used to significantly reduce the amount of data sent between data pods.

All essential disaster recovery capabilities such as a small Recovery Point Objective (RPO) and Recovery Time Objective (RTO) are gained natively without any third-party software components, which are costly and add overhead on the data flow.

Multi-Cloud/Zone/Region Replication



Fast-Restore using Replicated Snapshots



Instantaneous restore from any snapshot

Support for DR with RPO and RTO of minutes

ENCRYPTION

For the public cloud, Silk layers additional layers of abstraction on top of the Encryption at rest capabilities of the cloud provider. Silk virtualizes, compresses and shards the data over the set of 24 d.nodes uses our patented erasure code, making it even more difficult to reconstruct. Please refer to the cloud vendor for details of their encryption at rest and cryptographic erase after use capabilities for the physical media.

K-RAID™

The Silk K-RAID parity protection algorithm is both efficient and extremely robust. It can sustain two concurrent d.node failures while maintaining data availability and up to three concurrent d.node failures within each separate m.node without loss of data. As the Silk data pod scales capacity, so does the number of system-wide d.node failures that the system can sustain. The K-RAID has triple parity protection that adapts according to the failure at hand. An d.node failure is quickly recovered thanks to efficient metadata and real-time system health monitoring. It has minimal performance impact during the rebuild and no performance impact on Silk's performance once the rebuild is completed. Parity overhead from raw capacity is a very small 12.5%.

K-RAID is built from two logical RAID groups, each one with a parity (P1 and P2) and an additional parity for the two groups (Q), as show below :

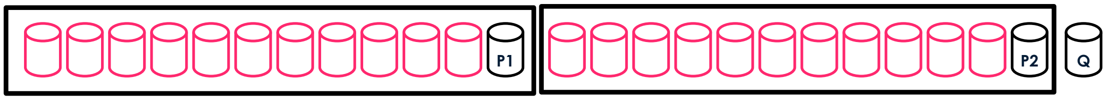


Figure 5: K-RAID logical layout of two RAID groups and a Q parity for both groups

The K-RAID layout is pseudo-randomly distributed throughout each m.node, meaning that there is no fixed parity or data role per d.node and there is no hot spare, meaning that all the d.nodes are utilized for protection and performance at all times.

The K-RAID is fully automatic and does not require any configuration or human intervention, which means another IT task is offloaded to the Silk data pod.

K-RAID FLOW

1. The flow of how K-RAID works in the scenario of an d.node failure is shown below:

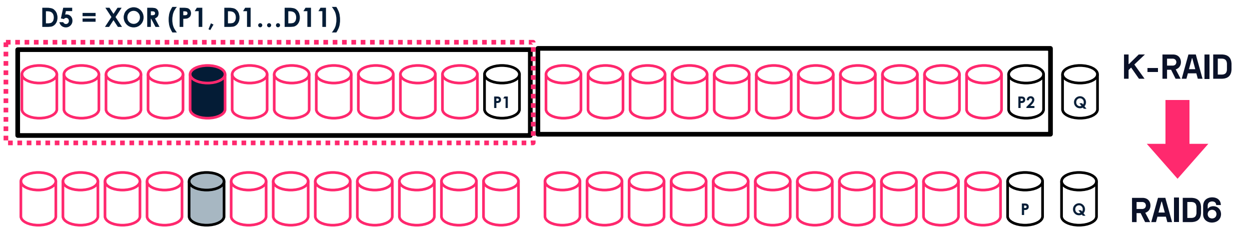


Figure 6: K-RAID flow during d.node failure

- The d.node failure is shown in the left RAID group. However, since the K-RAID layout changes within the m.node every 64MB, 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 be 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 d.node is performed from all the 23 healthy d.nodes, which results in considerably faster rebuild times.
- When reading from a RAID group that lost a data segment, the XOR calculations are performed only from that affected RAID group, which is built roughly from half the d.nodes in the m.node, meaning 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 RAID6 scheme, meaning it is still capable of handling two more concurrent d.node failures without losing data.
- When the failed d.node is restored, the K-RAID will be restored to its original layout.

2. The flow of how K-RAID works in the scenario of a second d.node failure is shown below:

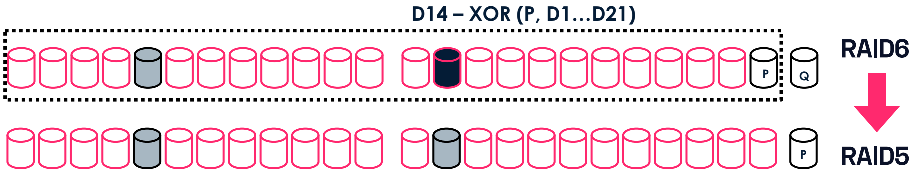


Figure 7: K-RAID flow during second d.node failure

- This is a standard RAID6 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 parity (P).
- Once the rebuild is complete, the K-RAID falls back to a RAID5 scheme, meaning it is still capable of handling an d.node failure without losing data.
- When the failed d.nodes are restored, the K-RAID will be restored to its original layout.

3. The Q parity – handling concurrent failures. When two d.nodes 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:

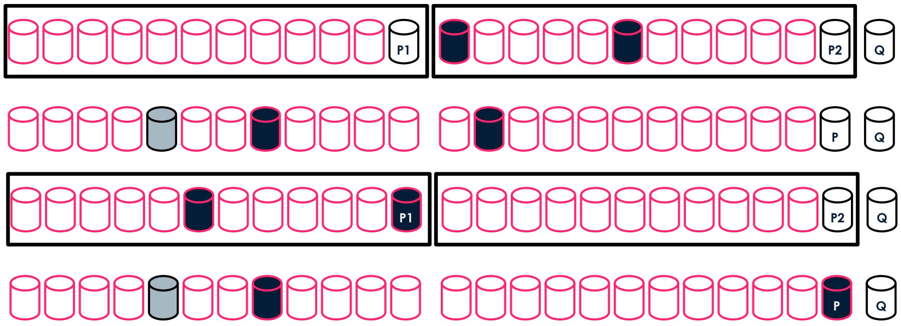


Figure 8: Using the Q Parity for two concurrent d.node failures

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

NO SINGLE POINT OF FAILURE (SPOF)

VisionOS supports a double-everything approach for all Silk data pods. Metadata at rest is protected by the dual-parity K-RAID™. However, Silk does not have passive or idle components; all of its resources are utilized at all times. There is full redundancy of every component in the system – both virtualized and physical -- and there is not a single component that can fail and cause unplanned down time or data loss. Silk data pods will stay online after the loss of any number of c.nodes above a predefined threshold for a minimum number required for operation.

NON-DISRUPTIVE UPGRADES (NDU)

Silk can upgrade any of its components with no impact on the availability or performance. The architecture is software-defined, so any new feature/enhancement/bug-fix/firmware to VisionOS can be deployed with no dependencies on maintenance windows or running workloads. In addition, hardware can be replaced/upgraded/added in the same manner.

Having NDU combined with a scalable architecture, Silk provides the best TCO: No fork-lift upgrades, no need to plan downtime. New technologies can be deployed in existing data pods, and growing application needs can be met by adding more capacity and/or performance. All of these operations are performed non-disruptively and automatically, with no human intervention.

DataManage

VisionOS provides a rich set of tools to monitor and manage the Silk Cloud Platform. These management services do not require additional hardware resources, since they are designed to natively run on provisioned c.nodes. With DataManage, there is no need to configure RAID groups, tune the data pod to a specific application, or create affinities between volumes and controllers.

GUI

The Silk user interface is slick and highly usable, keeping all the necessary information in a tiled dashboard. Each tile presents the essential details of Silk's management components: Health, Volumes, Replication, Performance, Capacity, and Events.

Clicking a tile will automatically open a more detailed and actionable view of that management component: Create volumes, snapshots, replication policies, capacity policies, assign hosts, manage host connections, and so on. The tiles are dynamic and can be rearranged according to personal taste. The GUI is accessible via any web browser and does not require any software installation or separate licensing.

CLI

The CLI is fully scriptable and allows full control of the features and functionality of the Silk Cloud Platform. 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 with examples on how to develop automatic flows.

POWERSHELL

Silk has a full SDK supporting Microsoft Powershell that includes cmdlets for all the native platform functions of Silk data pods.

DataConnect

Using the RESTful API platform, VisionOS allows 3rd party components and independent software vendors to monitor and manage the Silk Cloud Platform in the private cloud. In addition, other protocols such as SNMP, SMI-S and Syslog are also used with different platforms. The following details several examples of integrations with leading vendors in the enterprise IT industry:

CSI (CONTAINERS STORAGE INTERFACE) PLUGIN

The CSI Plugin is a native interface between Containers orchestration platforms such as Kubernetes and the Silk Cloud Platform. The plugin is simple to use and to install and is based on industry standard software infrastructure.

OPENSTACK

OpenStack's support for agile and scalable deployments of the cloud matches the scalable architecture of the Silk Cloud Platform. Leveraging Silk's RESTful API, the Cinder driver allows OpenStack environments to provision all-flash datasets for public clouds.

CISCO UCS

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

Silk Flex

Silk Flex is a data orchestration feature for managing resources in a Silk implementation. With Flex, customers can dynamically compose, optimize, manage, and re-allocate resources with no physical reconfiguration.

Orchestration can be performed in multiple manners. A user can perform orchestration operations using the Flex GUI and CL; infrastructure-as-code can perform orchestration operations using the RESTful API provided by Flex; and Flex can autonomously perform orchestration operations based on predefined schedules, policies, and AI-powered SLA-driven decisions. Resources can be orchestrated as new workloads emerge, move, and evolve over time whether they need to be scaled up, out, in, or down. Flex leverages Silk VisionOS to virtually associate c.nodes and m.nodes and build shared data assets that deliver industry-leading capabilities. Silk Flex delivers enterprise-class capabilities but with a game-changing level of flexibility. Resources can be built and managed with the swipe of a finger or a simple line of code.

Silk Clarity™

Silk Clarity is a cloud-based AIOPs engine that includes a comprehensive set of machine learning, management, and monitoring functionalities, including a unique capacity to leverage application-level intelligence, and big data analytics – all of which enable customers to get more productivity out of their environment and deliver higher performance for business-critical applications.

Clarity extends the capability of the Silk Cloud Platform to make it one of the most advanced cloud data management engines in the industry. Clarity's big data platform collects millions of active call-home data points from its customer base to drive automation and predictive alerting. Using advanced analytics and modeling, the platform helps uncover new insights and provides recommendations on preemptive resource utilization. Clarity is tightly integrated with Silk VisionOS and Flex, helping to gather, report, and act on aggregated performance trends, capacity utilization, data protection metrics, and real-time events.

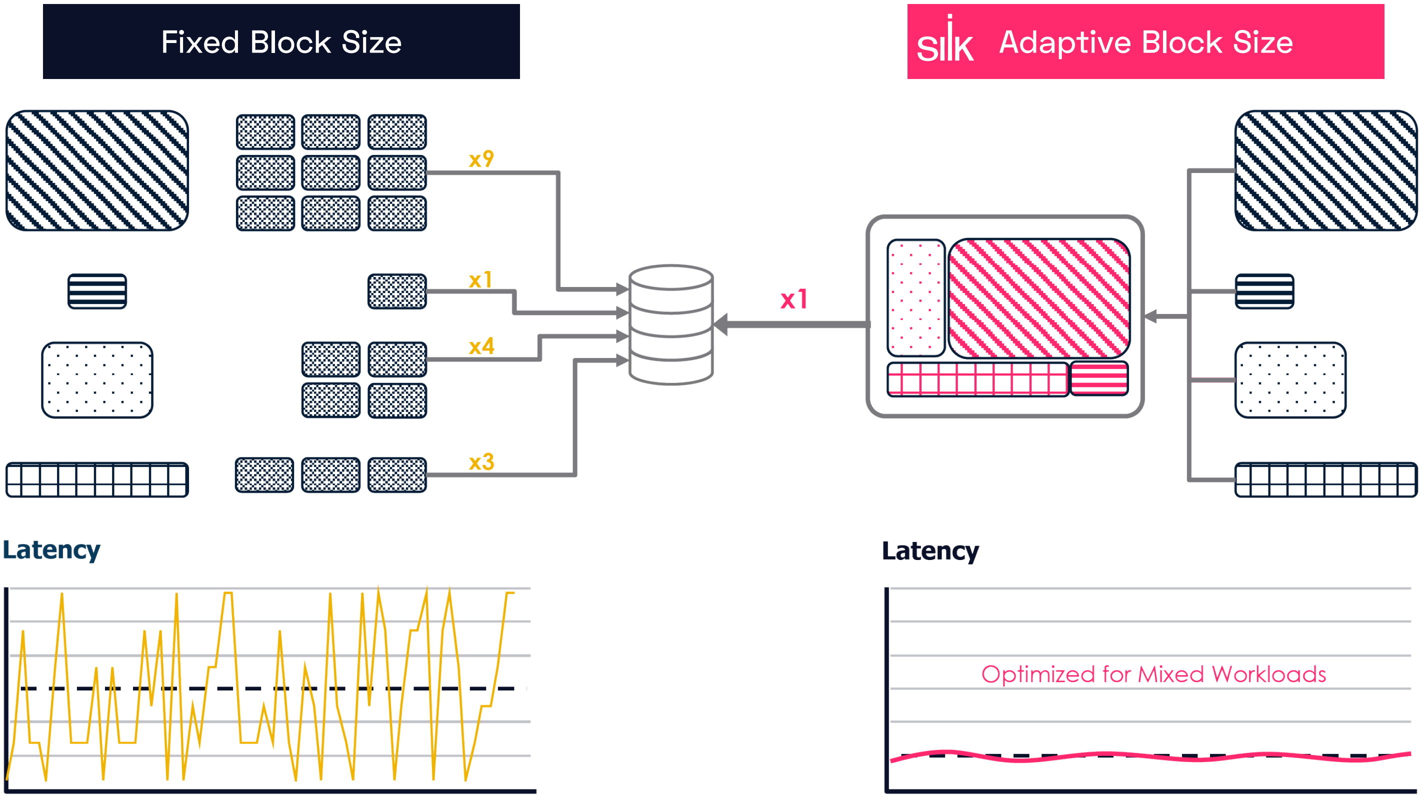
IO Flow

The core functionality of VisionOS is the IO flow. In simple words: it is how data is written to the data pods and how it is read. From the user's point of view, Silk accomplishes these tasks efficiently, reliably, and fast.

When taking a closer look at the IO flow, it can be observed that it encapsulates advanced technologies such as global inline selective deduplication, global adaptive block size, inline compression, distributed metadata, K-RAID, and more – all developed in a true scale-out architecture. This section details these technologies.

GLOBAL ADAPTIVE BLOCK SIZE

Workloads generated by real applications vary in their block size. Resources – specifically those that deploy deduplication – tend to use a fixed block size (usually 4KB, 8KB or 16KB), thus fragmenting the application's data blocks into small chunks or having to pad out smaller blocks with zeroes to fill out the larger block alignment requirements. This method results in limited bandwidth for the application and multiple IOs for each IO that is bigger or smaller than that fixed block size. The uniqueness of VisionOS is that it adapts dynamically to the application's block size, which in return generates the best performance for the application's real workload without compromising latency, IOPS, or throughput. The global adaptive block size algorithm allows Silk to support the real performance requirements of a multitude of application types all running concurrently, which is the core essence of data pods. This patented algorithm is crucial for deploying true scale-out resources.



METADATA MANAGEMENT

Metadata management is essential and critical in any data system; however, its significance grows tenfold when deploying a data system that supports scalability on one hand and features such as snapshots, deduplication, compression, and thin- provisioning on the other. The way VisionOS manages the metadata results in better performing deduplication and compression -- while allowing real scale-out active-active access to all volumes and snapshots from all c.nodes. It also facilitates fast recovery in the event of a failure scenario and an optimized garbage collection process.

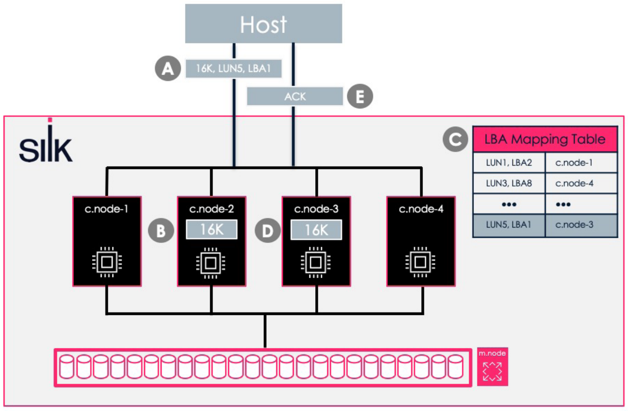
Metadata is kept both on DRAM and SSD media, using a unique cache algorithm for consistent performance. This advanced choice of media for metadata placement eliminates any restriction on the capacity size of the SSD and any restriction on the maximum data reduction ratios.

FLOW OF OPERATIONS

The flow below demonstrates a simplified IO flow of write and read. In the flow, a 16KB block is used. It shows how the global adaptive block size algorithm works in scale-out with a real application load. However, this is true also for larger block sizes as well and, of course, when hundreds of thousands of IO operations are running concurrently through the platform, additional mechanisms -- such as batching and queuing -- take place to provide further optimizations.

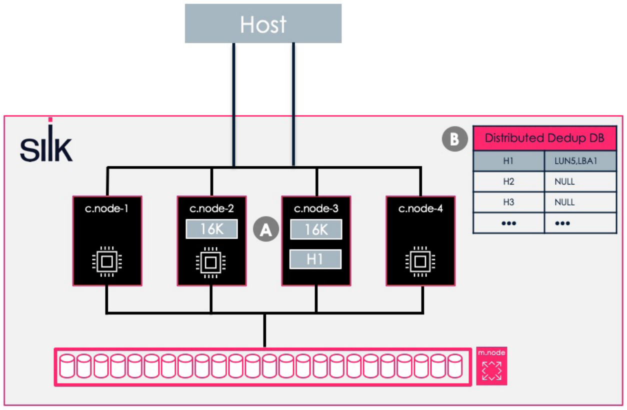
Unique Write

1. An application writes a typical block of 16KB to a logical block address (LBA1) within a block device (LUN5) on data pod (A). This block can arrive at any of the c.nodes, since the data pod uses a scale-out active-active scheme. In this case, the block arrived at c.node-2. Once the block is stored in the c.node (B), it must be stored in another c.node before returning an acknowledgement (ACK) to the application. 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, c.node3 is selected (C). This mapping table is identical in all the c.nodes. The block is then mirrored to the second c.node, c.node-3, which also stores the block (D). At this point, the first c.node can now return an ACK to the application's host (E). This means that any subsequent work performed while storing the new block is asynchronous and allows for low host-side latencies.



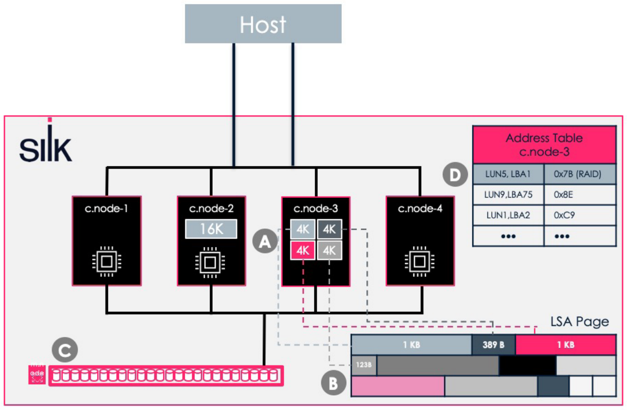
LBA Mapping Table	
LUN1, LBA2	c.node-1
LUN3, LBA8	c.node-4
•••	•••
LUN5, LBA1	c.node-3

2. On c.node-3, the deduplication process starts. Deduplication is distributed. A hash value H1 of the 16KB is created (A). All possible hash values are divided into several hash tables, according to the number of c.nodes in the data pod. The hash value is sent to the appropriate c.node, and a lookup is done for the hash value H1 (B). This will give a first indication of whether the respective 16KB was already written to the system. Since this is a unique write to the data pod, 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. The c.node that originated the hash lookups (c.node-3) is going to take ownership of the 16KB. 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.



Distributed Dedup DB	
H1	LUN5,LBA1
H2	NULL
H3	NULL
•••	•••

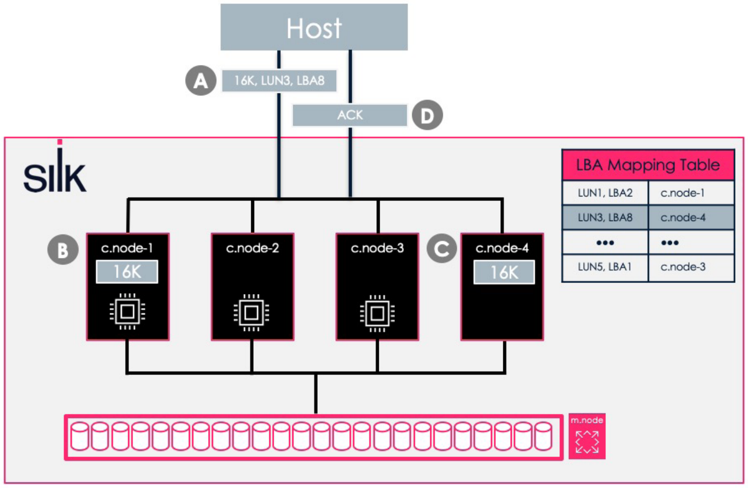
3. The 16KB is now compressed: It is marked into 4KB segments (A) and each 4KB is compressed separately - to the nearest byte (byte-aligned compression). The compressed 16KB is now placed contiguously in a log-structured data pod (LSA) page (B). This page is 3 MB in size. Once the LSA page has been filled, c.node-3 prepares a full RAID stripe calculating the parity and writes it to the K-RAID (C). c.node-3 keeps the location of the compressed 16KB in its address table, which translates (LUN, LBA) pairs to physical addresses on the K-RAID (D). Once the 16KB is stored to the K-RAID, both c.node-2 and c.node-3 free the 16KB block.



Address Table c.node-3	
LUN5, LBA1	0x7B (RAID)
LUN9,LBA75	0x8E
LUN1,LBA2	0xC9
•••	•••

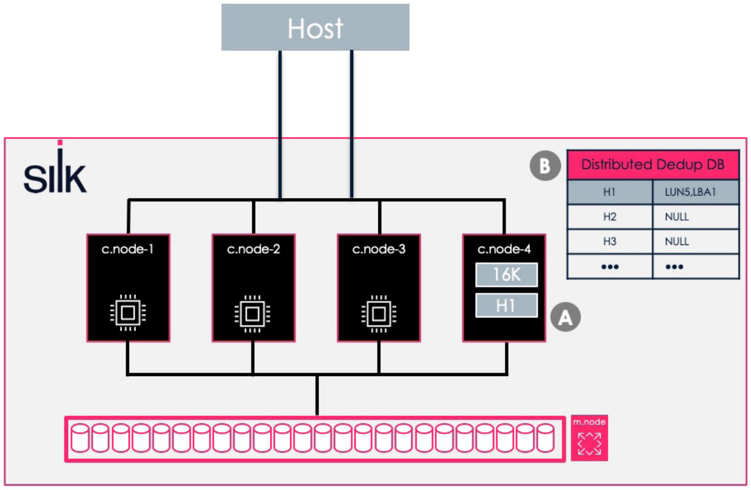
Variable Dedupe Write

1. The same 16KB block is written again to the data pod, with the following differences: The write is designated for LUN3, LBA8 (A). It arrives at c.node-1 and is stored (B). The mapping table indicates that the write should be mirrored to c.node-4 (C). The block is mirrored to c.node-4. At this point, the block is stored in two different c.nodes which have battery backups, c.node-1 can now return an ACK to the application's host (D).



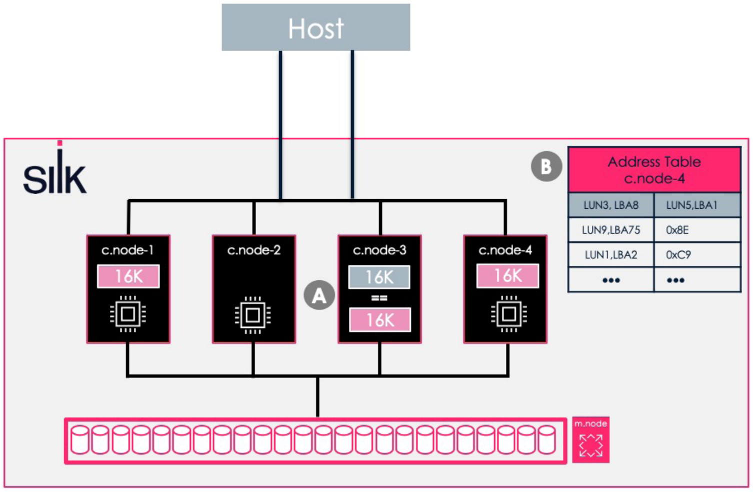
LBA Mapping Table	
LUN1, LBA2	c.node-1
LUN3, LBA8	c.node-4
•••	•••
LUN5, LBA1	c.node-3

2. On c.node-4, a hash value H1 is created for the 16KB block (A). The hash value is sent to the appropriate c.node, which is c.node-1 and a lookup is performed in the hash table. c.node-1 reports back that the hash value was queried with the address that appears as LUN5, LBA1 (B). Recall that the 16KB block that was written to LUN5, LBA1 was stored to the K-RAID by c.node-3.



Distributed Dedup DB	
H1	LUN5,LBA1
H2	NULL
H3	NULL
•••	•••

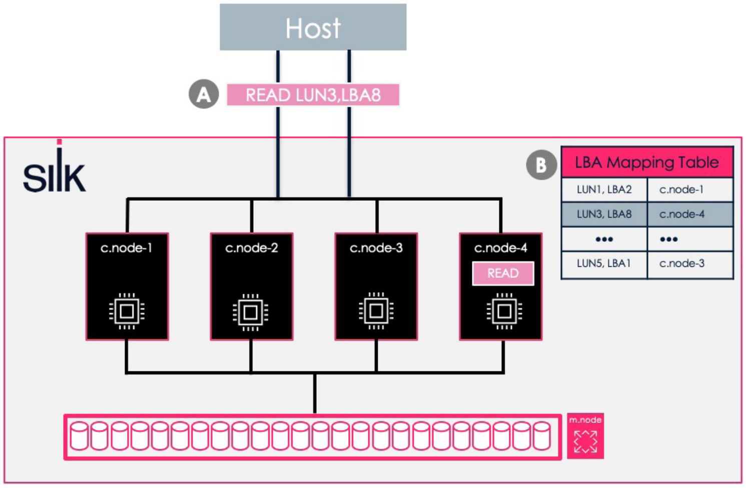
3. To avoid any probability of hash collision, it is necessary to compare the actual data. c.node-4 sends the 16KB block to c.node-3 for a full compare (A). This requires a single read operation by c.node-3 from the K-RAID, since it was stored as a single 16KB block. Once it checks out, c.node-4 updates its address table at LUN3, LBA8 to point to LUN5, LBA1 (B). These metadata updates are mirrored between c.nodes and eventually de-staged to the K-RAID, so metadata is kept highly available at all times. It is now possible for c.node-1 and c.node-4 to free the 16KB block from their DRAM.



Address Table c.node-4	
LUN3, LBA8	LUN5,LBA1
LUN9,LBA75	0x8E
LUN1,LBA2	0xC9
•••	•••

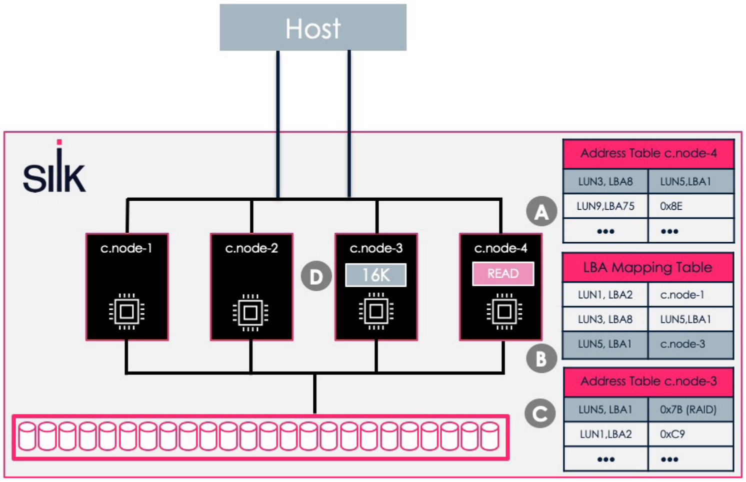
Read

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



LBA Mapping Table	
LUN1, LBA2	c.node-1
LUN3, LBA8	c.node-4
•••	•••
LUN5, LBA1	c.node-3

2. c.node-4 looks up LUN3,LBA8 in its Address table and finds reference to a logical address, LUN5,LBA1, rather than a physical address (A). Using the Mapping table (B), it requests the data from c.node-3, the owner of LUN5, LBA1. c.node-3 looks up the physical address (C), retrieves the data and decompresses it (D).

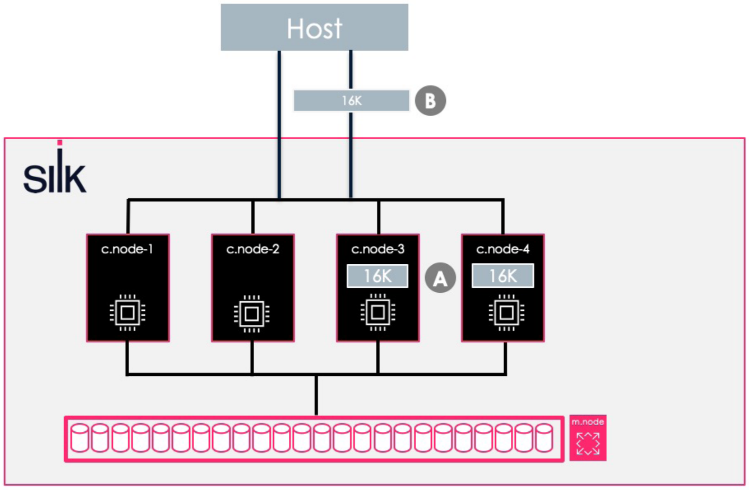


Address Table c.node-4	
LUN3, LBA8	LUN5,LBA1
LUN9,LBA75	0x8E
•••	•••

LBA Mapping Table	
LUN1, LBA2	c.node-1
LUN3, LBA8	LUN5,LBA1
LUN5, LBA1	c.node-3

Address Table c.node-3	
LUN5, LBA1	0x7B (RAID)
LUN1,LBA2	0xC9
•••	•••

3. c.node-3 sends the requested 16KB to c.node-4 (A). c.node-4 sends the requested 16KB back to the application's host (B).



Summary

Silk delivers a consistent and rich set of data services for public cloud infrastructure. These data services include global deduplication, inline compression, thin-provisioning, replication, high availability, snapshots, data mobility, predictive and preemptive self-healing, and more. Silk also provides a cloud agnostic solution which delivers the ability to easily move data across different cloud platforms. Silk's unique scale-up and scale-out technology disaggregates compute and capacity, enabling organizations to add or remove underlying cloud resources as needed to match business requirements.

About Silk

Silk is the leading platform to quickly move mission-critical data to the cloud and to keep it operating at performance standards on par with even the fastest on-prem environments. Silk works with global enterprise companies and cloud providers to ensure a seamless, efficient, and smooth migration process, followed by unparalleled performance speeds for all data and applications in the cloud.

The platform makes cloud environments run 10x faster and the entire application stack is more resilient to any infrastructure hiccups or malfunctions. Silk has offices in Israel and is headquartered in Needham, MA.

For more information, visit https://silk.us/.