KIOXIA

Introducing Cross-Domain Data Replication in KumoScale™ Software

Delivering Next-Generation NVMe-oF™ Data Resilience in the Data Center

The Problem: Keeping Data Safe and Available

In the modern data center, keeping data safe and available without sacrificing efficiency or performance is a multi-faceted problem:

- **Data Protection Requirements Vary by Application**
  Private cloud and service provider data centers are characterized by diversity. Many of the largest consumers of storage capacity in a data center context (e.g., web caches, analytic pipelines and machine-learning), do not require data protection, yet they run beside other applications where it is critical to protect data against loss and keep it available without interruption.

- **At-Scale Failure Domains can be Large**
  Traditional approaches to data protection focus on the failure of the storage element (i.e., SSD) itself. They typically use RAID or erasure coding within a storage subsystem that does not scale well - where a top-of-rack switch may serve dozens of client machines and a power-delivery subsystem may serve dozens of racks. The modern approach to data protection recognizes that everything fails eventually, so a critical measure of any redundancy scheme is whether it can span the 'blast radii' of these large scale infrastructure elements. Since data center hardware varies widely, the failure domain definition must be simple and general.

- **Performance Matters**
  The key performance metric for distributed storage is read latency. Bandwidth and input/output operations per second (IOPS) can be addressed via parallelism to an almost arbitrary extent, but latency cannot. The central conundrum of a cloud storage architecture is how to resolve the tension between low-latency server-attached flash storage and the safety and flexibility of networked storage.

  The NVMe™ over Fabrics (NVMe-oF) specification offers the promise of a 'best of both worlds' solution. However, seasoned systems architects know that it's perilously easy to squander this potential. The relationship between traffic load and queue latency is exponential for any system or subsystem, so a bottleneck anywhere between the application and storage medium can make for unacceptable delays. It is critical to avoid data protection designs that require multiple network hops between storage nodes for normal I/O operations.

- **IT Teams are a Critical, Constrained Resource**
  While efficient, performant and safe storage is of unquestioned value, the implementation of a cloud-scale storage infrastructure has historically required a significant investment of IT resources. For some data center operators, the appealing ROI of disaggregated flash storage as primary storage has remained tantalizingly out of reach, mostly because overloaded IT teams were not able to devote the resources necessary to design, deploy and operate it, and particularly, to react appropriately when failures occur. A key consideration in the selection of a networked storage system is how well it meshes with the administrative and monitoring infrastructure already in use - and how much time it requires from the operations staff.

Data Protection Alternatives:

To protect data and make it highly available in modern data centers requires an understanding of data protection schemes and choices:

Replication vs RAID vs Erasure Coding

All data protection schemes employ redundancy to guard against the failure or partitioning of storage media. Replication simply stores multiple copies of the data itself, while RAID and erasure coding store coded blocks which can be used to reconstruct missing data in the event of a failure. These coding systems can be characterized by two integers (k, n), and usually have the property that any 'k' out of 'n' code-words are sufficient to recover the initial payload. RAID and erasure coding systems have the appeal of greater storage efficiency. For example, if (k, n) = (6, 8), the code has a redundancy factor of 33%, and can survive the loss of any two of eight storage elements. However, coded systems also share important weaknesses relative to replication, as follows:

- **The Small Write Problem**
  To construct a coded 'stripe' for writing, a (k, n) coded data protection system begins with 'k' payload blocks, to which the 'n-k' code blocks are added of equal size. Modern storage systems typically suffer significant efficiency losses for I/O that is less than 4,000 bytes (4kB) in size, so chunk sizes are usually in 4kB multiples. Given the (k, n) code of (6, 8), and 4kB blocks, the storage stack would need to assemble six-blocks of user data into an eight-block (32kB) coded word, and then perform eight 4kB writes to eight storage devices.

  This form of data protection works very well for object stores, backup systems and other storage services optimized for streaming large objects, but for a general-purpose block storage system, it exhibits a fatal flaw. If a host performs a synchronous write of one or a few 4kB blocks, it could be an arbitrarily long time before enough blocks are present in the storage stack to construct a writeable code-word. In I/O performance tests, the results are anomalously long write latencies for very lightly loaded systems. In real world applications, systems can be hung up waiting for an I/O to complete, and may not return because it is stuck in the write queue waiting for a full code-word. This problem is particularly prevalent in 'b-tree' type databases, such as MySQL™, which performs all data writes in blocks of 16kB.

- **CPU Utilization**
  Another issue with coding schemes is that they require the storage stack to read each byte of data from memory into the CPU in order to calculate the code blocks, and subsequently write those blocks back-out to memory. While the computational portion of erasure coding has been highly optimized in recent years, this manipulation is equivalent to a data buffer copy. NVMe protocol-based I/O and recent improvements to the Linux® I/O stack are trying to reduce or eliminate buffer copies, as they exhaust CPU memory bandwidth and cache resources, and form a bottleneck in the I/O path. Having to code every write operation reintroduces the buffer copy in a way that can't readily be circumvented. Given the reasons and drawbacks of some data protection schemes, KumoScale software uses data replication as its core data protection mechanism.

Client-based vs Server-based Replication

The discussion to this point has presumed that data protection encoding was performed at the source of the write rather than in the storage 'back-end.' An alternative approach is to forward raw data to one of the storage nodes, and confine coding and forwarding of the data subsets to other nodes or failure-zones to the storage system (server-based replication). The main advantages of this approach is simplicity, as all of the storage details are left to the storage subsystem to handle, which shields the compute nodes from involvement in data protection. On the other hand, client-based replication can increase performance dramatically, supporting many more clients per storage node, which in turn, lowers storage costs per served-client substantially[1].

It is traditional to measure the cost-effectiveness of storage systems on the basis of dollars per byte of usable storage capacity. However, with the advent of the NVMe standard and cost-effective flash memory pricing, dollars per IOPS and read latency have become equally relevant measures, and in some cases, more relevant. At the same time, the extraordinary capacity and performance that NVMe SSDs deliver is driving a rapid move to disaggregate these SSDs from compute nodes and share them over the network. In this environment, it is important to understand factors that affect high-performance networked storage systems:

- **Identifying Network Bottlenecks**
  The typical network interface for a KumoScale storage node[2] is comprised of two, dual-ported 100 to 200 Gigabit Ethernet (GE) network interface cards (NICs). The nominal throughput of this configuration utilizes the PCIe® Gen3 interface, but is limited by each NIC's PCIe interface to about 22.7 gigabytes per second (GB/s) full-duplex (due to line coding and protocol overheads). In real-world deployments, such a storage node may serve volumes to hundreds or even thousands of VMs, so the aggregate bandwidth of clients will always far exceed that of the storage node. As such, maximizing storage efficiency is reduced to maximizing storage node network bandwidth utilization.

Data replication originating at the client, rather than at the storage node, can dramatically reduce the effective cost of storage (based on dollars per GB/s and dollars per IOPS). If replication is implemented completely on the target-side, data written to storage node 'A' must then be forwarded to storage nodes 'B' and 'C', thus traversing node 'A's' network interface three times. This consumes I/O capacity at the network interface (which is already the system performance bottleneck). Data replication originating at the client eliminates this problem.

- **Dynamic Mapping**
  There are many schemes to mitigate the small write problem but the most popular methods abandon the notion of a fixed location on the storage medium that corresponds to a particular logical block address at the client. Instead, dynamic remapping systems place incoming data intelligently irrespective of its logical address. This form of dynamic mapping requires the storage system to maintain a table listing for each logical address, as well as where the corresponding data was most recently written.

The KumoScale Cross-Domain Data Replication Solution

KumoScale software from KIOXIA (formerly Toshiba Memory) protects user data with a technique called Cross-Domain Data Replication (CDDR). This capability is based on data replication and enables users to flexibly specify the number of replicas on a per-storage-class basis (per application). In CDDR, write operations are processed synchronously in parallel (forwarded to each replica), and then acknowledged back to the application only once it has been committed to non-volatile media at each replica.

CDDR Functional Description

CDDR takes a data center-level approach to resilience against drive failure or network partitions as KumoScale software users can specify arbitrary failure domains by tagging storage nodes, compute nodes and other resources. It can then create volume replicas across multiple KumoScale appliances located in different failure domains. The solution also addresses both equipment failures and maintenance downtime scenarios, and can be applied to only those applications that require it through storage class specifications. New 'cloud-native applications' which handle data resiliency at the application level may not require storage-layer resiliency. The ability to specify the type of resiliency as a storage class parameter represents a significant savings potential relative to a conventional array (which implements data protection locally), and in a uniform way for all applications.

The CDDR solution utilizes the Linux 'md' module located in host servers for replica synchronization, leveraging its maturity and stability and inheriting its feature-set that includes rich monitoring and alerting capabilities. The replication-layer is configured and monitored by an agent on the host server. In a Kubernetes® containerized framework, this agent is the CSI driver. In a bare-metal environment, this agent is a job that runs on the host.

Upon replica failure, the Linux 'md' module immediately starts logging data updates and generates an alert on the degraded state of the volume. When the replica reappears, KumoScale software automatically reconnects it and is synchronized via the 'md' module with the latest changes. Upon permanent replica disconnect, CDDR provides the means to reconstruct the replica in a different location, adhering with topology constraints. Once the volume is reconstructed, it will remove the configuration of the missing replica.

Volume allocation and placement, replica configuration and volume connectivity over the network fabric are all managed by the CDDR solution. Notifications and alerts are sent to a preconfigured application or email address (for bare-metal or VM environments) and to syslog (for containerized environments).

Benefits of CDDR

KumoScale software's approach to data replication offers several benefits including selective protection, topology awareness and performance (bandwidth and latency).

- **Selective Protection**
  Applications that are designed for scale-out, cloud-native environments often include their own form of data protection. This includes key-value type databases such as MongoDB®, Cassandra® and Aerospike®. For these applications, implementing redundancy at the storage layer would exhaust resources without materially increasing resilience. On the other hand, since most legacy applications are designed on the presumption that storage is reliable, the storage layer must supply resilience and availability. KumoScale software meets these requirements by enabling resilience-level and placement policies to be specified independently for each storage class (for each application.)

- **Topology Awareness**
  In a large-scale data center, the physical infrastructure is divided into well-defined 'failure domains.' Depending on the scale and design of the power delivery, cooling, networking and other systems, a failure domain might be a rack, a row of racks, or even an entire data center. Small-scale, resilient solutions that fail to span the largest failure zone can provide a false sense of security. KumoScale software makes it simple to identify failure zone topologies to its Provisioner Service by means of labels. Creating policies referring to labels ensure that volume replicas are mapped in separate failure domains.

- **Performance (Bandwidth and Latency)**
  KumoScale software implements replication via the Linux 'md' layer in the client. All setup, monitoring and repair of failed replicas is handled by the KumoScale Container Storage Interface (CSI) driver for containerized deployments, and by the client agent for bare-metal or virtual machine (VM) environments. Performing replication from the client takes advantage of the enormous aggregate network bandwidth of data center compute nodes and avoids creating unnecessary bottlenecks at the storage node. The result is a 100% to 200% improvement in IOPS per dollar and bandwidth per dollar versus target-side replication[3].

Key Components of CDDR

The KumoScale CDDR solution is comprised of several components and follows a sequence of events which take place when a resilient volume is provisioned (Figure 1). In this example, Kubernetes container orchestration is used:

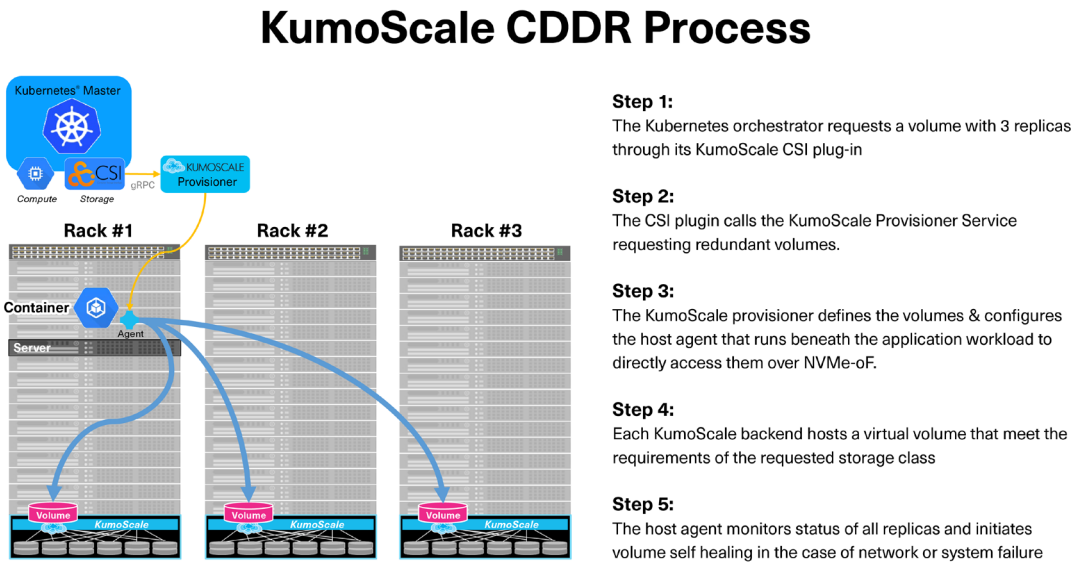

Figure 1: The CDDR process in KumoScale software

- **Data Center Provisioning / Orchestration Infrastructure**
  Data center provisioning or container orchestration request a resilient volume for an application host that includes the storage class specification detailing the desired service class and the required number of replicas.

- **CSI Plug-in**
  In a Kubernetes containerized environment, the KumoScale CSI Plug-in forms part of the Kubernetes Master. It forwards volume requests to the KumoScale Provisioner Service, associates the resulting volume replicas to the relevant host as namespaces, and initiates the NVMe-oF connection from the host to the replicas. The plug-in also creates a file system and a mount point over the NVMe device. In a bare-metal environment, these functions would be performed by a provisioning automation tool, such as Ansible™ playbooks.

- **KumoScale Provisioner Service**
  The provisioner service is responsible for mapping each volume replica to a specific storage node and SSD, based on the topology parameters of the storage class specification and the KumoScale inventory. It then provisions the storage volumes by issuing REST commands to the relevant KumoScale storage nodes. It also sets the volume handle in the application server plug-in.

- **KumoScale Storage Nodes**
  The storage nodes respond to KumoScale Provisioner Service commands and maps the volume to physical drives. They also process I/O commands and enforce access rights.

- **Client (or Host)**
  The client (or host) connects to the newly created namespaces via the NVMe-oF standard. The replication layer (via the Linux 'md' layer in the client) implements the data-path portion of the solution, replicating write commands to multiple KumoScale appliances, and distributing the read load among them. An agent (for bare-metal or VM environments) or the CSI driver (for containerized environments) monitor the state of the replication layer and generate events to the syslog.

- **Logging**
  The resilient configuration state is logged in the data center's syslog. Notifications and alerts are sent to the storage administrator.

Functional Highlights of CDDR

The functional highlights and key capabilities of KumoScale software include event log and alerting, failure recovery, self-healing, planned downtime and application programming interface (API) integration:

- **Event Log and Alerting**
  The resilient configuration is continuously monitored. Once a trigger event is identified, a message is sent to the data center log. Alerts can be customized per the data center administrator's requirements. For example, an email may be generated upon certain trigger events or an API call with information about the nature of the trigger event may be initiated to an alert system.

- **Failure Recovery**
  When a connection to a replica is lost for any reason, the replication layer begins to maintain a log of all write operations to the volume. Upon reconnection of the missing replica, a resync process is initiated, which brings the stale replica up-to-date using data from the remaining replica(s). This process can be throttled to control the impact on network bandwidth and storage performance. Once the replica is fully synchronized, it is brought back into service. As long as at least one replica remains functional, the application will not be aware of the failure.

- **Self-Healing**
  If a missing replica is not returned to service within a programmable timeout period, the volume will be declared permanently out of service. This triggers the provisioning of a new, empty volume on an available storage node. The new volume will observe all specifications of the storage class as initially created, including failure zone topology constraints. The new volume is connected as a replica, and synchronized using data from the healthy replicas. Should the failed replica later reappear, any data it may contain is wiped clean and the logical entity is destroyed. The healing process may be triggered by the storage administrator once a failure is detected or preconfigured by a script to run automatically in the event of a failure.

- **Planned Downtime**
  Data center administrators may need to schedule a storage node to be taken offline for upgrades or maintenance. In these instances, updating the resilient configuration ahead of time to avoid 'surprise' replica loss is recommended. The CDDR solution supports this scenario by maintaining the resilient configuration, eliminating a degraded state.

  The storage administrator initiates a process which adds a new replica to the configuration in advance of the downtime. The configuration is then at a state of 'N+1' replicas, where 'N' is the steady-state configuration. When one of the racks is taken down, the configuration returns to 'N' replicas and the volume does not enter a degraded state.

- **API Integration**
  As with all aspects of the KumoScale software control plane, the CDDR API is designed to fully integrate with existing, deployed infrastructures in the data center, as follows:

  - *Supports containerized environments and Kubernetes frameworks via the CSI driver*
  - *Supports bare-metal installations using Ansible playbooks and modules*
  - *Supports customer logging mechanisms for alerts and event logging*
  - *Supports telemetry systems (such as Prometheus™ and Graphite™) for statistics*
  - *Exposes a RESTful API for simple integration into any provisioning system*

Summary

The CDDR solution is designed for the modern data center infrastructure. It leverages the multiple KumoScale software appliances in a topology-aware environment and maps volume replicas across these appliances to ensure that the storage will remain resilient to a failure (e.g., network, power rack, SSD or KumoScale software). It is built to interface with existing data center systems and common orchestration frameworks.

Additional information is available at https://kumoscale.kioxia.com/.

NOTES:

[1] Based on 2x replication that requires two writes to storage for every write from the client. If the second write is forwarded from the first target, the data must first enter the target and then depart again, consuming 2x the bandwidth for a simple write that could adversely affect storage costs.

[2] As of the publication of this tech brief – August 2020.

[3] Based on 2x replication that requires two writes to storage for every write from the client. If the second write is forwarded from the first target, the data must first enter the target and then depart again, consuming 2x the bandwidth for a simple write. In comparison to target-side replication, client-side replication could result in improvements in IOPS per dollar and bandwidth per dollar.