STRONGLINK

Product Description

Version 9.0 | June 2017

STRONGBOX DATA SOLUTIONS

Contents

Who is StrongBox Data Solutions?

StrongBox Data Solutions (SBDS) is trusted by many of the world's largest data environments to simplify both data and storage management with solutions that help customers dramatically reduce storage costs, increase user productivity, and reduce IT management complexity.

What we do

Our goal is to make data storage simple, flexible and affordable, and we pride ourselves on offering customer support that is unmatched in the industry.

How we do it

1. We engineer solutions with the end user in mind, so our products are easy to use while also easy to manage, making even the most complex IT operations simple and automated.
2. We believe all organizations deserve powerful, secure storage solutions—not just those with big budgets.
3. We believe that you should be able to choose the right products to meet your business requirements. We designed our solutions to deliver value, to be reliable, to enhance the resources you already own, and to be scalable in order to support seamless technology upgrades for an IT investment that is truly future-ready.
4. We build on industry standards so there is no vendor lock-in. Our customers choose SBDS because we offer the best solutions, not because they're locked into proprietary software or architectures.

What is StrongLink?

StrongLink® is a software solution that leverages the power of artificial intelligence to give you complete control of your data AND your storage, for any content, any filesystem, anywhere.

StrongLink presents all your storage resources in a global namespace, and constantly optimizes data placement across them by using metadata-driven analytics, user-defined business policies, and automated optimization algorithms. StrongLink can present any data across all storage types via any standard file protocol, even if the underlying storage doesn't support that protocol.

Why did we build StrongLink?

Approximately 90% of the world's data was created within the last two years[1]. Of the massive amount of data produced annually, a stunning 80%[2] will be archived for compliance or "just in case we need it" reasons, but will never be read again. The remaining data may, at best, be accessed and re-used only once per year.

Many organizations:

- Are overwhelmed by increasing volumes of data.
- Have legacy, incompatible data silos resulting in stranded data or complex management.
- Need to ensure accountability and data integrity across file systems.
- Need an effective bit-rot audit/recovery methodology.
- Are struggling with knowing exactly what data is being stored, ensuring data provenance, retention and periodic purges for compliance purposes.
- Are trying to avoid NAS/SAN sprawl.
- Need to control CapEx and OpEx while maximizing existing infrastructure.
- Need to distribute data to multiple locations for collaboration and/or disaster recovery (DR).
- Are trying to use the cloud effectively, both for processing and data storage, but are worried about their data being held hostage and/or data security breaches.
- Are attempting to optimize workflows across resources.
- Are looking for a way to enable cross-platform data lifecycle management.

[1] Source: Active Archive Alliance

[2] Source: IDC Digital Universe Study, sponsored by EMC

What does StrongLink do?

StrongLink can:

- Aggregate your existing data silos into a single global namespace
- Export the global namespace (or a subset thereof) using many standard protocols, including NFS, CIFS/SMB, HTTP(S), sFTP, S3, CEPH, and others.
- Manage and export the same global namespace across multiple sites.
- Manage data access with your exising authentication system, including Access Control Lists that have granularity down to a single digital asset and user.
- Maintain a complete and immutable audit trail of who did what to any data item in the system. The audit system can not be altered or deleted, even with system admin priviledges.
- Transparently and automatically maintain multiple instances of files based upon file retention policies, for DR, collaboration, or other needs. All files are verified with multiple check-sums for consistency across all instances as files change.
- Automatically extract file type-specific metadata as data is ingested into the system.
- Enables users to define custom metadata attributes and manually or automatically apply them during or after the data ingest process.
- Maintain incremental versions of every file/filesystem and associated metadata, optimized to reduce storage impact. This enables users to restore files or entire file systems to a previous point-in-time.
- Support cross-platform tiering policies to automatically keep data on the appropriate storage types or locations, based on metadata tags, or usage needs.
- Transparently integrate cloud storage, including configurable automatic encryption and periodic auditing of data to ensure long-term data integrity.
- Replicate metadata and managed data assets across all your sites, based on configurable policies
- Facilitate business demand-driven elastic expansions of your data management capacity through a transient cloud infrastructure or the temporary deployment of VMs on your internal infrastructure
- Leverage its self-healing, no single point of failure design so there is no downtime even when there is a node failure in a StrongLink Constellation (cluster).

How does StrongLink work?

StrongLink is a modern, no master, distributed server application that is designed to fully leverage today's multi-core CPUs, virtual environments and cloud infrastructure. It was created using time-tested and well-supported Open Source technologies to build the core data and metadata engines, management services layer. We deliberately avoided proprietary protocols and technologies wherever possible to ensure your data/metadata is never held hostage.

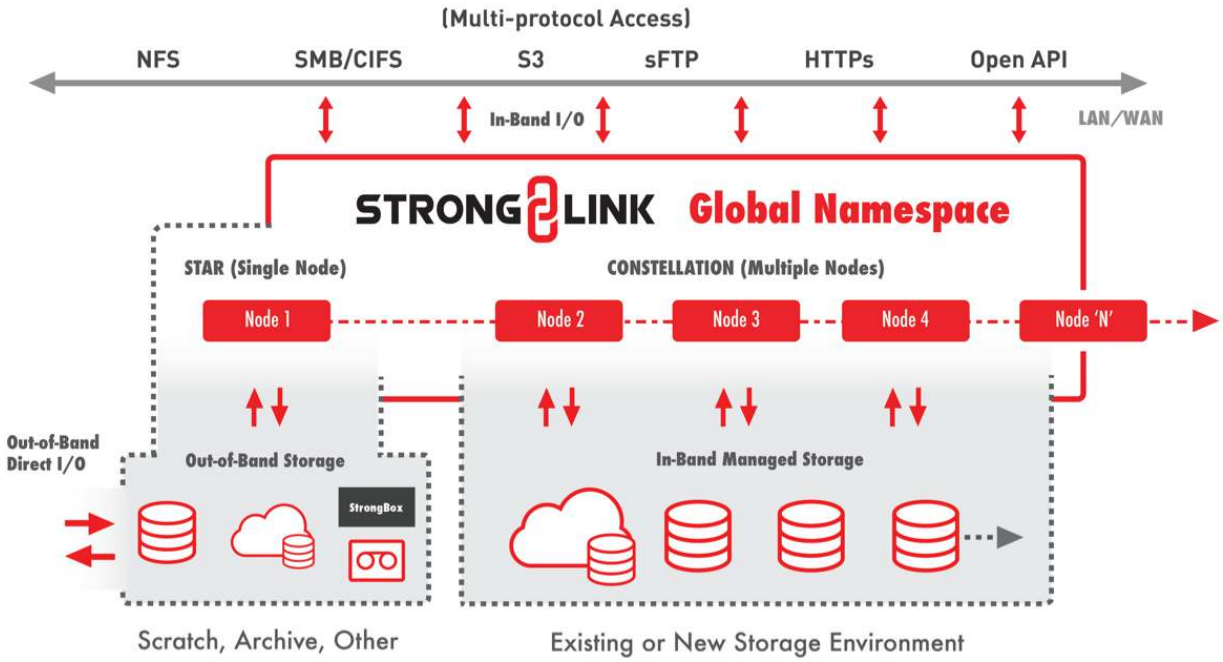

Figure 1

Figure 1 shows a logical view of a StrongLink deployment, including both in-band and out-of-band choices. The primary takeaway from this diagram is that StrongLink is designed to leverage—not replace—your existing infrastructure investment. This is a logical consequence of the fact that many organizations have large data sets, and "fork lift upgrades" just aren't practical anymore.

StrongLink is deployed in one of three deployment options, including out-of-band, in-band, and hybrid, as described in the next sections.

1) Out-of-Band Model

In this model, StrongLink is deployed alongside your existing storage (see Figure 2), and is not in the data path. Your client systems continue to mount and access your storage directly as before, with your storage seeing StrongLink as simply another client system. This model is most appropriate when StrongLink is used as a smart archival system, a cloud gateway, or as an offload from scratch storage, for example.

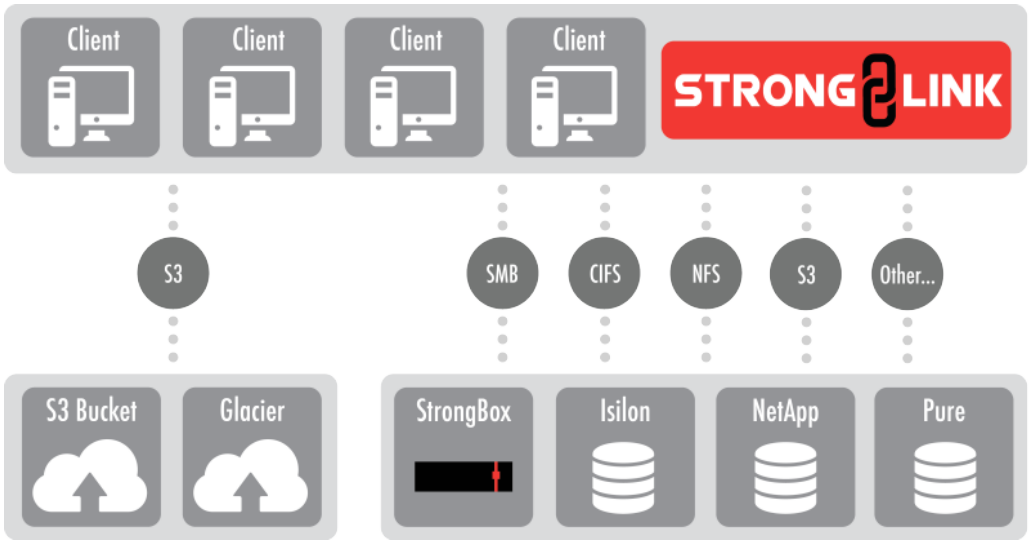

Figure 2 – Out-of-Band Configuration

Deployment is as simple as using the StrongLink web GUI to:

- Configure StrongLink to access your network
- Set up appropriate authentication to access the storage
  - User with full read access privileges on the storage; or
  - Integration into existing LDAP/AD domain
- Set up 1st level global namespace points where your existing file structures will be harvested
- Tell StrongLink about the shares exported from those of your existing storage systems that need to be actively archived, or determine other post-ingest actions that might be needed for the files.
  - This could be anything, from automating the addition of custom metadata, initiating copies to archive, cloud or DR, etc.
- Initiate the process.

Once the metadata harvest is initiated, StrongLink will walk all the mounted stores and harvest all available from all the files and filesystem hierarchy. This can be done with data in place. The harvesting process consists of extracting metadata from known file types, storing the metadata in the StrongLink data engines, generating dual checksums of the file, and initiating any policies required on ingest, such as adding custom metadata, managing downstream actions such as making copies to archive or DR, or triggering workflow automation based on other use cases as needed.

The primary upside of the out-of-band model is that it's very non-intrusive. End users have absolutely nothing to do; they just keep using the remote mounted storage as usual.

The primary downside is that since the storage can be accessed directly by users in the out-of-band model, it means StrongLink cannot guarantee data provenance, or maintain and audit trail of all file actions, such as is possible in the in-band model below.

StrongLink can be configured to delete the original copy on your existing storage once any archival copies have been verified. StrongBox Data Solutions strongly recommends that this feature not be enabled unless multiple archive copies are created and verified.

StrongLink will continue to monitor the out-of-band storage for new files, or for updates to previously archived files. As these changes occur, the new files or updated versions will be harvested as previously described. When a file is updated, unless versioning has been specifically disabled, this results in a new version of the metadata record in the database and a new copy of the file in the archive. Although the StrongLink versioning system is optimized to minimize the storage footprint, the system can be configured to limit the number of versions that are maintained to further optimize storage utilization.

As a final note, StrongLink does not assume an active management role *for your existing storage* in the out-of-band model. Meaning, it does not assume responsibility for automatically performing periodic purging on your existing storage to ensure there is always some open space available, nor does it do backups of your existing storage in the traditional sense. This model is primarily intended for minimally intrusive active archive systems.

2) In-Band Model

In the In-Band Model, StrongLink is deployed between your client machines and your existing (and/or new) storage, as shown in Figure 3. Clients no longer mount your existing storage directly, but instead mount exports in StrongLink of the entire global namespace, or selected portions thereof. From a user perspective, if they are used to seeing an SMB mount point today, they will see that same mount point once StrongLink has been initiated. But they can also view the entire global namespace across all storage based upon their permissions, via any protocol.

Thus, any filesystem or storage type can be presented to users and applications via any protocol, regardless of whether the underlying storage supports that protocol.

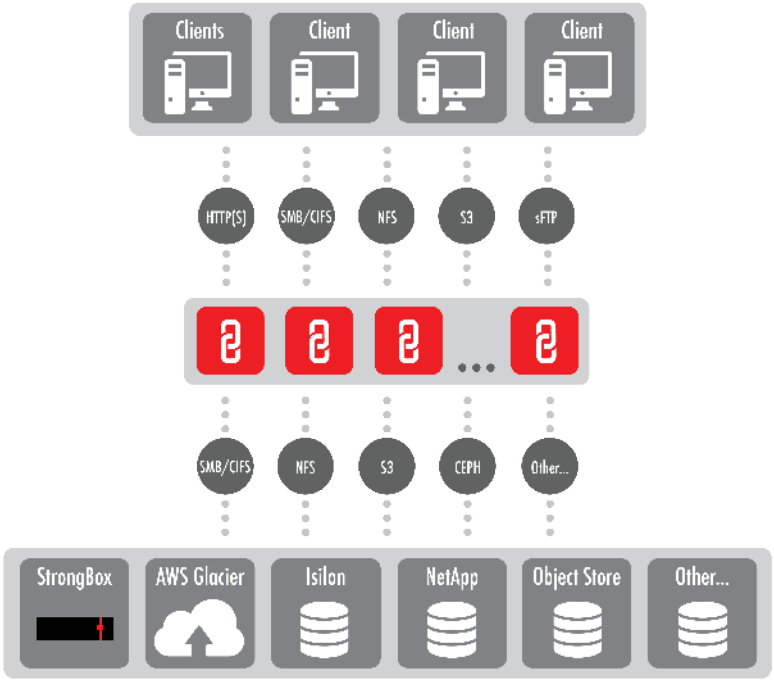

Figure 3 – In-Band Configuration

With the in-band model, a transition period is required to ingest data from your existing storage and migrate it to storage that is actively managed by StrongLink. The ingest is essentially the same as the initial metadata harvesting that was discussed in the out-of-band model, but with one significant difference. By definition, the out-of-band model deployment does not include any managed storage behind the StrongLink system, whereas the in-band model does.

Thus in the in-band deployment model, copies of your data can be made on one or more stores managed by StrongLink. These stores could be any storage type (SSD, spin-down or traditional rotating storage, cloud, or tape). The stores can be configured as a tiered structure that allows active placement of data, based on cost and access requirements. Additionally, individual stores can be aggregated into logical pools, even though they may be from different vendors or file systems. These are called Smart Pools, and may be used to load balance across otherwise incompatible storage types, to increase I/O performance and the effective size of the pool, and other uses. Inclusion of tape-based stores allows this model to perform both traditional backup (short-term retention) and archival (long-term retention) functions as well. StrongLink's ability to transparently manage multiple copies of your data allows it to perform periodic data integrity audits, either on a spot check or comprehensive basis, depending on your individual requirements. If the audit reveals damaged copies, these can be restored from good copies. Of course, since not all data is critical enough to warrant this level of protection and there is an associated storage space cost, this functionality is also fully configurable on a per-store and per-file type basis.

The deployment process is only slightly more complex than the one used for the out-of-band Model. Once the previously described first four steps are completed, the web GUI is used to:

- Create desired stores
- Create desired Smart Pools
- Select copy creation policies and criteria (this is how tiering is implemented)

The last step is determining how the data is to be ingested, i.e. by either using the "pull model" like the out-of-band configuration does, or by "pushing" the data into StrongLink using standard tools like *robocopy* or *rsync*. Another way to "push" data into StrongLink is to employ drag 'n drop GUI tools like *Windows Explorer*, *WinSCP*, *FileZilla*, *Finder*, a Linux tool like *Midnight Commander*, or the file manager GUI, bundled into your Linux desktop.

StrongLink also provides a native client for managing your entire storage environment:

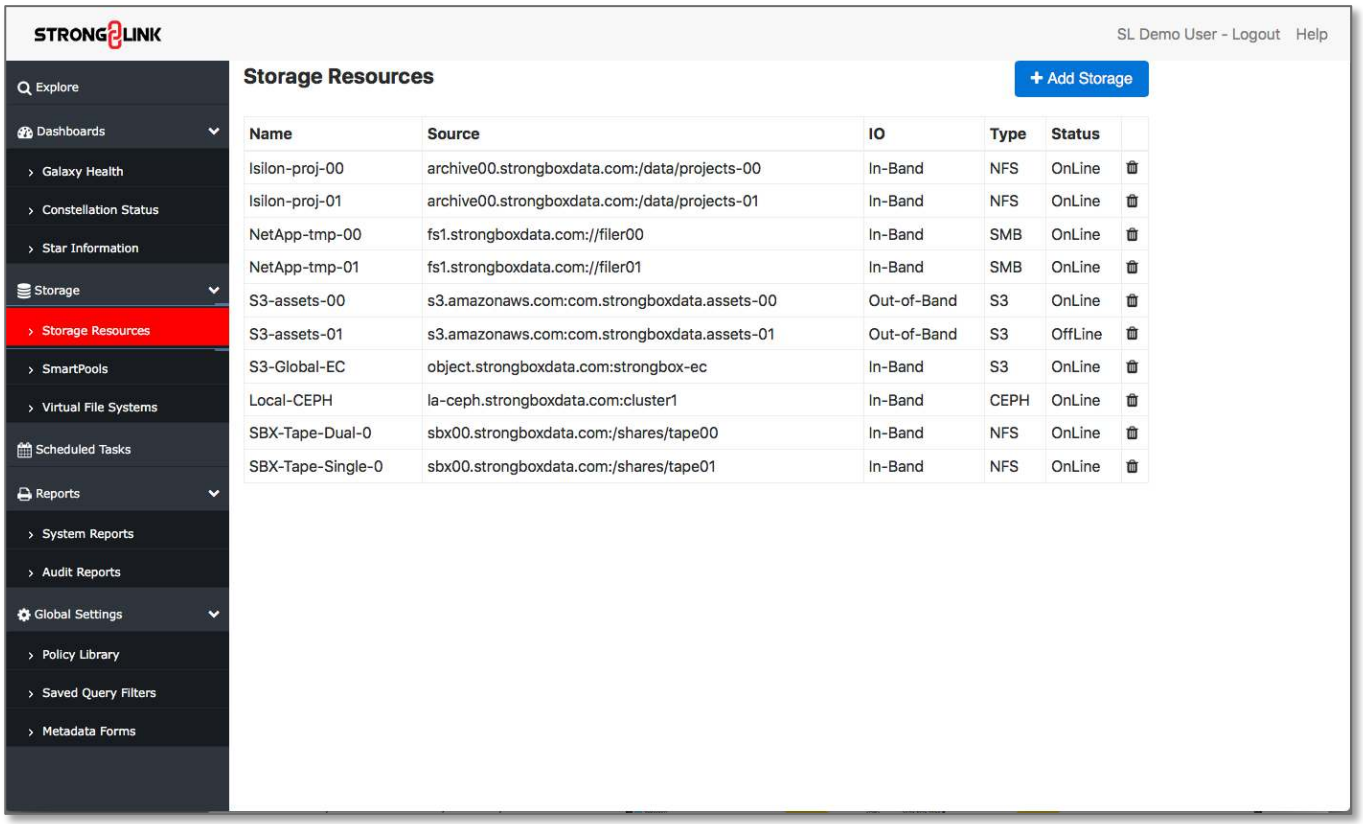

Figure 4 – Storage resources pane within the StrongLink Control Panel,

3) Hybrid Model (Both In-Band and Out-of-Band)

As previously mentioned, unless a completely new site is being deployed a transition period is required when setting up a StrongLink system in an in-band configuration, which is where the Hybrid model comes into play. The first step will be to provision some storage (new storage as part of a modernization project or a portion of the existing storage) behind the StrongLink system. This storage, which we call “managed storage”, is set up as one or more stores and/or Smart Pools. These stores and/or pools become a target for data that is ingested from the existing unmanaged storage. If there is a tape (or cloud) archival tier in the configured StrongLink hierarchy, copies will be made to that tier as the data is ingested, based on the policies that have been established.

As data on the managed storage is verified, the original copies on the unmanaged (existing) storage can be removed. This allows old existing storage to be retired or moved behind the StrongLink system to become additional managed storage. As the managed stores are verified and come online, they are exported to your clients as part of the global namespace. Usually, all or most of the unmanaged storage is eventually converted to managed stores, or retired so the system reverts to the in-band model. However, it’s possible to maintain some out-of-band unmanaged storage for latency sensitive scratch storage, or for archival purposes, or other uses if that configuration better suits your workflow requirements.

This same hybrid methodology allows transparent upgrades to your physical storage as new technologies come online. You simply stand up the new storage platform, set it up as one or more new stores behind StrongLink, start making copies from your older technology managed storage, during which StrongLink verifies that the new copies are valid. Once the new copies are validated, the metadata database can be bulk updated to remove references to copies on the old storage. Once that has been done, the old storage is then shut down. The entire process occurs automatically, behind the scenes, while StrongLink is serving your data out to your client systems.

StrongLink Architecture

Prior to diving into StrongLink's architecture, we should define a few terms.

- **Collection**: a special type of resource that contains references to other resources, much like a file system directory has references to the files it contains. A collection may contain other collections as well as normal resources.
- **Component**: a program that embodies a constrained subset of the overall functional capability for the StrongLink system.
- **Constellation**: a tightly coupled group of nodes that runs the StrongLink software, similar to a cluster.
- **Datastore**: also called a store. A datastore is a physical system that contains one or more User Data Items (UDIs). Examples include a local file system store, a CEPH object store, an S3 bucket or a Glacier object store.
- **Galaxy**: a loosely coupled group of constellations. Generally, there is one Constellation per site. The Galaxy groups multiple sites together into a single global namespace. Galaxies are usually deployed to implement Disaster Recovery strategies, for tying together remote offices, or to improve Quality of Service when accessing the same data at multiple locations.
- **Member**: a resource that belongs to one or more collections. A member is considered a child of any containing collections; members of the same collection are siblings. Finally, a collection is the parent of its members.
- **Namespace**: a symbol used to group a set of like items. Namespaces are hierarchical just like a traditional file system. StrongLink namespaces use the "/" separator just like a Unix directory path.
- **Node**: a single physical machine or VM instance, referred to as a Star in StrongLink. Generally, a node will have multiple CPUs and/or cores. In most StrongLink deployments, nodes (Stars) will be part of a Constellation and possibly a Galaxy so each node may be configured differently, both in terms of software and hardware, to achieve the overall system objectives. Nodes are most commonly configured as database engine nodes, I/O nodes or workflow execution nodes.
- **Resource**: a metadata record that may or may not have an associated UDI. The resource is versioned (unless specifically disabled) and can either be populated when an UDI is ingested into the system or created independently from the UDI ingest. Additional metadata fragments can be added at any time, either programmatically or manually. Any portion of the record can be modified or removed by users with the appropriate access.

- **Smart Pool**: a group of datastores connected by a policy that controls how UDIs (file/object) are distributed among the member stores.
- **Star:** a single node implementation of StrongLink. Able to expand to a Constellation.
- **Tag**: a generic key-value pair that can be attached to a resource, store, pool, namespace or UDI.
- **UDI**: a "user data item". This is the StrongLink designation for an original user data asset. Typically, this will be a file in conventional file systems, or an object in an object store.

StrongLink is a multi-component, no-master server application. The components communicate over an internal mesh network topology. All inter-component communication payloads are encrypted; since the messages are quite small, the encryption does not adversely impact system performance.

Breaking the application into several components provides several benefits:

- An application-level self-healing capability that is far superior to O/S-based HA solutions.
- Efficient use of multi-core processors.
- Isolation of related functionality to a single component of manageable complexity.
- Inherent availability of all functions within the system for configuration on all nodes in the system.
- Extreme flexibility when configuring software running on the various nodes in the system, which supports tuning based on the physical locality and connectivity of existing and new storage, as well as matching node hardware to the specific component mix run on the node.
- Reliable automatic live updates with minimal impact to system operations.
- 24x7 availability, with no need for backup windows.

StrongLink includes three database engines to store unstructured and highly structured data types which are used to run the system, as well as for analytics and machine learning. The engines support sharding for scalability and replication for reliability.

The unified global namespace is the key to the system. Think of it as a virtual file system. Much like a Unix file system, datastores can be mounted at various points under the root of the global namespace so that all of the resources on that store are now part of the global namespace. The global namespace is exported as a file system to the host operating system. From there, it can easily be exported to client systems via standard protocols such as NFS, SMC/CIFS, S3, etc..

The StrongLink store manager also supports aggregating datastores into Smart Pools using various configurable policies to distribute new UDIs among the stores that form the pools. Stores can be dynamically added to or removed from a pool at any time. They can also be temporarily taken offline for maintenance.

StrongLink uses trigger events, an internal script engine, job queues, and a periodic job manager to enable workflow automation. The most common use cases for this are automatic UDI copy generation, and automatic datastore purging of UDI copies to free up space for new data.

The first case is pretty straightforward. The replication policy is configured to generate N copies to N destination stores, based on your selection criteria. It could be as simple as all UDIs created in datastore “foo” are copied to stores “bar” and “Charlie”. Or, you can be more selective and only copy MP3 files created by user “Joe” on store “mix” to stores “production” and “archive”.

The store purge case is also easy to understand. Simply configure the store with a high water mark and a low water mark. These would typically be specified as a percentage of datastore total space, but where total capacity isn’t fixed (e.g. a store backed by a tape library), the high- and low-water marks would be specified as an absolute capacity (e.g. 10 TB). The store manager watches the datastore utilization and when the high-water mark is exceeded, a purge operation is put onto the system management queue. When this job executes, it will query the system to find UDIs that can be deleted based upon policies, which then reduce the datastore space utilization. The discovered UDIs will be deleted on a best effort basis until the datastore reaches the low-water mark.

The default query policy for this use case is a simple ‘least recently used’ (LRU) query, but you can configure additional constraints such as ‘the file must have a copy on datastore “Archive”’, for example. Keep in mind that reaching the low water mark may not be possible when adding additional constraints beyond the simple LRU policy.

Data and Metadata

As noted in the previous section, metadata is referred to as a 'resource' and user data (usually in files or objects) as a 'UDI'. As a cardinal rule, StrongLink *never* modifies your data. However, we do access it, usually during the ingest process, for these purposes:

- **Chunking**: sometimes your original data is broken into chunks for various reasons:
  - Maximizing performance on the underlying managed storage
  - Overcoming file size limitations on the underlying managed storage
  - Improving recall performance when accessing small segments of large UDIs; e.g. recalling a 10-second clip out of a 3-hour video file
- **Checksum generation**: unless specifically disabled, StrongLink generates at least two checksum types when your data is ingested. If the UDI is chunked on ingest, StrongLink generates checksums for the entire item and each chunk. This allows StrongLink to efficiently repair damaged copies from valid ones by only replacing damaged chunks.
- **Encryption**: encryption to internal storage is optional. All data copied to the cloud is encrypted as it is uploaded. In addition, all communication between StrongLink nodes is encrypted.
- **Proxy generation**: unless specifically disabled at ingest, for image UDIs that exceed a minimum set of dimensions, a small proxy image will be generated for rapid display when resources are browsed. This may be extended to add multiple proxies of different resolutions.
- **Data type detection**: StrongLink can generally deduce the UDI data type from a file extension when the source data is on file-based storage. For object-based source storage and for unrecognized file extensions, the system looks at the initial portion of the data to determine its type and uses standard media types (MIME types), as defined by IANA (www.iana.org). This is very similar to what the Unix *file* command does.
- **Metadata extraction**: Once the UDI data type is determined, StrongLink will apply a format-specific analyzer during the ingest process to extract relevant metadata. For example, image formats are often tagged with EXIF metadata. StrongLink will extract this metadata automatically and populate the resource record with it.

Querying

A Resource is a metadata record in the StrongLink database. Specifically, the record contains one or more previously defined metadata fragments. These fragments are defined using JSON Schema, a well-documented and widely supported IETF RFC. Using JSON Schema allows StrongLink to validate incoming metadata prior to updating the database record. StrongLink comes with several predefined metadata fragments, and you can define your own as well. StrongLink stores the metadata in JSON format, which is a huge benefit as JSON is so widely known and accepted in the Web Application space. StrongLink uses JSON with a few application-specific keywords added to minimize the query system learning curve.

So, what's the big deal about querying anyway? Put simply, many organizations only have a fuzzy idea of exactly what data they have, and their retained data sets have become way too large to understand without some sort of data management system. By aggregating all the metadata sources StrongLink becomes 'data aware' as part of an automated data classification step upon ingest. In this way, users are provided a complete picture of data they are storing, whether there are multiple copies, how often it has been accessed in the past 'N' weeks, and so on.

Then there's another critically important piece to the puzzle: when you have billions of data items under management, it's important that the query system provide search results in a timely fashion, even for complex searches. StrongLink has been carefully crafted to begin returning result sets in seconds, even when the result set contains millions of resources.

Versioning

Software developers are familiar with Source Code Management Systems that retain multiple versions of a program's source code. Storage system administrators can do this through snapshots which allow them to restore a previous version of a file per a user request. However, to do so globally across a heterogeneous storage environment in a way that doesn't require IT support is generally not possible in conventional storage-centric approaches.

For StrongLink, this is core functionality. StrongLink maintains multiple versions of both resources (metadata) and UDIs (file/object). Other than available disk space, there is no limit to the number of versions that can be retained; pruning is available to retain specific versions or perhaps the last 'N' versions of a particular resource. The resource is versioned semi-independently from the UDI. Any modification to metadata will always result in a new version of the resource but not a new version of the associated UDI.

So, if you do an initial ingest of a file, then the result would be version R1 of the resource and version U1 of the UDI; R1 would have a pointer to U1. Now, if you do 4 different updates to metadata but make no updates to the UDI, then the result would be versions R2, R3, R4 and R5 of the resource, all of which point to version U1 of the UDI. However, if you ingest an updated copy of the UDI, then the result would be version R6 of the resource, which points to version U2 of the UDI. Versions R1 through R5 of the resource would continue to point to version U1 of the UDI. This per resource versioning system completely obviates the need for the traditional snapshotting supported by NAS system vendors so long as UDI versions are properly archived prior to pruning. The reason is that the resource record (which is quite small by comparison) is never deleted, except in the case of a hard destroy command.

When a resource is pruned, the pruned versions of the resource have a flag set so they don't show up in normal queries. Moreover, any UDIs that are only associated with pruned resources are deleted (except from archive stores) as this can recover significant storage space. Because protection of your data is of paramount concern to us, a 'force' flag must be set to prune UDIs that don't have copies on an archive store; if a flag is not set, a list of resource IDs that were not pruned will be returned.

Data Integrity

Data integrity includes several aspects:

1) the integrity of the StrongLink database;

2) the integrity of the data (the UDIs) you store via StrongLink[3]; and

3) the integrity of the datastores themselves.

StrongLink databases are automatically backed up at least once per day. In small systems with no replica sets, the databases have to be locked for writing during the backup period. Larger systems or smaller ones with high availability requirements will be configured with replica sets, which may be taken offline for the backup period without system interruption. This enables 24x7 operation with no backup windows required for the databases.

If the StrongLink system is configured with an archive tier, the database backup files will be automatically ingested and copied to a specific share in the archive. This will segregate the database backups onto a special group of tapes for easy recovery. If the system doesn't have an archive tier, the user must select a datastore as the target for the backups. One way to do this is to set the system up with a Glacier store in which to place the database backups.

One of StrongLink's primary functions is to maintain the integrity of the UDIs over time. This is achieved by maintaining multiple copies and periodically comparing them against the checksum(s) calculated when the data was ingested. If a bad copy is found, it can be restored from a good copy. If your data set includes particularly large files, then it's a good practice to have them chunked upon ingest. In fact, certain stores (primarily object stores) require this for performance reasons. A side benefit of chunking is that a UDI can be repaired more quickly since only individual chunks that don't match their checksum have to be replaced. From a performance perspective, copies on tape aren't audited as often. There are several configuration parameters available to control how copies are made and how often they are audited for validity.

The last piece of the data integrity puzzle is the managed datastores themselves. After a catastrophic event, it's obviously not very useful to have a perfect set of database backups if you haven't backed up the datastore file systems as well. While cloud stores like S3 and Glacier are handled automatically by the provider, you must still perform regular backups of any in-house datastores you may have, such as Isilon or NetApp storage.

---

[3] StrongLink assumes no responsibility for the integrity of data stored on external (Out-Of-Band) storage given that end user clients have direct access to that storage and can modify or delete those files at will.

With StrongLink that can be accomplished by creating a multi-instance policy for files, by which synchronized copies are maintained in an active archive, cloud, or other device. With multi-instancing, rules may be set for all or parts of the data set which ensure that there are always multiple copies in redundant locations. StrongLink maintains coherency between the primary copy and any downstream instances. In may use cases, this capability eliminates the need for conventional backup applications.

Constellations and Galaxies

As noted in the definitions above, a Constellation is a group of Stars (nodes), and a Galaxy is a group of Constellations. You may be tempted to think of a Constellation as a cluster, and there are some similarities, but we coined the term because there are major differences as well. Generally speaking, the typical HPC cluster is fairly homogeneous, so any node can run any of the standard applications the cluster supports. However, in many cases clusters imply a head-node that acts as the central brain for distributed capabilities.

A StrongLink Constellation has no head node as the system is distributed across the Stars. In addition individual stars can be tuned, both at the hardware and software level, to optimally perform specific functions. For example, you might have one node configuration that is biased towards big I/O to the datastores, another towards running database shards or replica sets, another towards workflow execution, and another towards servicing client API requests very responsively.

These different types of StrongLink Stars will be available in the latter part of 2017.

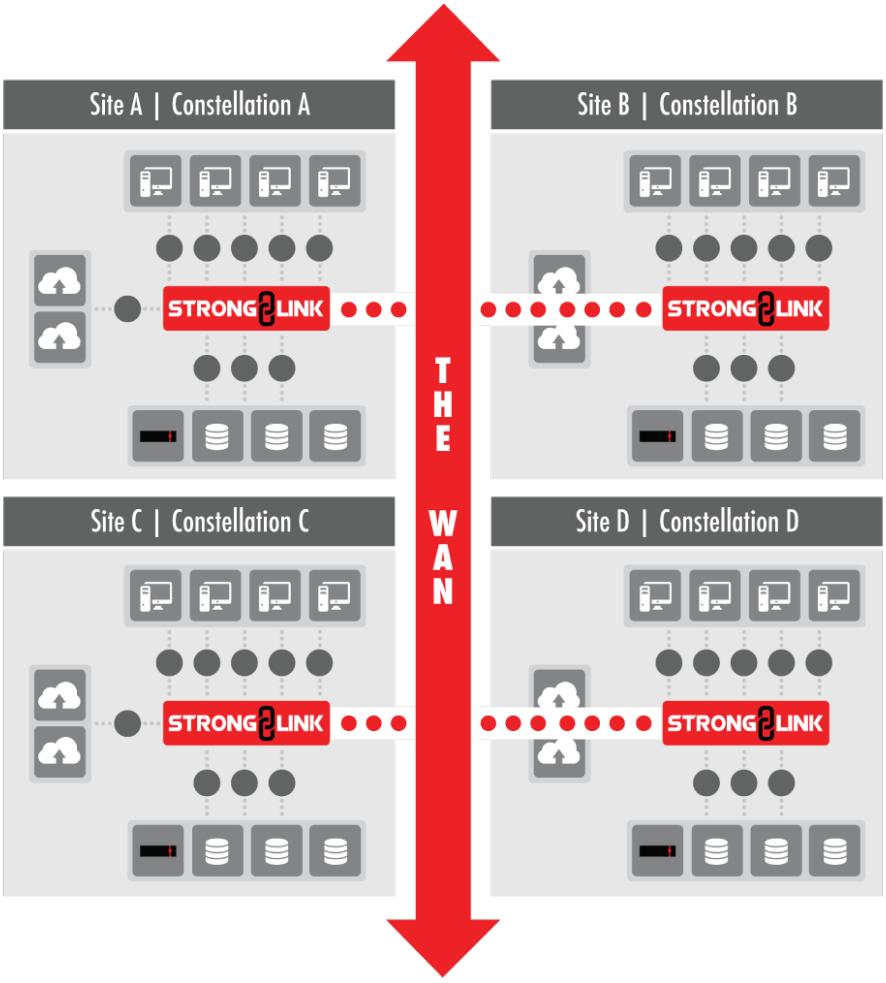

Figure 5 – StrongLink Constellation

StrongLink's multi-component, no-master Constellation is the crux of its self-healing capability. A well-deployed Constellation will have several instances of each component running on a decent-sized subset of the constellation. Generally, each component should be spread across a different subset of nodes, as this promotes system reliability and doesn't overload every node in the Constellation.

For organizations with a large number of assets to manage, some nodes should only be dedicated to running database shards or replica sets, especially if there is workflow automation or a non-trivial amount of metadata search in the workload.

Note: These types of choices will be outlined as part of an initial needs assessment and configuration proposal done by SBDS engineers. In all cases, StrongLink configurations may evolve non-disruptively as use cases develop over time. In other words, choices such as these do not lock customers into one way of doing things forever.

In most cases, an organization with multiple sites will deploy one Constellation per site and then federate them into a Galaxy. Strictly speaking, the boundary between constellations is a WAN link[4]. Figure 5 (previous page) shows a Galaxy with four constellations.

Galaxies may be deployed to implement DR or QoS policies. By design, *all* metadata is replicated (with configurable periodicity) to all constellations in a Galaxy. This provides *the same global namespace to all sites* in the Galaxy and assurance that all the metadata will survive the complete loss of a site. This has implications in terms of support infrastructure such as LDAP or AD. For example, to consistently maintain the owner information for a resource, the user data from the directory service must be the same at all sites.

Conversely, StrongLink doesn't automatically replicate UDIs to all sites in the Galaxy because while metadata is quite small (usually around 1kB per version of a resource), file sizes file sizes vary considerably; some organizational WAN links would quickly be overrun if StrongLink replicated everything to all sites. There's also the issue of how much storage is available at each site. That is why the organization must decide which UDIs to be replicated between constellations based upon business rules, or other triggers. This can be done by datastore, Smart Pool, namespace, data type, or a combination thereof.

Another aspect to consider when replicating UDIs is whether the target site has an archive tier. If so, then a common practice is to replicate the UDIs to a working cache datastore at the remote site(s) so they can then be archived at the remote site. Once that is done and the checksum has been verified, the UDI copies in the remote site working cache can be deleted.

One final note: a datastore cannot span constellations. By that, we mean that if Constellation *A* has a datastore named '*foo*', then only nodes in Constellation *A* can service requests to store data in or recall data from '*foo*'. It is however totally legal to make UDI copies to datastores that are part of another Constellation within the Galaxy. So, if Constellation *A* is part of a Galaxy that has another Constellation *B* with datastore '*bar*', it's common to make a UDI copy from '*foo*' to '*bar*'. This is, in fact, the very definition of replication for StrongLink. Remember, the single resource points to all copies of the UDI. In this case, the same resource record will exist in the databases at both sites *A* and *B*. And both of those records will point to the two UDI copies on '*foo*' and '*bar*'.

[4] The one exception to this is a StrongLink Star Gate, or remote office gateway, which allows small organizations with a central office and one or more small satellite offices to deploy an extended Constellation in a Hub-and-Spoke topology. StrongLink Star Gates will be available in Q4-2017.

With this in mind, let's examine a few failure scenarios to see the impact on client systems. To keep it simple, we'll use the configuration described in the previous paragraph. In normal operation, the clients at Site A will mount their shares from Constellation *A* while the clients at Site B will mount their shares from Constellation *B*. All clients at both sites can potentially see the exact same global namespace[5].

Scenario 1: A node fails in Constellation *A*. In this case, assuming the constellations were deployed correctly to support self-healing, there will be almost no impact to the clients at site A and zero impact to the clients at site B. The worst that can happen is the node with the DB writer is the one that drops out. In this case, it will take a few seconds for the database nodes to promote a new writer from the existing readers[6]. This is all automated.

Scenario 2: The physical storage for a datastore at site A (e.g. '*foo*') goes offline for service. If all the UDIs on '*foo*' have copies on another store at site A, then there would be zero impact to clients at either site. If some UDIs have copies on another store at site A, but some only have copies on another store at site B, then there might be a slight delay opening the UDIs that have to be accessed across the WAN. If there are no copies on faster tiers such as SSD or an Isilon, but there is a copy on the archive tier (either at site A or site B), then the archive copy would be recalled to cache and served to the client from the cache. The takeaway from this scenario is that StrongLink always tries to open the UDI copy on the fastest tier, i.e. the one closest to the client. It should be noted that chunking can significantly improve StrongLink's ability to mask WAN latency.

Scenario 3: The entire Constellation A is taken offline for some reason. If the WAN is still up, the clients can remount their shares from the Constellation B. This requires minimal manual intervention. The additional load this puts on the Site B Constellation and the consequent impact on users at Site B should be considered. CIFS/SMB and NFS operation across WAN links may be problematic as well, due to limitations of the protocol or link speed. That said, customers may allow key clients may remount from the B Constellation and work at a reduced performance level.

A better alternative, especially when WAN links aren't very fast or the local outage will be extended, would be to copy critical assets to a local system at Site A using an alternative StrongLink-supported protocol such as sFTP, do whatever updates are required, and then re-ingest it to the B Constellation to create a new version of the resource. When the A Constellation comes back online, it will immediately begin updating its databases to reflect

[5] What the client actually sees is a function of their access permissions and how the namespace export was defined.

[6] There is one writer per database shard so users need not be concerned about this being a performance bottleneck.

any changes that occurred at Site B while it was down. Once the metadata syncing is complete, the process of replicating new UDIs meeting the configured replication criteria from Site B to Site A will begin.

The system administrator can configure the A Constellation with a bandwidth limit on the UDI replication process to prevent swamping the WAN link. It should be noted that UDI replication is a PULL process whereas metadata replication is a PUSH process. When a Constellation gets a metadata record from another constellation, it will determine if the associated UDI meets the replication criteria, and attempt to start a copy if it does. The source Constellation will respond with a flow control response, where required, to stay within its bandwidth limit. Otherwise, it will allow the pulling Constellation to proceed with the copy. The pulling Constellation will meter its read requests to stay within its own configured bandwidth limitation.

Data Provenance

Providing 100% certainty of data provenance is one of the core drivers of StrongLink's architecture. So, what does this really mean?

The first component of data provenance is access control. StrongLink maintains tight access control for every resource in the system. Access is based on each individual user account. These accounts can be created locally within StrongLink, or can be part of your enterprise authentication system; both LDAP and AD are supported. When an account is created, it has no access rights granted. The process of granting an account access to various resources in the system involves creating roles, ACLs and ACSes. Time for some more definitions:

- **ACL**: An Access Control List is a set of ACSes.
- **ACS**: An Access Control Specifier defines the access rights a role has for the selected entity in the system.
- **Domain:** A namespace for a related group of users.
- **Role**: a named entity given access rights to the system, which are granted to one or more users. Permissions to access the system cannot be granted to a user. They must be granted to a role and that role must be granted to the appropriate user(s). A role is analogous to a Unix GID. A newly created role has absolutely zero access to the system; it must be granted one or more ACLs or ACSes to have defined access to the system.

- **User**: A named entity given controlled access to the system. User names are generally associated with a person, but may also be used to represent a software client, or a function or group of functions within a software client. Permissions to access the system cannot be granted to a user. They must be granted to a role and that role must be granted to the appropriate user(s)[7]. This is analogous to a Unix UID. A newly created user has absolutely zero access to the system.

It should be noted that user names and role names are *always* qualified by the domain in which they were created. As such, they must be unique within a domain, but not within the system as a whole.

The second component of data provenance is a comprehensive, secure and immutable audit trail. StrongLink audits *every* operation in the system. The audit records are encrypted before being written to the rotating audit logs which are marked immutable once written. In systems configured with an archive tier, the audit logs are automatically archived. If there is no archive tier, then the system administrator is responsible for configuring a target storage space in which to copy the logs and for regularly backing up that space.

The final component of data provenance is the ability to prevent the modification of a resource or its associated UDI. StrongLink supports marking a resource as read only (R/O). When this occurs, StrongLink will also set the associated UDI to R/O. Moreover, for any UDI copies that reside on underlying storage that supports immutability, the UDI will be set to the immutable state. When a resource is put into the R/O state, only the system administrator can change it back to R/W, and that action is recorded in the audit log as are any subsequent modifications or deletions. In other words, once a file is marked R/O, no one except for the system administrator can set the file back to R/W, not even the creator or a user with write permission on the resource.

[7] When integrating with LDAP or AD systems that have granted permissions directly to a user, StrongLink automatically creates a role, based on the user name, which is assigned to the user. The access granted directly to the user in the directory server is granted to the user-specific role within StrongLink.