STRONGLINK

Product Description

STRONGBOX
DATA SOLUTIONS

Contents

Who is StrongBox Data Solutions?

StrongBox® Data Solutions (SDS) is a worldwide leader in big data storage, providing powerful solutions to many of the world's largest corporations, governments and organizations. SDS is a pioneer in simplifying data management and reducing storage costs with revolutionary technology.

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, 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 SDS because we offer the best solutions, not because they're locked into proprietary software or architectures.

What is StrongLINK?

StrongLINK is a system that helps manage and exploit your ever-growing data set effectively and efficiently.

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 the amount of data they have
- Have legacy data silos which are often driven by application-specific data server components
- Need accountability for their data
- Need an effective bit-rot audit/recovery methodology
- Are struggling with accurate data provenance, retention and periodic purges for compliance purposes
- Are trying to avoid NAS/SAN sprawl
- Need to control CapEx and OpEx while maximizing their existing infrastructure
- Need to distribute data to multiple locations for quality of service (QoS) 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 standardize their approach to workflow implementation to enable a more efficient use of resources
- Are embracing the concept of Data Life Cycle 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 a number of standard protocols, including NFS, CIFS/SMB, HTTP(S), sFTP and S3
- Manage and export the same global namespace across all your sites
- Manage data access with Access Control Lists that have granularity down to the individually managed digital asset level
- Maintain a complete and accurate audit trail of who did what to any data item in the system with no possibility of alteration or deletion
- Transparently and automatically maintain multiple check-summed copies of your data, based on configurable policies
- Automatically extract file type-specific metadata as your data is ingested into the system
- Define custom metadata attributes and manually or automatically apply them during or after the data ingest process
- Retain multiple versions of every managed digital asset and associated metadata in the system
- Support tiering policies in order to automatically keep data on the appropriate storage tiers, based on cost and access requirements
- Transparently integrate cloud storage, including configurable automatic encryption and periodic auditing of data asset copies made to cloud stores
- 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
- Keep running, even in the presence of a partial system failure

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 such as nodejs, MongoDB, MySQL and ZeroMQ. We deliberately avoided proprietary protocols and technologies wherever possible to ensure your data is never held hostage.

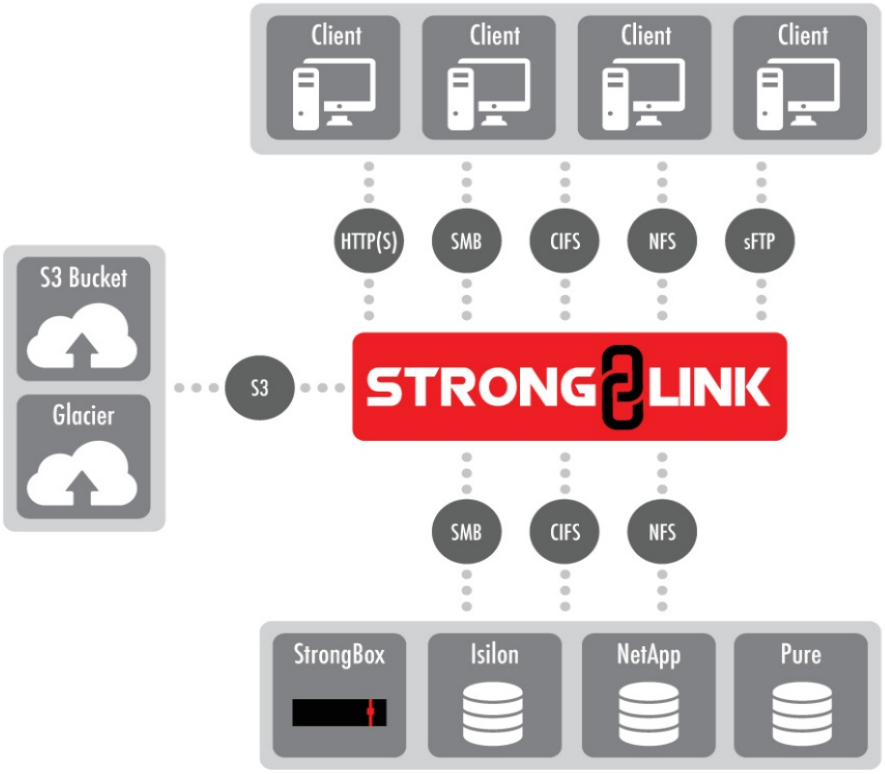

Figure 1

Figure 1 shows a logical view of a StrongLINK deployment. 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 usage models, described in the next sections.

1) Out-Of-Band Model

In this model, StrongLINK is deployed alongside your existing storage (see Figure 2). Your client systems continue to mount and access your storage directly, with your storage seeing StrongLINK as yet another client system. This model is most appropriate when StrongLINK is used as a smart archival system.

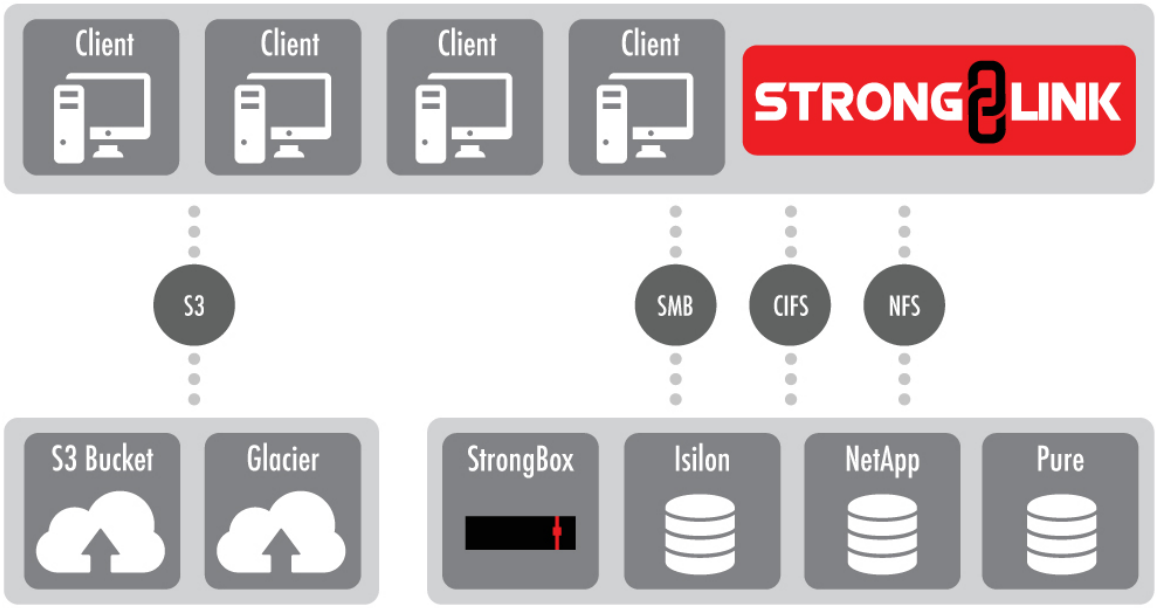

Figure 2

Deployment is as simple as using our 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 ingested
- Tell StrongLINK about the shares exported from those of your existing storage systems that need to be actively archived
- Pull the trigger

StrongLINK will walk all the mounted stores and ingest all the files. The ingest process consists of extracting metadata from known file types, storing the metadata in its database, generating checksums of the file, and copying the file to the archive storage.

The primary upside of this 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 a user can delete a file while StrongLINK is ingesting it, something that can easily be avoided by training users to verify the archive status of a file before deleting it. If a remote file or object storage system supports setting an immutability or R/O bit on a file or object, StrongLINK will attempt to do so during the ingest. Once complete, the original access permissions are restored.

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

StrongLINK will continue to monitor the configured storage for new files or updates to previously archived files. As these changes occur, the new files or updated versions will be ingested 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. Versioned data assets can be pruned to keep archive space from growing unbounded.

As a final note, StrongLINK does not assume an active management role *for your existing storage* in this 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 solely 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 of portions or the entire global namespace maintained by StrongLINK.

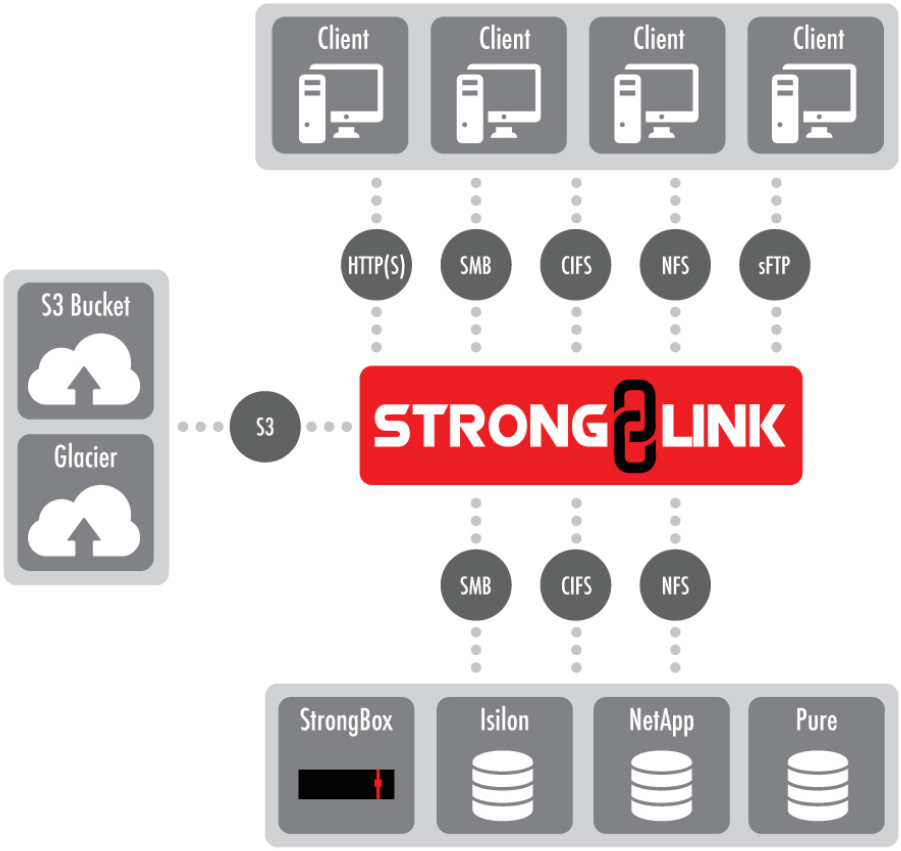

Figure 3

With this 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 quite similar to that 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 SSD or rotating storage behind the StrongLINK system (appliance or VM), whereas the In-Band Model does.

So, in this deployment model, copies of your data will be made on one or more stores managed by StrongLINK. These stores could be SSD, spin-down or traditional rotating storage, or LTFS tape. The stores can be configured in a tiered structure that allows active placement of data, based on cost and access requirements. Additionally, individual stores can be aggregated into pools, generally to increase I/O performance and the effective size of the pool. 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 store 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 Model does, or by "pushing" the data into StrongLINK using traditional 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:

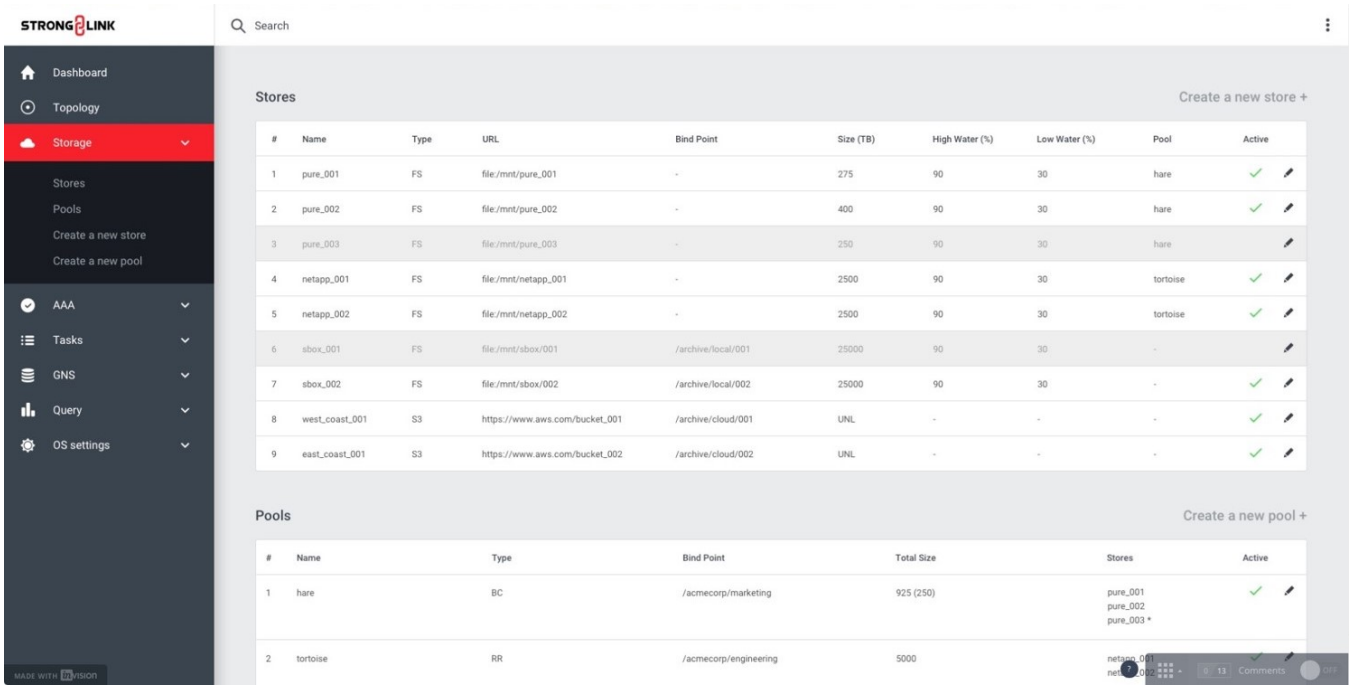

Figure 4

3) Hybrid Model

As previously mentioned, unless a completely new site is being deployed, a transition period is required when standing up an In-Band StrongLINK system, which is where the Hybrid Model comes into play. Some storage (new storage as part of a modernization project or a portion of the existing storage) is provisioned behind the StrongLINK system. This storage, which we call “managed storage”, is set up as one or more stores and/or pools. These stores and/or pools become a target for the ingestion of data 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 configured.

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 of the unmanaged storage is eventually converted to managed or retired so the system reverts to the In-Band Model. However, it’s possible to maintain some Out-Of-Band unmanaged storage purely for archival purposes if that configuration better suits your workflow requirements.

This same methodology allows transparent upgrades to your physical storage as new technologies come online. You simply stand up the new storage, set it up as one or more new stores behind StrongLINK, start making copies from your older technology managed storage, and verify 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. This is a perfect example of where temporary augmentation of the StrongLINK system using elastic cloud capability (or temporary deployment of VMs on your internal infrastructure) makes a lot of sense. This allows the migration to proceed at a rapid pace without degrading service to your client systems or busting your budget with CapEx for capacity you don’t typically require.

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.
- **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 couples multiple sites together into a single global namespace. Galaxies are usually deployed to implement Disaster Recovery strategies 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. Generally, it will have multiple CPUs and/or cores. In most StrongLINK deployments, nodes 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.
- **Pool**: a group of datastores connected by a policy that controls how UDIs are distributed among the member stores.

- **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.
- **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 or an object in an object store.

StrongLINK is a multi-component, no-master server application. The components communicate over a mesh network topology implemented with ZeroMQ, a well-supported, robust, efficient and highly successful network communications package. All inter-component communication payloads are encrypted; because the messages are quite small, the encryption does not adversely impact system performance. We've deliberately avoided SSL to eliminate the certificate acceptance problems that commonly occur with self-signed certificates. 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, zero backup windows operation

StrongLINK leverages two open source databases to store unstructured and highly structured data types which are used to run the system. Both databases 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 several standard protocols.

The StrongLINK store manager also supports aggregating datastores into 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[3]. The most common use cases for this are automatic UDI copy generation and automatic datastore purging to ensure zero failure. 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 to 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 is a simple least recently used query, but you can configure additional constraints such as ‘the file must have a copy on store “archive”’. Keep in mind that reaching the low water mark may not be possible when adding additional constraints beyond the simple LRU policy.

[3] General scripting to enable customer-specific workflows is not publicly exposed until V1.6, which releases at the beginning of Q2, 2017. It is available prior to that as a PSS item.

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 one of multiple checksum types when your data is ingested. If the UDI is chunked on ingest, we generate 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.
- **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.
- **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. 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

The 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 a number of predefined fragments, and you can define your own as well. StrongLINK stores the metadata in JSON format, 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 what data they have to work with, and their retained data sets have become way too large to grok without some kind of data management system. By collecting all the metadata, we provide the ability to understand what you have so you can leverage it in the future, long after you've forgotten what file contains last year's research results from "Project X". You could also submit queries to get a list of all images with location x, y in the field of view on Feb 3rd, 2001 between 3 pm and 4 pm local time, for example. Then there's another critically important piece to the puzzle: when you have billions of data items under management, it's pretty 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[4].

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.

[4] Despite our best efforts, there are certain instances where queries can take a significant amount of time to produce a result set. The most common example of this is applying a sort to the result set as part of the query. We can and do perform the result set sorting as it is being generated, but, by definition, we cannot complete the sort until we have the full result set. We've done our absolute best to minimize these instances.

StrongLINK maintains multiple versions of both resources and UDIs. 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. Now, 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 in StrongLINK[5]; and 3) the integrity of the datastores themselves.

Our 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 have replica sets so we can take a replica set offline for the backup period, and the system will continue to operate normally. 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 backup the database backups in some fashion. One way to do this is to set the system up with a Glacier store in which to stash 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 idea to have them chunked when ingested. 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 a number of configuration parameters available to control how copies are made and how often they are audited for validity.

[5] 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.

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.

Constellations and Galaxies

As noted in the definitions above, a constellation is a group of 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. StrongLINK constellation nodes can be tuned, both at the hardware and software level, to optically 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.

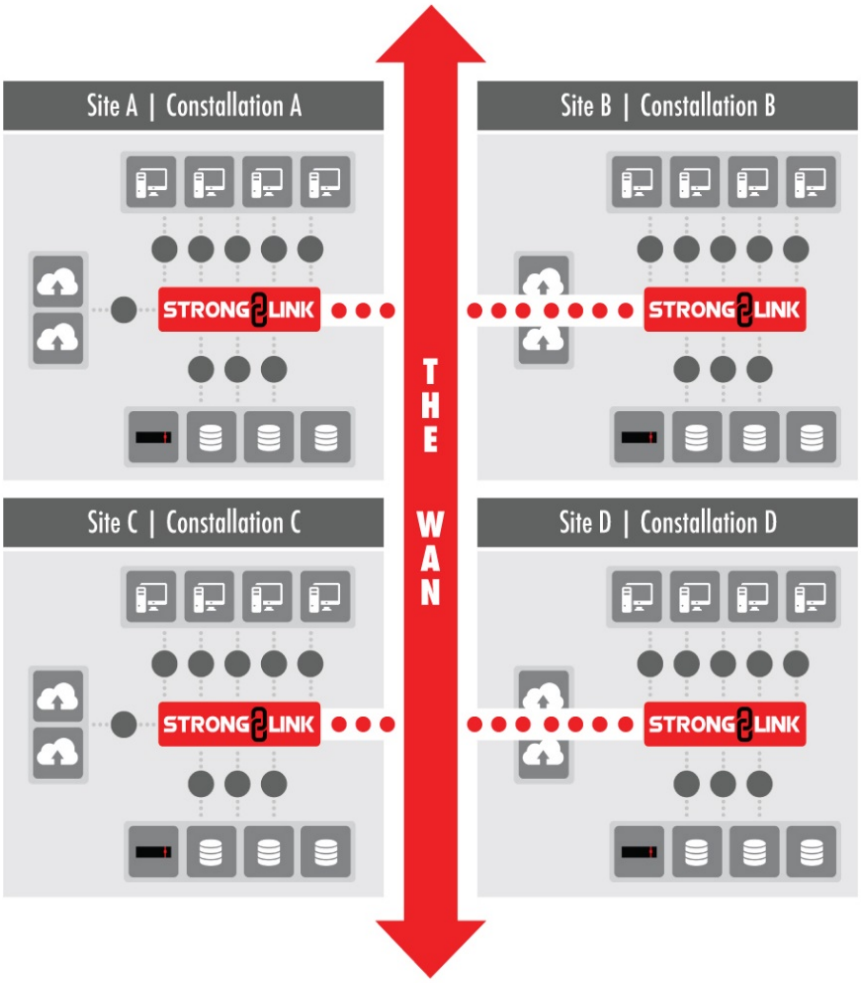

Figure 4

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 by trying to run too much on every mode. It's also a more cost-effective overall solution since the node hardware doesn't have to be the latest and fastest in most cases. 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.

In most cases, an organization with multiple sites will deploy one constellation per site and then federate them into a galaxy. More strictly speaking, the boundary between constellations is a WAN link[6]. Figure 4 (next page) shows a galaxy with four constellations.

Galaxies are 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 great 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, in order 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 metadata is quite small (usually around 1kB per version of a resource) while UDI sizes vary considerably, and some organizational WAN links would quickly be overrun if we replicated everything. There's also the issue of how much storage is available at each site. That is why the organization must configure the UDIs to be replicated. This can be done by store, 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*.

[6] The one exception to this is StrongLINK's 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.

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[7].

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[8].

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.

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

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

Scenario 3: the entire *A* constellation takes a powder. Assuming that the WAN is still up, the clients can remount their shares from the *B* constellation. 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 and NFS operation across WAN links may be problematic as well. That said, key clients can 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 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 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% rock solid 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 accounts. These accounts can be created locally within StrongLINK, or can be part of your enterprise directory 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 particular type of 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)[9]. This is analogous to a Unix UID. A newly created user has absolutely zero access to the system.

[9] 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.

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.