Global-Scale Archival Goals

- Durability
- Data is stored for centuries or longer.
- Verifiability
- Data is not subject to substitution attacks.
- Availability
- Data is accessible most of the time.
- Where data is defined on 99% of availability.
- Maintainability
- System recovers from server and network failures.
- Efficiently incorporates new resources.
- Atomicity
- Updates are applied atomically.
- Privacy
- Information is only visible to those who have access rights.
- Performance
- Response time is bounded.

Archival Model

- Archive Data Structures
  - Archive is a linearly ordered sequence of versions.
  - Each version is a read-only sequence of bytes.
  - E.g. an archive might be a file directory or a database record.
- Naming
  - Globally Unique Identifier (GUID).
  - Archives are uniquely specified by a version GUID (V-GUID).
  - Versions are inscrutable and provide for time travel.
- Operations
  - Update Operations: Add versions to the end of the sequence of a given archive.
  - Read Operations: Read data from a specific version.
- Serializer provides consistency
  - Entity in network that provides atomicity.
  - Provides an A-GUID to V-GUID mapping.
  - Creates a serial order and submits updates.
  - Verifies that the client has update privileges.
  - Atomically applies update to the archive and generates a new V-GUID.
  - Sends fragments from an update to storage servers.
- Interface
  - Generate new archive interface.
    - create(name, identity, keys) => A-GUID.
  - Query Interface
    - query(A-GUID, Specific) => V-GUID.
    - Specific = exact range, version?, etc.
  - Read interface: read(V-GUID, offset, length) => data.
  - Write interface:
    - write(A-GUID, data) => V-GUID.
    - A-GUID, V-GUID, data, (offset, length).
    - replace(V-GUID, offset, data, allowbr) => V-GUID or null.
    - allowbr denotes whether operation allowed to generate branch.

Archival Process: Data Integrity

- Top hash is a block GUID (B-GUID).
- Fragments and blocks are self-verifying.

Archival Mode

- No archiving
- Inline archiving
  - Synchronous: m = 16, n = 32
  - Delayed archiving
    - Asynchronous: m = 16, n = 32

Efficient Repair

- Local
  - Durability enhancement techniques such as RAID.
  - Servers proactively copy data to new disk.
  - Servers periodically verify the integrity of local data.
- Distributed
  - Exploit Tapestry's distributed information and locality properties.
- Global
  - As effective as distributed mechanisms.

Future Directions

- Tapestry is a location-independent routing infrastructure.
  - Fragments and serializers are both named by opaque bit strings (GUIDs).
  - Tapestry can perform location-independent routing of messages directly to objects using only GUIDs.
  - Tapestry is an IP overlay network that uses a distributed fault-tolerant architecture to track the location of every object in the network.
  - Tapestry has two components: a routing mesh and a distributed directory service.
  - Routing in Tapestry
    - Nodes are connected to other nodes via neighbor links.
    - Any node can route to any other by resolving one digit at a time:
      - e.g. 000 => 2188 = 5098 = 7598 = 4598
      - Each GUID is associated with a particular Root node.

Enabling Technology: Tapestry

- Publish
  - Send a message toward the root.
  - Leaving back-pointers at each hop.
  - If a node 432 stores a fragment with GUID, name 4598:
    - Publish path: 4432 => 0478 => 9098 => 7598 => 4598

Conclusion

The OceanStore archive combines several techniques to satisfy the goals of a global-scale archival system.

- Erasure codes provide durability and availability.
- Verification trees provide verifiability.
- Introspective failure analysis, automatic repair, and location independent routing promote maintainability.
- The serializer provides atomicity.
- End-to-end encryption (not discussed in this paper) provides privacy.

Result

- Archival storage that is online and inline.
- Data is durable and accessible.
- Archival storage that has good user perceived latency.

Privacy

- Operations
  - Generate new archive interface.
    - create(name, identity, keys) => A-GUID.
  - Query Interface
    - query(A-GUID, Specific) => V-GUID.
    - Specific = exact range, version?, etc.
  - Read interface: read(V-GUID, offset, length) => data.
  - Write interface:
    - write(A-GUID, data) => V-GUID.
    - A-GUID, V-GUID, data, (offset, length).
    - replace(V-GUID, offset, data, allowbr) => V-GUID or null.
    - allowbr denotes whether operation allowed to generate branch.

Verifiability

- Update Operations: Add versions to the end of the sequence of a given archive.
- Read Operations: Read data from a specific version.

Availability

- Erasure codes provide redundancy without overhead of replication.
- Divide an object into m fragments.
- Reencode them into n fragments.
- A rate r = m/n code increases storage cost by a factor of 1/r.
- Key property is that original object can be reconstructed from any m fragments.
- E.g. using an r = 2/3 code, divide a block into m = 16 fragments, and encode the original m fragments into n = 64 fragments.
- Increase storage cost by a factor of 4.
- Example implementations
  - Reed-Solomon Codes.
  - Turbo Codes.
  - Interleaved Reed-Solomon.

Case for Erasure Codes

- An archive is implemented on a collection of independently failing disks.
- Failed disks immediately replaced by new, blank ones.
- Each archival fragment for a given block is placed on a unique, randomly selected disk.
- A repair epoch
  - Time period between a global sweep, where a repair process scans the system, attempting to restore redundancy.

Archival Mode

- Globally Unique IDentifier (GUID).
- Archives are uniquely specified by a version GUID (V-GUID).
- Versions are inscrutable and provide for time travel.
- Entity in network that provides atomicity.
- Provides an A-GUID to V-GUID mapping.
- Creates a serial order and submits updates.
- Verifies that the client has update privileges.
- Atomically applies update to the archive and generates a new V-GUID.
- Sends fragments from an update to storage servers.

Archival Process: Data Integrity

- Top hash is a block GUID (B-GUID).
- Fragments and blocks are self-verifying.

Archival Mode

- No archiving
- Inline archiving
  - Synchronous: m = 16, n = 32
  - Delayed archiving
    - Asynchronous: m = 16, n = 32

Efficient Repair

- Local
  - Durability enhancement techniques such as RAID.
  - Servers proactively copy data to new disk.
  - Servers periodically verify the integrity of local data.
- Distributed
  - Exploit Tapestry's distributed information and locality properties.
- Global
  - As effective as distributed mechanisms.

Future Directions

- Tapestry is a location-independent routing infrastructure.
  - Fragments and serializers are both named by opaque bit strings (GUIDs).
  - Tapestry can perform location-independent routing of messages directly to objects using only GUIDs.
  - Tapestry is an IP overlay network that uses a distributed fault-tolerant architecture to track the location of every object in the network.
  - Tapestry has two components: a routing mesh and a distributed directory service.
  - Routing in Tapestry
    - Nodes are connected to other nodes via neighbor links.
    - Any node can route to any other by resolving one digit at a time:
      - e.g. 000 => 2188 => 5098 => 7598 => 4598
      - Each GUID is associated with a particular Root node.

Enabling Technology: Tapestry

- Publish
  - Send a message toward the root.
  - Leaving back-pointers at each hop.
  - If a node 432 stores a fragment with GUID, name 4598:
    - Publish path: 4432 => 0478 => 9098 => 7598 => 4598

Conclusion

The OceanStore archive combines several techniques to satisfy the goals of a global-scale archival system.

- Erasure codes provide durability and availability.
- Verification trees provide verifiability.
- Introspective failure analysis, automatic repair, and location independent routing promote maintainability.
- The serializer provides atomicity.
- End-to-end encryption (not discussed in this paper) provides privacy.

Result

- Archival storage that is online and inline.
- Data is durable and accessible.
- Archival storage that has good user perceived latency.

Availability

- Erasure codes provide redundancy without overhead of replication.
- Divide an object into m fragments.
- Reencode them into n fragments.
- A rate r = m/n code increases storage cost by a factor of 1/r.
- Key property is that original object can be reconstructed from any m fragments.
- E.g. using an r = 2/3 code, divide a block into m = 16 fragments, and encode the original m fragments into n = 64 fragments.
- Increase storage cost by a factor of 4.
- Example implementations
  - Reed-Solomon Codes.
  - Turbo Codes.
  - Interleaved Reed-Solomon.

Case for Erasure Codes

- An archive is implemented on a collection of independently failing disks.
- Failed disks immediately replaced by new, blank ones.
- Each archival fragment for a given block is placed on a unique, randomly selected disk.
- A repair epoch
  - Time period between a global sweep, where a repair process scans the system, attempting to restore redundancy.

Archival Mode

- Globally Unique IDentifier (GUID).
- Archives are uniquely specified by a version GUID (V-GUID).
- Versions are inscrutable and provide for time travel.
- Entity in network that provides atomicity.
- Provides an A-GUID to V-GUID mapping.
- Creates a serial order and submits updates.
- Verifies that the client has update privileges.
- Atomically applies update to the archive and generates a new V-GUID.
- Sends fragments from an update to storage servers.

Archival Process: Data Integrity

- Top hash is a block GUID (B-GUID).
- Fragments and blocks are self-verifying.

Archival Mode

- No archiving
- Inline archiving
  - Synchronous: m = 16, n = 32
  - Delayed archiving
    - Asynchronous: m = 16, n = 32

Efficient Repair

- Local
  - Durability enhancement techniques such as RAID.
  - Servers proactively copy data to new disk.
  - Servers periodically verify the integrity of local data.
- Distributed
  - Exploit Tapestry's distributed information and locality properties.
- Global
  - As effective as distributed mechanisms.