OceanStore: An Architecture for Global-Scale Persistent Storage

John Kubiatowicz, David Bindel, Yan Chen, Steven Czerwinski, Patrick Eaton, Dennis Geels, Ramakrishna Gummadi, Sean Rhea, Hakim Weatherspoon, Westley Weimer, Chris Wells, and Ben Zhao

Presented by: Khoo Yit Phang
CMSC 818T
November 28, 2006
Why OceanStore?

Data Storage as Utility
Inherently Untrusted Infrastructure
The Data Utility Vision

- What if we could treat data [storage] as a utility?
  - Multiple providers
  - Economies of scale – high availability, performance, reliability

I’m skeptical, utilities nowadays are notorious for exactly the opposite of the above. Though, data is not as location sensitive as gas/electricity/etc.
Inherently Untrusted Infrastructure

1. No servers are trusted with data
   - All data is encrypted

2. Data should be “anywhere, anytime”
   - Nomadic data, promiscuous caching
   - Ensures high availability, robustness

Both follow from the (idealized) data utility model, where providers are freely interchangeable.
OceanStore Architecture

Naming
Access Control
Data Location & Routing
Update Model
Hybrid Serialization
Deep Archival
OceanStore API
Introspection
OceanStore Architecture

Naming
Access Control
Data Location & Routing
Update Model
Hybrid Serialization
Deep Archival
OceanStore API
Introspection
Quick Stuff

Naming
- GUID identifies all objects (files, directories, servers)
- Self-certified by hashing private key + name

Access Control
- Reader restriction – cryptography at client
- Writer restriction – Access Control Lists at server

Façade Legacy API
- E.g. provides POSIX-like file interface
OceanStore Architecture

Naming
Access Control
Data Location & Routing
Update Model
Hybrid Serialization
Deep Archival
OceanStore API
Introspection
Combined Data Location & Routing

- Two Stage Routing
  - Fast probabilistic Attenuated Bloom Filters
  - Deterministic Plaxton randomized hierarchical distributed mesh

- Data Location
  - Objects have a preferred root nodes (primary tier)
  - Each node maintains destination map of nearby objects
  - Short-circuit queries directly to destination

With data replication, this scheme is fault-tolerant.
Attenuated Bloom Filters

- Each node maintains an array of Bloom filters per edge.
- Bloom filter $i$ is a union of objects at exactly $i$ hops away.
Attenuated Bloom Filter (e.g.)
Attenuated Bloom Filter (e.g.)
Attenuated Bloom Filter (e.g.)
Attenuated Bloom Filter (e.g.)
Randomized Hierarchical Distributed Data Structure (Plaxton Mesh, Tapestry?)

- Basic idea: treat node-ID as location in tree, rooted at the destination node—$O(\log n)$ traversal.
- Route by resolving ID piece-wise.
Plaxton Mesh (e.g. to 4598)
Plaxton Mesh (e.g. to 4598)
Plaxton Mesh (e.g. to 4598)
Plaxton Mesh (e.g. to 4598)
Plaxton Mesh + Location
Bayou-like Update Model

- **Predicate-action**: Commit updates when predicates hold true.

- **Can only operate on cipher-text due to untrusted servers!**
  - Limited updates: replace-block, append, insert-block, delete-block
  - Indirection index-block to support insert/delete.
Hybrid Serialization Scheme

Concurrent optimistic and pessimistic updates:

1. *Primary tiers* use slow Byzantine agreement
2. *Secondary tiers* use fast tentative epidemic updates
Serialization – Update
Serialization – Agreement
Serialization – Disseminate
Deep Archival (encode)

- Archive data robustly using *erasure codes* (e.g. interleaved Read-Solomon, Tornado)
Deep Archival (decode)

- Can reliably recreate the object using any $n$ fragments
- Fragmentation increases reliability
Introspection

- *How to administer/tune millions of servers?*
- Introspection facility to summarize node activity
- Can be used for:
  1. Recognizing object clusters
  2. Managing replicas/archive fragments
In Summary