Logan: Automatic Management for Evolvable, Large-Scale Archival Storage

Mark W. Storer, Kevin M. Greenan, Ian F. Adams, Ethan L. Miller, Darrell D.E. Long, Kaladhar Voruganti*

Storage Systems Research Center, University of California, Santa Cruz; *Network Appliance

The Archival Management Problem

Archival storage systems designed to preserve scientific data, business data, and consumer data must maintain and safeguard tens to hundreds of petabytes of data on tens of thousands of media for decades.

While existing solutions can provide good energy efficiency and relatively low cost, they do not adapt well to continuous improvements in technology, becoming less efficient relative to current technology as they age. This implies an endless series of wholesale migrations and upgrades to remain efficient and up to date.

Our approach, Logan, manages node addition, removal, and failure on a distributed network of intelligent storage appliances, allowing the system to gradually evolve as device technology advances. By automatically handling most of the common administration chores — integrating new devices into the system, managing groups of devices that work together to provide redundancy, and recovering from failed devices — Logan reduces management overhead and thus cost. Logan can also improve cost and space efficiency by identifying and decommissioning outdated devices, thus reducing space and power requirements for the archival storage system.

Managements Tasks in Logan

- Scale Out: expand the capacity of the system by creating redundancy groups and assigning available free-space on devices in the management group
- Recovery: in the event of a device failure, determine where data will be recovered to
- Maintenance: monitor the health and efficiency of the management group and actively identify nodes that are ready to be decommissioned

Scalable Hashing with LH*

Distributed data structure that relies on linear hashing to place items into distributed hash buckets that gracefully expand when efficiently loaded

- Completely distributed, no reliance on a centralized, master site
- Provides primitives for search, insertion, splitting, deletion that do not require atomic updates to multiple clients

Figure 3: When inserting Item 25, bucket B1 exceeds its maximum load. Since it holds the splitting token, it splits to form bucket B3, passes the splitting token to the next bucket and rehashes the items in bucket B1.

Challenge: Global Knowledge in a Distributed System

As a distributed system grows, it becomes impractical for any single device to maintain accurate global knowledge of the system due to memory and messaging overhead.

A peta-scale archival system would require hundreds of thousands of devices, so we utilize a hierarchical solution involving semi-autonomous management groups that expand and divide as the system grows.

Figure 4: Eight management groups arranged in a hypercube of dimension 3. Nodes involved in routing from groups 2 to 5 shown in blue. Routing is done in O(lg n) time, since each hop brings the message one bit closer to its destination.

Figure 5: When parent P (node 2), produces a child C (node 6), it provides the child with a list of its grandparents G1 and G2 (nodes 0 and 3). The child then calculates which, if any of its bitwise neighbors it needs an introduction to from its grandparent.

Challenge: Automating Management Tasks

As the system grows, tasks normally done manually by an administrator become increasingly complex and expensive. To automate these tasks, we use a series of statistics gathered from the nodes in the management group, and heuristic algorithms to find “good” solutions.

References
