UNIVERSITY OF MINNESOTA

DISC

Integrated Infrastructure for Secure and Efficient Long-Term Data Management

Presented by Yongdae Kim
University of Minnesota

PI: Andrew Odlyzko
co-PI: David Lilja, Yongdae Kim

HEC-IWG Workshop'06

Introduction

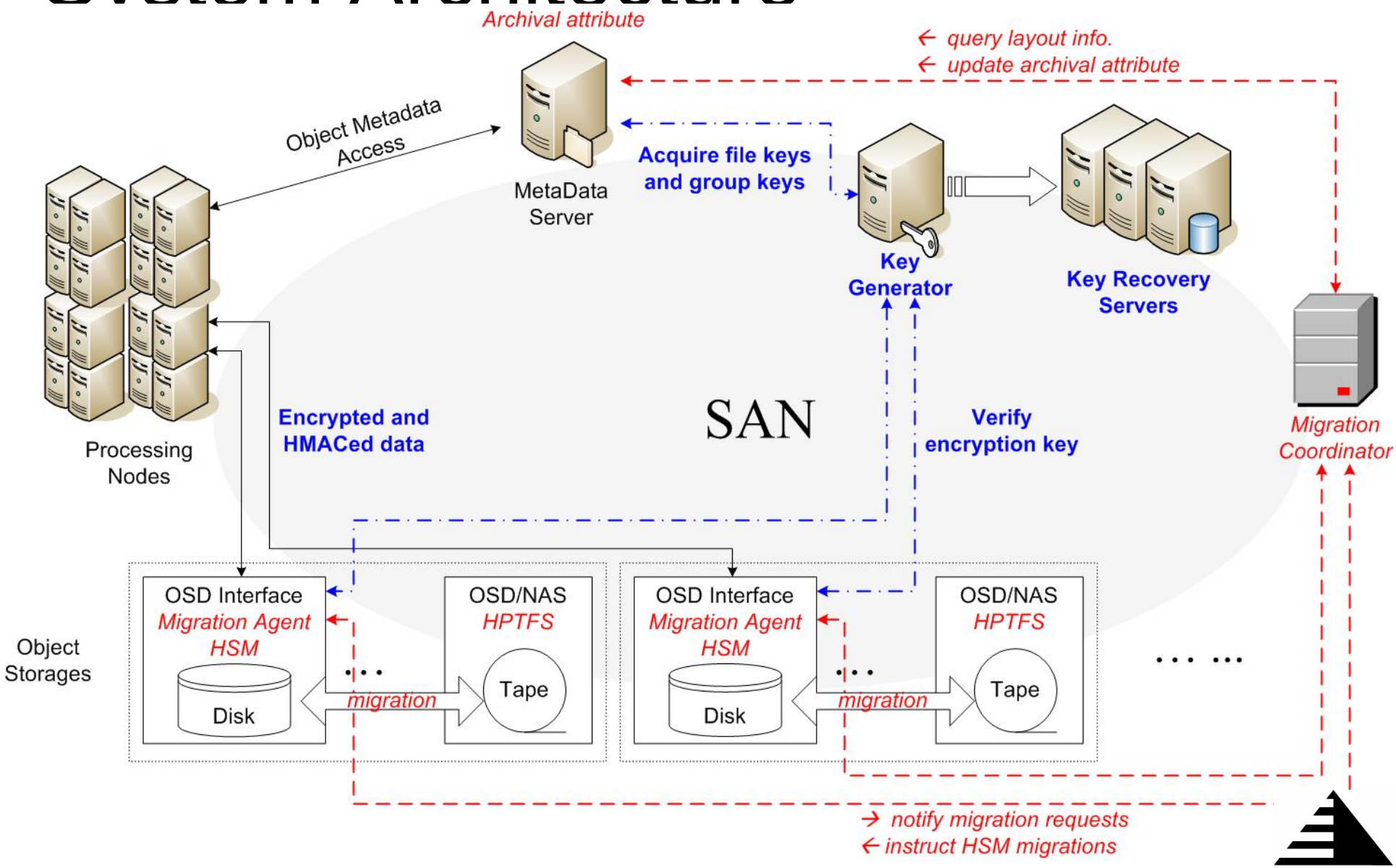
- HPC
  - Improvement of SGS on-line storage system through Lustre and Panasas Active Scale
  - Archival storage has gotten less attention ⇒ key bottleneck for HPC
    - 35 TB/hr in 2003 ⇒ 350 TM/hr in 2006 [Grider 2006]
- Other businesses require SGS archival
  - check images, medical imaging, video/audio, email records
  - infrequently accessed but usually must be retained for long periods of time and must be readily accessible when needed
  - Legal/government mandates, e.g. Sarbanes-Oxley, HIPAA
- Long-term protection of cryptographic keys: a major challenge
  - Loss of keys
  - User and group membership changes
  - Retrieval of old data

Requirements and Focus

- Requirements for Long-term Data Archiving and Protection
  - High data archive and restore throughput
  - Automated and transparent management of data migrations in storage hierarchy
  - Efficient backup and retrieval of keys
  - Key recovery
  - Long-term management
    - group reorganization such as creation/deletion/split/merge
  - Usability
  - Scalability

- Focus of this project: Investigate archiving on OCFS
  - Transparent backup and archive functions
  - High-performance backup, restore, and data access operations
  - Efficient techniques for ensuring long-term data security and accessibility

System Architecture



Data Archiving

- Local HSM agent on OSD
  - automates the data migration between the OSD's internal storage and a designated archival storage on the SAN
  - allows parallel data migration paths to achieve high aggregated migration throughput
- Migration Coordinator
  - initiates parallel data migrations to take advantage of the parallel data paths provided by the physical topology
  - guarantees the consistent archiving state of a set of related data objects
  - helps to eliminate heavy loaded DMAPI
- High Performance Tape File System (HPTFS)
  - eases the sharing and usages of tape libraries as archival storages
  - enables accessing tape-based archival storage using either OSD interface or NAS interface.

Key Management

- Transparent encryption and key management
  - to improve usability and manageability
- Securing data at rest
  - End-to-end encryption = Writer encrypts, reader decrypts
  - Previous key management works focused on providing solutions satisfying a single requirement
    - e.g. Hierarchical key management for improving scalability, Key rolling for efficient recovery of past keys, Broadcast encryption and group key distribution for efficient revocation
  - This projects investigate key management solutions that satisfy multiple requirements at the same time.
- Key recovery and backup
  - Adopting and improving cryptographic key recovery mechanism for storage

Blending Multiple Requirements

- Limited Roll-back
  - Previous solutions allow to roll-back indefinitely
    - Not necessarily secure for all environments
  - Can we limit the number of roll-back so that the new user might have access to only specified number of keys (without sacrificing significant performance penalty)?
- Efficient Hierarchical Access Control
  - RBAC (Role-based Access Control) provides efficient grouping based on roles
  - Hierarchical key management may reduce number of keys managed by individual nodes
  - But, it fails to achieve similar efficiency as RBAC
    - i.e. revocation of higher-level node = revocation of all nodes under the high-level node
  - No effort to merge/split of groups in hierarchical key management
  - Can we apply broadcast encryption/group key management to improve these problems?

Blending Multiple Requirements II

- Ultimate goal: Hierarchical key management with limited key roll-back
  - Much more difficult than previous problems
  - All group keys have to be roll-back to the previous keys
  - Should be able to specify the number of roll-back period

Guaranteed Key Recovery

- Files are encrypted with a key, and the key is encrypted with a group key, and both are stored at the storage device.
  - As long as the group key is available and the integrity of the encrypted file is preserved, we will be able to decrypt the file.
- For key recovery purpose, additional keys will be stored.
  - Only few principles will be able to compute the key using threshold cryptography.
- Main question: how do you know if a user is actually using the key it is supposed to use?
  - Expensive cryptographic solution exists.
  - Need TPM?
  - Can we find more efficient solutions?