

# Integrated Infrastructure for Secure and Efficient Long-Term Data Management

Presented by David Du  
University of Minnesota

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



# Introduction

---

## ▶ HPC

- ▶ Improvement of SGS on-line storage system through Lustre and Panasas Active Scale
- ▶ Archival storage has gotten less attention  $\Rightarrow$  key bottleneck for HPC
  - ▶ 35 TB/hr in 2003  $\Rightarrow$  350 TB/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 key management
    - ▶ group reorganization such as creation/deletion/split/merge
  - ▶ Usability
  - ▶ Scalability
  
- ▶ **Original focus of this project: Investigate archiving on OCFS**
  - ▶ Transparent backup and archive functions: Mostly Done
  - ▶ High-performance backup, restore, and data access operations: Mostly Done
  - ▶ Efficient techniques for ensuring long-term data security and accessibility: On-going



# High-Performance and Transparent Backup

- ▶ David Du, Dingshan He, Changjin Hong, Jaehoon Jeong, Vishal Kher, Yongdae Kim, Yingping Lu, Aravindan Raghuvier, Sarah Sharafkandi, "Experiences in Building an Object-Based Storage System based on the OSD T-10 Standard", *14th NASA Goddard, 23rd IEEE Conference on Mass Storage Systems and Technologies, (2006)*
- ▶ Dingshan He, Xianbo Zhang, David H.C. Du, Gary Grider, "Coordinating Parallel Hierarchical Storage Management in Object-based Cluster File Systems", *14th NASA Goddard, 23rd IEEE Conference on Mass Storage Systems and Technologies, (2006)*
- ▶ Xianbo Zhang, David Du, Jim Hughes, Ravi Kavuri, "HPTFS: A High Performance Tape File System", *14th NASA Goddard, 23rd IEEE Conference on Mass Storage Systems and Technologies, (2006)*
- ▶ Nagapramod Mandagere, Jim Diehl, David Du, "GreenStor: Application-aided Energy-efficient Storage", *24th IEEE Conference on Mass Storage Systems and Technologies, (2007)*
- ▶ Dingshan He, Nagapramod Mandagere, David Du, "Design and Implementation of a Network Aware Object-based Tape Device", *24th IEEE Conference on Mass Storage Systems and Technologies, (2007)*



# Secure File Systems

---

- ▶ Vishal Kher, Eric Seppanen, Cory Leach, Yongdae Kim, "SGFS: Secure, Efficient and Policy-based Global File Sharing", *14th NASA Goddard, 23rd IEEE Conference on Mass Storage Systems and Technologies, (2006)*
- ▶ Vishal Kher, Yongdae Kim, "Building Trust in Storage Outsourcing: Secure Accounting of Utility Storage", *IEEE International Symposium on Reliable Distributed Systems, (2007)*

# Software Release I

---

- ▶ The DISC-OSD project at the University of Minnesota provides an implementation of an object based storage ecosystem. This toolkit contains a SCSI OSD ANSI-T10 compliant target, initiator drivers and a file system.
- ▶ It is currently open-sourced at source forge (<http://sourceforge.net/projects/disc-osd/>).
- ▶ It was downloaded more than 100 times so far.



# Software Release II

---

- ▶ CoreFS is a very basic network file system built on top of FUSE. The file system allows users to access files stored on the file server from remote machines.
- ▶ The goal of this file system is to give file system developers some form of basic distributed file system, which can be later modified as per the implementer's requirement. We will provide some core features - hence the name "coreFS".
- ▶ We are currently implementing encrypted file system on top of coreFS.
- ▶ It is currently open-sourced and was originally available from <http://www.cs.umn.edu/research/sclab/coreFS.html>.
- ▶ Now, it has been moved to sourceforge.  
(<http://sourceforge.net/projects/corefs/>)



# Current Focuses

---

- ▶ Data Deduplication
  - ▶ Efficient dedup algorithms
- ▶ Long-term key management
  - ▶ Key Management
  - ▶ Theoretical Foundation of File Encryption
- ▶ Efficiency of Data center (not covered)
  - ▶ Greenstor: Power management for data center
- ▶ Solid State Drives and Hybrid Drives (not covered)
- ▶ Application support: Intelligent Storage + Database (not covered)



# A Novel Data De-duplication Approach With Global Information

Guanlin Lu, Yu jin, David Du



# Motivation

---

- ▶ What is deduplication?
- ▶ Why is it useful?
  - ▶ In network communication
    - “Cache-based compaction”
  - ▶ In storage backup case, to reduce the backup size
  - ▶ In versioning control system, to reduce the size of all versions of a file.
    - “XDFS”
  - ▶ In Virtual Machine applications
- ▶ What will have impact on its performance (both time complexity and space saved)?

# Content-based Chunking Approach & Its Limitations

---

## ▶ Main idea

- ▶ Identify and then eliminate overlapping regions of data.

## ▶ Basic methods

- ▶ use deterministic sampling approach to determine boundary regions. A chunk is defined as data spanning two neighboring boundaries.
- ▶ Compute Rabin's fingerprints as the key of each fixed length sliding window
- ▶  $(\text{Key} \bmod \text{Intra-file-sampling-factor}) = 0$  indicates a chunk boundary
- ▶ Compute hash value of each chunk, e.g. SHA-1 or MD5

## ▶ Pros

- ▶ Among these deterministic sampling determined anchors, higher chance for identical ones to be found, compared to randomly chosen ones.
- ▶ Only need single-pass through the data
- ▶ Chunk size is variable

## ▶ Cons

- ▶ Be ignorant about data set characteristics
- ▶ May miss valuable boundaries for space reduction because of indistinguishably treating each key.



# Our Novel Approach

---

- ▶ **Fundamental Approach Based on Global Information**
  - ▶ keys with high frequency tend to be the boundaries
  - ▶ Missing them may degrade content based chunking significantly
  - ▶ However, identifying high frequency keys is costly
- ▶ **Proposed approach**
  - ▶ Using sampling techniques to get key frequency distribution without paying high cost (approximation vs. accurate)
  - ▶ Sampling techniques can also filter out keys of low frequency
  - ▶ Our preliminary experiments indicate this approach outperforms current algorithms both in backup data sets and daily directory data sets of different users.

# Experiment results

---

## ▶ General settings:

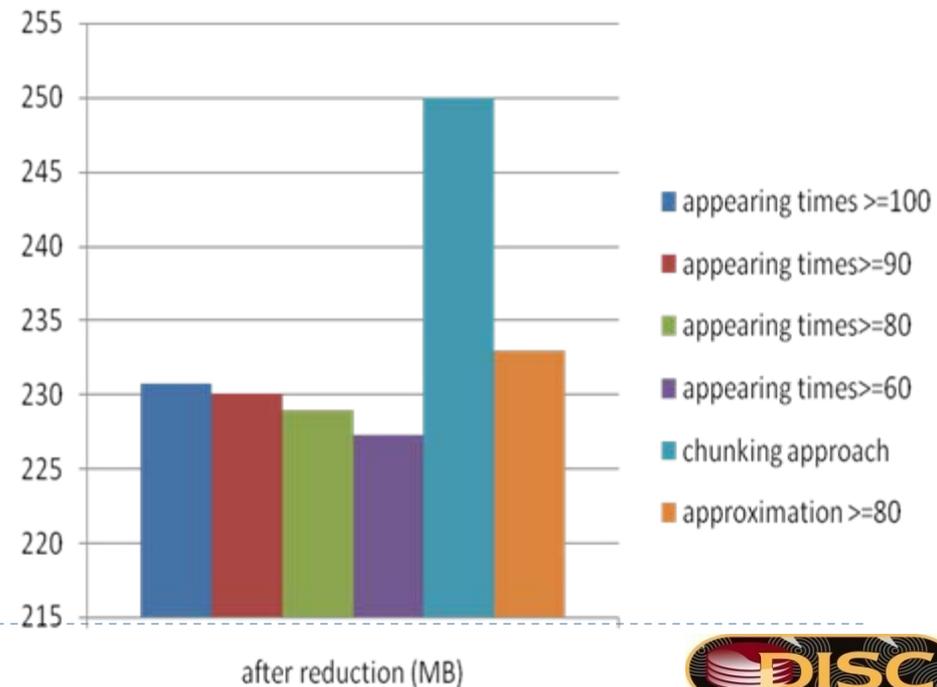
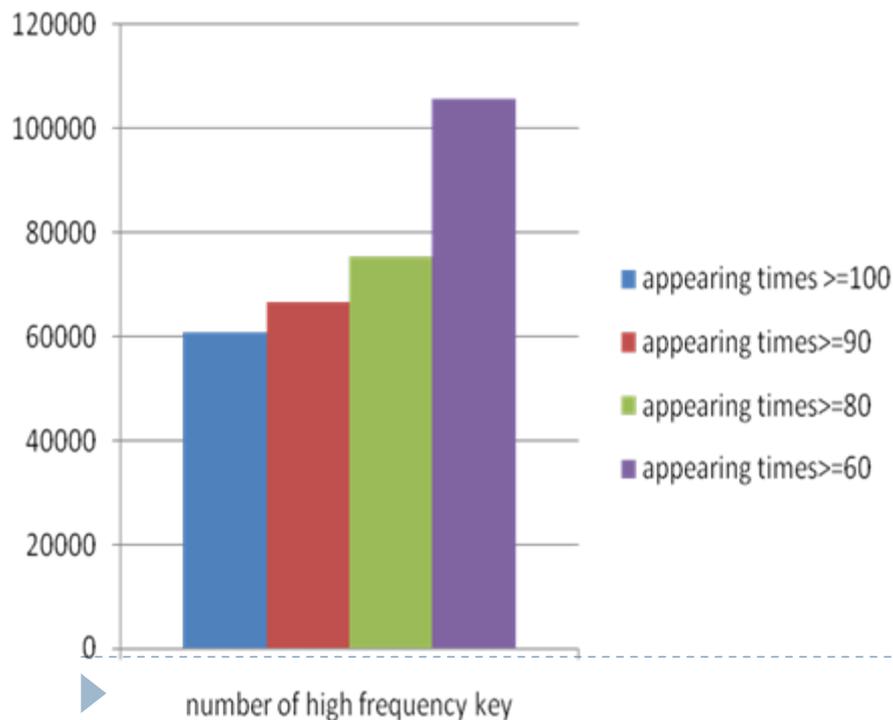
- ▶ Data set: one user's Linux home directory of our research group
- ▶ Total file number: 180
- ▶ Size: 313.4 MB
- ▶ Chunking parameters:
  - ▶ Sliding window length = 40 bytes; Intra-file sampling factor = 8192; skipping files < 2 KB; **MAX\_CHUNK\_LENGTH: UNLIMITED; MIN\_CHUNK\_LENGTH: 40 bytes**

**We compare the result of standard chunking approach to the result of our approach. In this preliminary experiment, we only treat keys as boundaries whose appearing time is above a certain threshold.**

# Experiment results

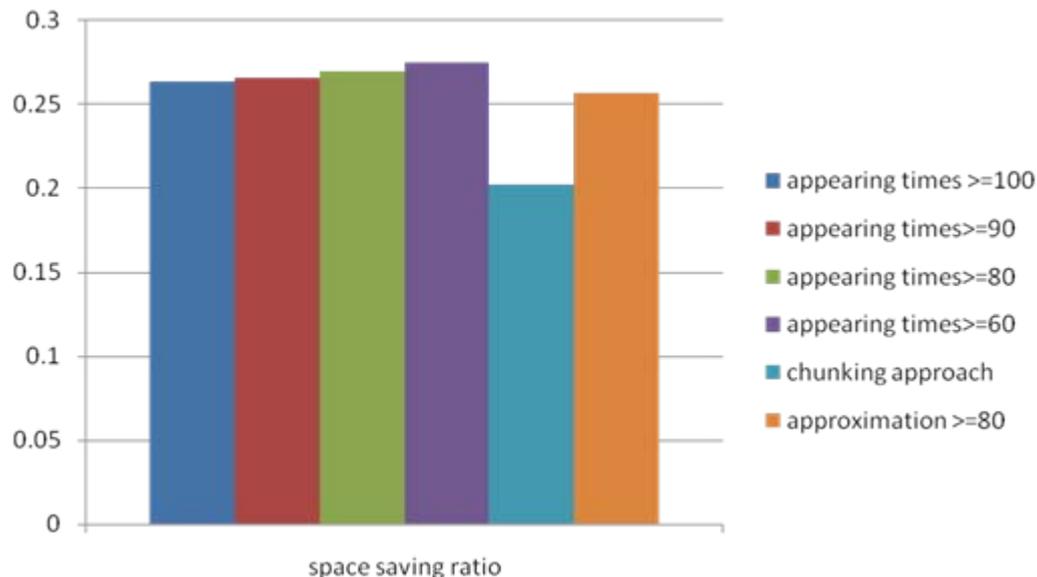
## ▶ Accurate case:

- ▶ Consider only high frequency keys that we collected
- ▶ The total number of distinct keys for this data set: 178,054,348
- Approximation: Using our sampling approach to get the frequent key

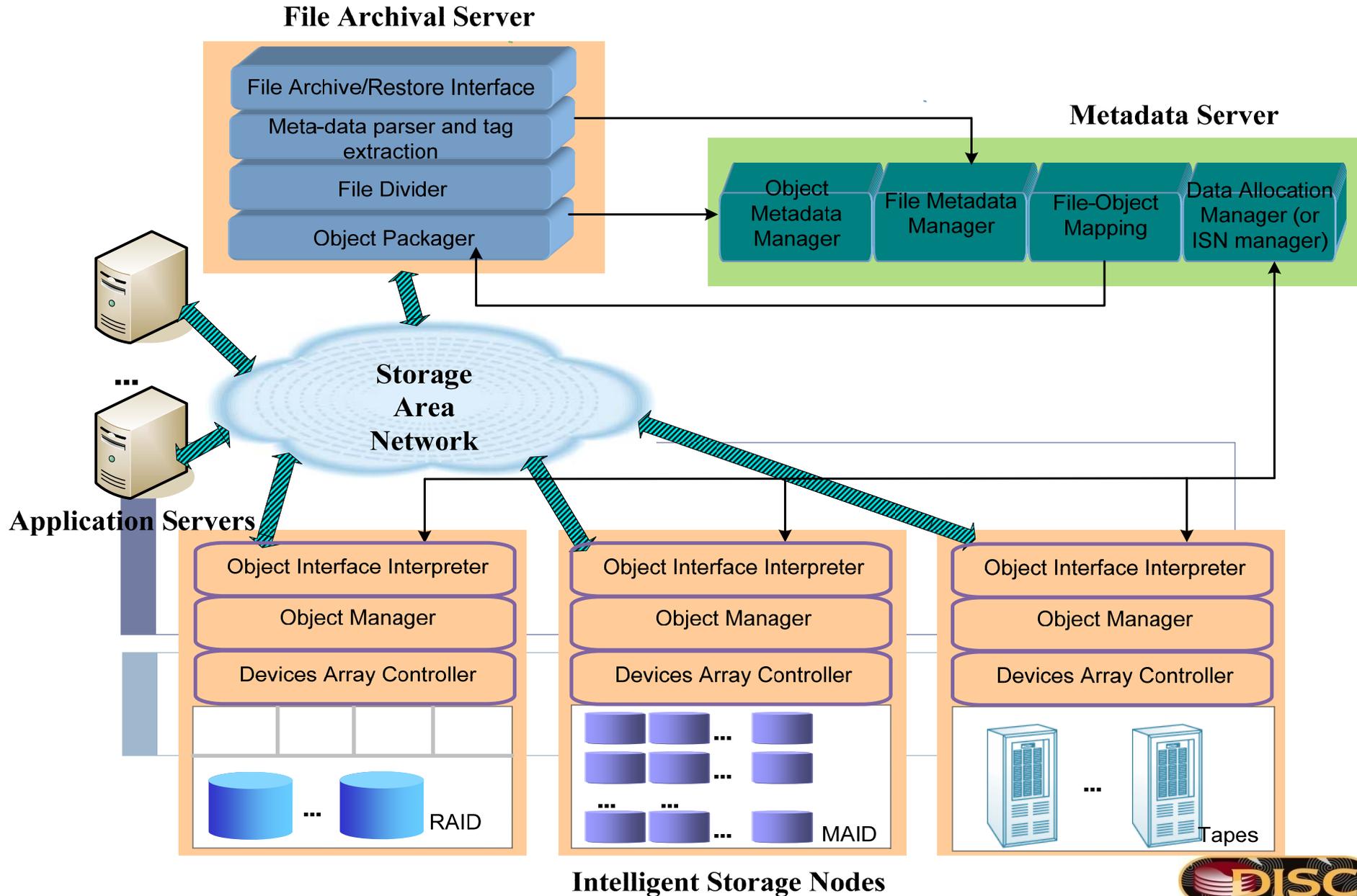


# Experiment results (cont.)

- ▶ Previous result scaled in space saving ratio
  - ▶ Space saving ratio =  $(\text{size before de-dup} - \text{size after de-dup}) / \text{size before de-dup}$

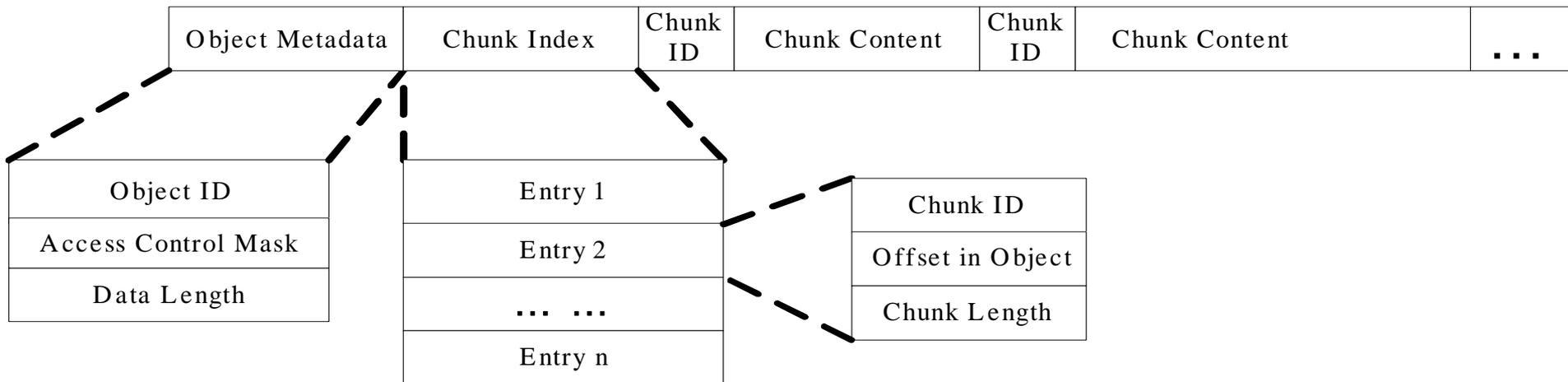


# Application-Driven Meta-Data Aware Deduplication



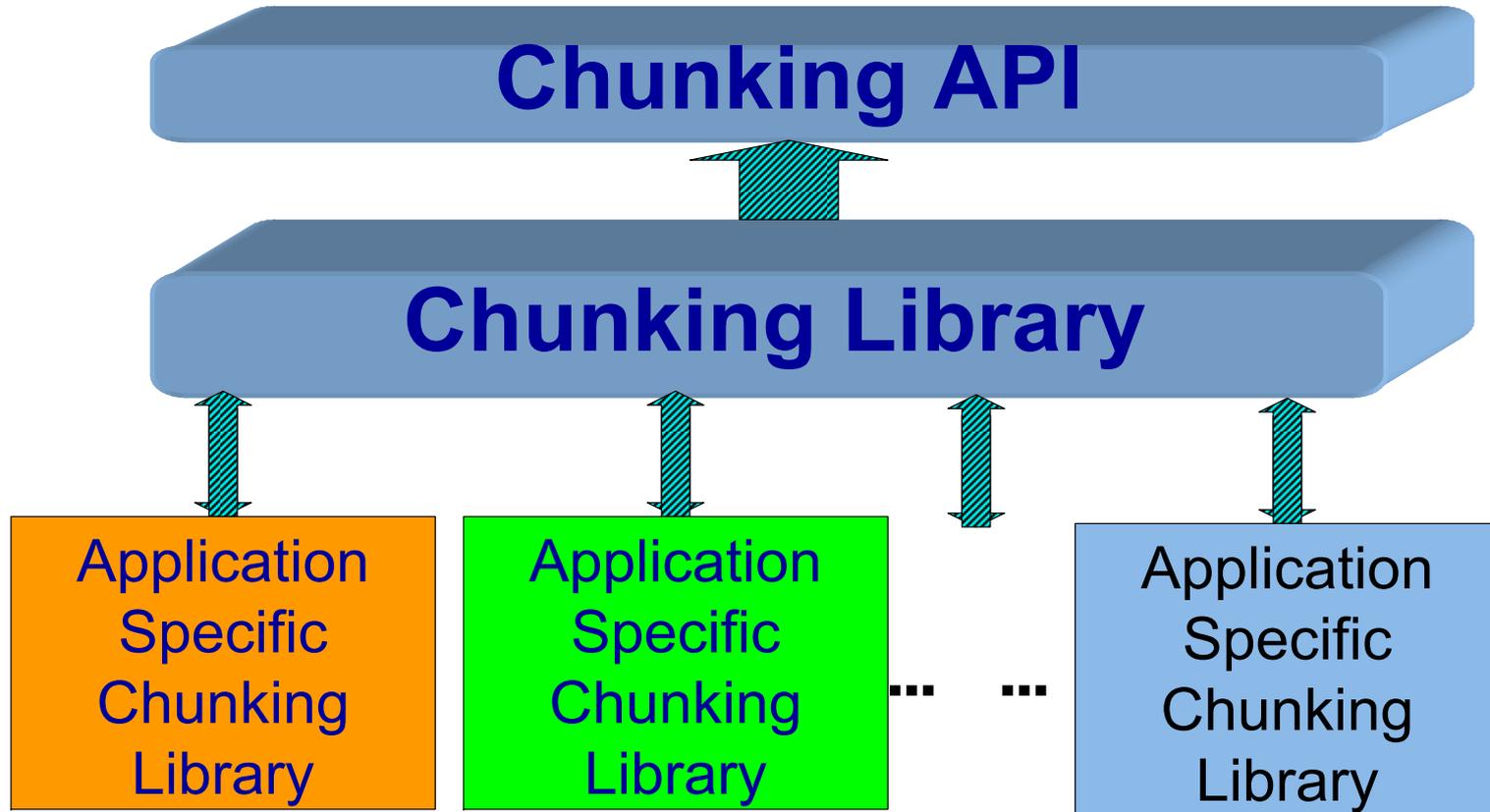
# Structure of an object stored on the Intelligent Storage Nodes (ISN)

---

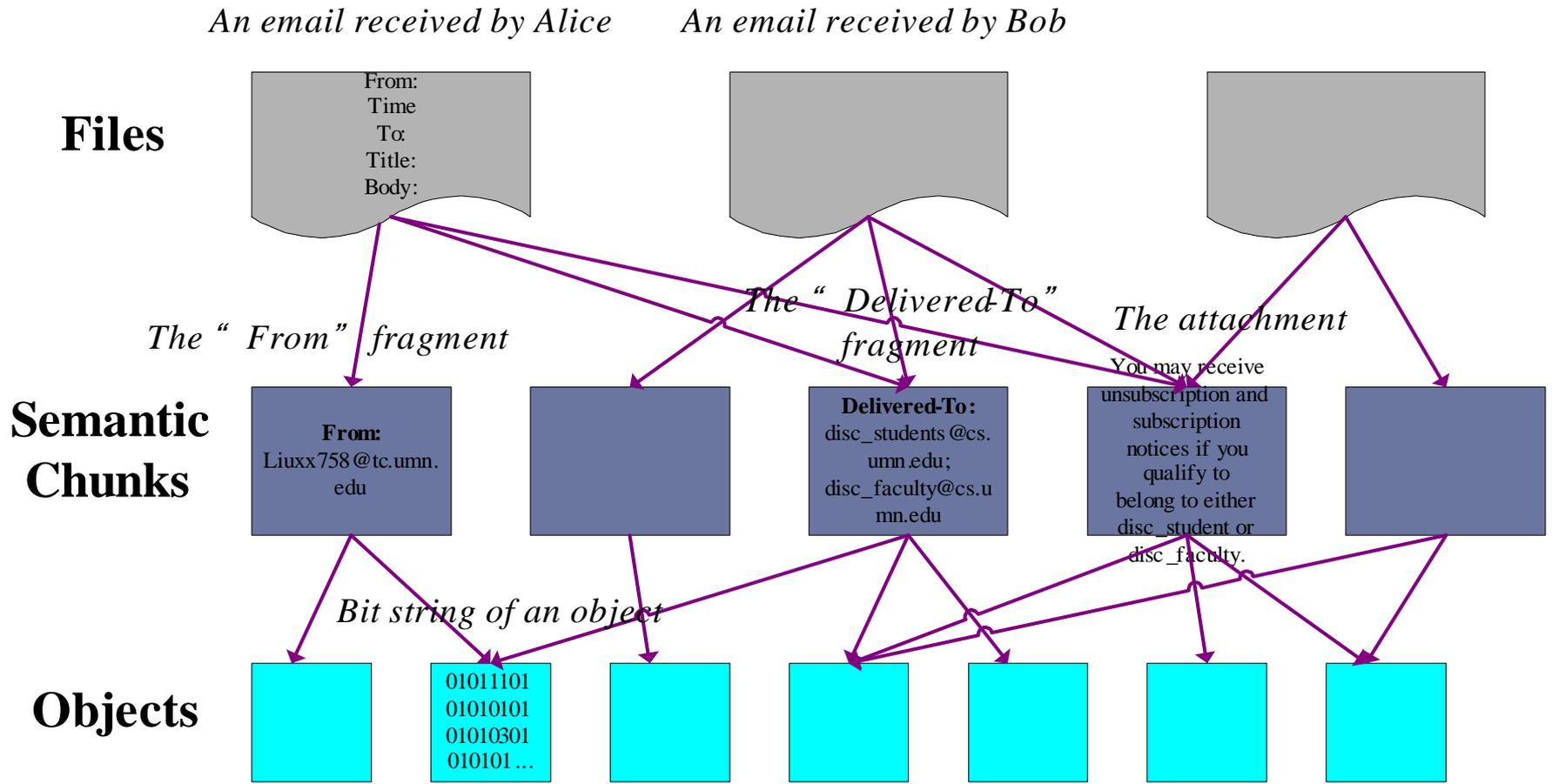


# Encapsulation of Chunking API for Different Applications

---



# Examples of Email Compression



# Comparison of Compression ratio on Email Archival system using different compress methods

---

- ▶ Original Space (Megabytes) : 161
- ▶ Robin Fingerprint : 10.3
- ▶ Gzip : 11.8
- ▶ Semantic de-duplication: 9

We have tested out four applications including email, web document, audio and flash video archival with similar results.

---

# Efficient Deduplication Algorithms Under Different Application Environments

Guanlin Lu, David Du



# Assumptions for different application environments

---

## ▶ Application Environment I

- ▶ Regarding each file as a pure bit stream, no other information available.

## ▶ Application Environment II

- ▶ With descriptive information available
  - ▶ E.g. name, type, owner, version, c/m time, annotations, etc...
- ▶ With format information available
  - ▶ E.g. MIME,HTML, XML, JPEG,MPEG, etc...



# Challenges under Environment I

---

## ▶ Two Main Challenges

- ▶ The number of files needed to compare with is large.
- ▶ The definition of similarity measure
  - ▶ Using the real delta encoding gain as similarity measure is too expensive. Therefore we need to find efficient approximate measures.
  - ▶ models of approximate similarity measure
    - Shingle-set definition (quite a few previous works based on this)
    - Fingerprint Frequency Vector (proposed)



# Related works under this environment

---

- ▶ [1] propose an approximate measure of pairwise similarity of two files used to estimate the gain of expensive real encoding.
- ▶ [2] formulate the problem of finding optimal encoding scheme for a collection of files as finding maximum branching on a direct weighted graph. It also tries to reduce the edge comparison cost of branching finding algorithm through reducing original graph into a k-nearest-neighbor graph.
- ▶ [3] propose a way to cluster near duplicated documents, although the discussion is rough and preliminary and many factors are not considered, it is very first one to indicate clustering may be promising.
  - [1]DRED: delta-encoding via resemblance detection
  - [2]Cluster-based delta compression
  - [3]Identifying and filtering near-duplicate document

# Challenge I. Analysis

---

## ▶ Analysis of the challenge

- ▶ A main high cost is the large number of comparisons among files, which can be decomposed into 2 major costs:
  - ▶ (1) Cost of a large number of pairwise comparisons in getting neighboring relations
  - ▶ (2) Cost of a large number of pairwise comparisons in getting the optimal encoding policy
- ▶ Pairwise real delta encoding is prohibitively expensive given a large number of files involved.

# Challenges under Environment II

---

- ▶ The dataset size can be much larger than what we've assumed in Environment I.
- ▶ Data classification + format based de-duplication
- ▶ Some preliminary ideas would be first classifying files based on given descriptive information, and for each class, de-duplicate using format information, some initial work of format based de-duplication is carrying on now.

# Key Management

Aaram Yun, Chunhui Shi, Nohhyun Park, Yongdae Kim,  
David Du



# 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

Revocation-efficient  
hierarchical key manager  
with limited key roll-back



- ▶ Swiss Army Knife
- ▶ Existing mechanisms satisfy one of the requirement
- ▶ Different organizations/applications have different key management requirement and file/membership dynamics
- ▶ All group keys have to be roll-back to the previous keys
- ▶ Should be able to specify the number of roll-back period



# Potential Key Management Solutions

---

- ▶ **IBE (Identity Based Encryption)**
  - ▶ ID is the public key
- ▶ **HIBE (Hierarchical IBE)**
  - ▶ ID is hierarchically structured
  - ▶ Higher IDs can decrypt ciphertexts for lower IDs
- ▶ **ABE (Attribute-Based Encryption)**
  - ▶ Instead of ID, a set of attributes is used
- ▶ **KP-ABE (Key-Policy ABE)**
  - ▶ Encryption is done w.r.t. an attribute set
  - ▶ An user with matching policy can decrypt
- ▶ **CP-ABE (Ciphertext-Policy ABE)**
  - ▶ Encryption is done w.r.t. a policy
  - ▶ An user with matching attribute set can decrypt



# Revocation for IBE

---

- ▶ Revocation is a big challenge for public key crypto
- ▶ Especially for IBE: revoking an ID?
- ▶ Epoch based solution: encrypt to (ID, time-period)
  
- ▶ Potential problems: Inconvenient for storage encryption
  - ▶ In order to have access to data encrypted with older epoch, the user has to keep all the older private keys
  - ▶ Or, has to request them again
  - ▶ More painful if fine-grained epoch required



# Revocation for ABE

---

- ▶ **Inequality based solution exists for ABE**
  - ▶ Instead of a single epoch, an inequality of form ‘epoch  $\leq$  x?’ or ‘epoch  $\geq$  x?’ can be formed inside the policy
  - ▶ Don’t have to keep many private keys
  - ▶ Applicable to both KP-ABE and CP-ABE
  
- ▶ **HIBE with epoch-based revocation?**
  - ▶ HIBE is natural for hierarchically structured organizations (e.g. Role-based Access Control)
  - ▶ Not very convenient to do epoch-based revocation
    - ▶ Higher ID could decrypt a ciphertext for lower ID, regardless of epoch



# Proposed Solution: HABE

---

- ▶ **HABE (Hierarchical ABE)**
  - ▶ A natural extension of both HIBE and ABE
  - ▶ Enables HIBE style hierarchy management
  - ▶ Enables the inequality based revocation of ABE
  
- ▶ **Current status**
  - ▶ Designed a solution
  - ▶ Analyzing security



# Theoretical Foundation of File Encryption

Aaram Yun, Yongdae Kim



# File encryption

---

- ▶ Actual content of a file is encrypted with a File Encryption Key using symmetric cryptography
- ▶ The file encryption key is controlled by users' attributes and credentials.
- ▶ Question: How to encrypt the file content?

# Previous works

---

- ▶ Many disk encryption software use CBC mode, often with a fixed IV for each chunk
  - ▶ Many don't include data authentication
- ▶ Some existing data authentication schemes emphasize minimizing the additional storage, instead of minimizing performance

# Goal

---

- ▶ Formally define the security notion for authenticated and encrypted file
- ▶ Design a secure, efficient scheme which accomplishes the security notion



# Our approach

---

- ▶ Use authenticated encryption mode of operation to encrypt and authenticate each chunk of data
- ▶ Protect the nonces used for the chunk encryption by some authentication
- ▶ Very straightforward and clean method for providing encryption and authentication



# Comparison with Hash Tree method

---

- ▶ Hash tree based method (e.x., Oprea and Reiter, USENIX Security 2007)
  - ▶ Encrypt each chunk with a nonce
  - ▶ Provide the authenticity by protecting the ciphertext by hash tree
  - ▶ Protect the nonces
- ▶ If nonces are protected, then it is not necessary to protect the whole ciphertexts by a hash tree
  - ▶ Each ciphertext of each chunk may be protected separately, instead of linking all of them together by the hash tree
  - ▶ For this role, the authenticated encryption scheme is more suitable



# Current Status

---

- ▶ Formally defined the security notion
- ▶ Have a design using authenticated encryption and universal hash-based MAC
- ▶ Proved the security of the scheme
  
- ▶ Working on details of the scheme
  - ▶ How to represent the nonces
  - ▶ How to store the authentication info
  - ▶ Which AE and MAC to choose



# Conclusions

---

- ▶ Long-term data preservation imposes many challenges
- ▶ We have also addressed a small portion of the problem
- ▶ Deduplication can be very useful
- ▶ There are rooms to improve the current depup algorithms
- ▶ Long-term key management is still not solved
- ▶ How to keep data securely and safely preserved needs more research