Generating Realistic Datasets for Deduplication Analysis

Vasily Tarasov¹, Amar Mudrankit¹, Will Buik², Philip Shilane³, Geoff Kuenning², Erez Zadok¹

¹Stony Brook University  ²Harvey Mudd College  ³EMC Corporation
Overview
Overview

Deduplication
Overview

Deduplication
Overview

Deduplication

Wait, deduplication performance depends on the data…
Everyone uses different datasets
Outline

1. Deduplication and datasets survey
2. Framework design
3. Framework implementation
4. Evaluation
5. Conclusions & future work
Outline

1. Deduplication and datasets survey
2. Framework design
3. Framework implementation
4. Evaluation
5. Conclusions & future works
Deduplication Basics

Coarse-grained **data compression** technique:

Original data: 1,000 GB
Deduplication Basics

Coarse-grained data compression technique:

Original data: 1,000 GB

Duplicates (3 of 10)
Deduplication Basics

Coarse-grained data compression technique:

Original data: 1,000 GB

Deduplicated data: 800 GB

Duplicates (3 of 10)

Mapping:
Deduplication Basics

Coarse-grained **data compression** technique:

Original data: 1,000 GB

Deduplicated data: 800 GB

20% space savings

Duplicates (3 of 10)

Mapping:
Typical Ingest Datapath

1. Chunking
   - Determine **boundaries** in the data
     - File boundaries
     - Fixed-size chunking
     - Variable chunking using Rabin fingerprints
     - More intelligent ways, e.g., content-type-aware

2. Hashing
   - Easy-to-compute chunk **identifiers**
   - Collision-resistant
   - Cryptographic
   - SHA256, MD5 (plus byte-by-byte comparison)

3. Indexing and mapping
   - Find duplicates in the index
   - Create mapping entries

---

Index:

Mapping:

Disk

0xBE 0xEF 0xBE 0xEF
Typical Ingest Datapath

1. Chunking
   - Determine **boundaries** in the data
     - File boundaries
     - Fixed-size chunking
     - Variable chunking using Rabin fingerprints
     - More intelligent ways, e.g., content-type-aware

2. Hashing
   - Easy-to-compute chunk **identifiers**
   - Collision-resistant
   - Cryptographic
   - SHA256, MD5 (plus byte-by-byte comparison)

3. Indexing and mapping
   - Find duplicates in the index
   - Create mapping entries

Index:

Mapping: \( x_1 \)
Typical Ingest Datapath

1. Chunking
   - Determine **boundaries** in the data
     - File boundaries
     - Fixed-size chunking
     - Variable chunking using Rabin fingerprints
     - More intelligent ways, e.g., content-type-aware

2. Hashing
   - Easy-to-compute chunk **identifiers**
   - Collision-resistant
   - Cryptographic
   - SHA256, MD5 (plus byte-by-byte comparison)

3. Indexing and mapping
   - Find duplicates in the index
   - Create mapping entries

Index:

Mapping:

Disk
Typical Ingest Datapath

1. Chunking
   - Determine **boundaries** in the data
     - File boundaries
     - Fixed-size chunking
     - Variable chunking using Rabin fingerprints
     - More intelligent ways, e.g., content-type-aware

2. Hashing
   - Easy-to-compute chunk **identifiers**
   - Collision-resistant
   - Cryptographic
   - SHA256, MD5 (plus byte-by-byte comparison)

3. Indexing and mapping
   - Find duplicates in the index
   - Create mapping entries

Disk

Index:

Mapping: $x_1 \ x_2 \ x_3$
Typical Ingest Datapath

1. Chunking
   ◆ Determine **boundaries** in the data
     ▪ File boundaries
     ▪ Fixed-size chunking
     ▪ Variable chunking using Rabin fingerprints
     ▪ More intelligent ways, e.g., content-type-aware

2. Hashing
   ◆ Easy-to-compute chunk **identifiers**
   ◆ Collision-resistant
   ◆ Cryptographic
   ◆ SHA256, MD5 (plus byte-by-byte comparison)

3. Indexing and mapping
   ◆ Find duplicates in the index
   ◆ Create mapping entries

---

Index:

```
X_1  X_2  X_3  X_4
```

Mapping:

```
X_1  X_2  X_3  X_4
```

Disk

---

EMC²
How Efficient is Deduplication?

- High deduplication ratios for certain datasets
  - Backups: $15 \times$ [Wallace2012]
  - Virtual machine images: $7 \times$ [Smith2008]
  - Multi-tenant shared storage: $3 \times$ [Meyer2011]
- Fastest growing segment of storage industry [NetApp/IDC 2011]
  - $80\%+$ of corporations are exploring deduplication [IDC 2011]
- Challenges:
  - Performance
    - Computationally intensive
    - Out of RAM indexes
    - Fragmentation
  - Manageability, reliability
How Efficient is Deduplication?

- High deduplication ratios for certain datasets
  - Backups: \(15\times\) [Wallace2012]
  - Virtual machine images: \(7\times\) [Smith2008]
  - Multi-tenant shared storage: \(3\times\) [Meyer2011]
- Fastest growing segment of storage industry [NetApp/IDC 2011]
  - \(80\%+\) of corporations are exploring deduplication [IDC 2011]
- Challenges:
  - Performance
    - Computationally intensive
    - Out of RAM indexes
    - Fragmentation
  - Manageability, reliability

Variety of optimizations:
- Intelligent caching
- Intelligent prefetching
- Bloom filters
- Content-aware chunking
- ...
Survey of Datasets

- 33 research papers
- FAST, ATC, MSST, SYSTOR conferences
- 120 datasets
Survey of Datasets

- 33 research papers
- FAST, ATC, MSST, SYSTOR conferences
- 120 datasets

- 14% Synthetic, simplistic
- 7% Public, less than 1GB
- 9% OS Images
- 53% Private
- 14% Public, but hardly possible to reproduce
Survey of Datasets

- **33** research papers
- **FAST, ATC, MSST, SYSTOR** conferences
- **120** datasets

![Pie chart showing the distribution of datasets](chart.png)

- **53%** Private
- **17%** Public, less than 1GB
- **14%** Public, but hardly possible to reproduce
- **9%** Synthetic, simplistic
- **7%** OS Images
- **3%** Research papers
Survey of Datasets

- 33 research papers
- FAST, ATC, MSST, SYSTOR conferences
- 120 datasets

- Private
- Public, less than 1GB
- Synthetic, simplistic
- Public, but hardly possible to reproduce
- OS Images

- 53% Private
- 17% Public, less than 1GB
- 14% Synthetic, simplistic
- 9% Public, but hardly possible to reproduce
- 7% OS Images
Survey of Datasets

- 33 research papers
- FAST, ATC, MSST, SYSTOR conferences
- 120 datasets

- 53% Private
- 17% Public, less than 1GB
- 14% Public, but hardly possible to reproduce
- 9% Synthetic, simplistic
- 7% OS Images
Survey of Datasets

- 33 research papers
- FAST, ATC, MSST, SYSTOR conferences
- 120 datasets

Private

Public, less than 1GB

Public, but hardly possible to reproduce

OS Images

Synthetic, simplistic

17%
9%
7%
14%
53%
Survey of Datasets

- 33 research papers
- FAST, ATC, MSST, SYSTOR conferences
- 120 datasets

- Synthetic, simplistic
- Public, less than 1GB
- Public, but hardly possible to reproduce
- Private
- OS Images

53% 17% 14% 7% 9%
Survey of Datasets

- 33 research papers
- FAST, ATC, MSST, SYSTOR conferences
- 120 datasets

- 53% Private
- 17% Public, less than 1GB
- 14% Public, but hardly possible to reproduce
- 9% Synthetic, simplistic
- 7% OS Images
The Need for Datasets

To perform true, versatile, and easy evaluation and comparison of deduplication systems

Requirements:

- Accessible
- Realistic
- Sufficiently large
- Easy to distribute
- With controllable characteristics

Generate datasets
Outline

1. Deduplication and datasets survey
2. **Framework design**
3. Framework implementation
4. Evaluation
5. Conclusions & future works
Insight

Complete Dataset

File System Mutation

Original FS Tree

Updated FS Tree

Complete Dataset

User 2

changes

time

changes

User 3

changes

time

changes

changes

time

changes
Framework Objects

- **FSTREE**
  - Per-snapshot
  - In-memory

- **DIRECTORY**

- **FILE**

- **CHUNK**
  - Chunk identifiers
    - Roughly correspond to hashes
  - Per-file lists of chunks
Framework Objects

- **FSTREE**
  - Per-snapshot
  - In-memory
- **DIRECTORY**
- **FILE**
- **CHUNK**
  - Chunk identifiers
    - Roughly correspond to hashes
  - Per-file lists of chunks

- **2TB snapshots**, 16KB file size, 10 files/directory
- **9GB RAM**
Framework Objects

- **FSTREE**
  - Per-snapshot
  - In-memory

- **DIRECTORY**

- **FILE**

- **CHUNK**
  - Chunk identifiers
    - Roughly correspond to hashes
  - Per-file lists of chunks

---

- **2TB snapshots**, 16KB file size, 10 files/directory
  - 9GB RAM

- **14TB snapshots**
  - 64GB RAM
Framework Objects

- **FSTREE**
  - Per-snapshot
  - In-memory

- **DIRECTORY**

- **FILE**

- **CHUNK**
  - Chunk identifiers
    - Roughly correspond
  - Per-file lists of chunks

**Statistics**

- 2TB **snapshots**, 9GB RAM
  - 16KB file size, 10 files/directory

- 14TB **snapshots**, 64GB RAM

- 71 snapshots
  - Total: **1PB** dataset
  - still 64GB of RAM
Mutation Conveyor

profile

FSTREE

FS-MUTATE

FS-CREATE

profile

FSTREE

FS-MUTATE

FS-CREATE

profile

FSTREE

FS-MUTATE

FS-CREATE

profile

FSTREE

FS-MUTATE

FS-CREATE
Mutation Conveyor

profile

FSTREE

FS-MUTATE

profile

FSTREE

FS-MUTATE

profile

FSTREE

FS-MUTATE

FS-CREATE

FS-CREATE

FS-CREATE

FS-CREATE
Mutation Conveyor

profile

FSTREE

FS-MUTATE

FSTREE

FS-MUTATE

FSTREE

FS-MUTATE

profile

profile

On-disk file system - Tar-like files - Incremental backups -

FS-CREATE

FS-CREATE

FS-CREATE
Initial File System Tree

/some/file/system

FS-SCAN ➔ FSTREE

FS-PHDUPLICATE ➔ empty FSTREE

Content profile ➔ Meta profile ➔ FS-IMPRESSION

FS-PHROFILE ➔ /some/file/system

/some/file/system
Initial File System Tree

Using existing file system

/some/file/system

FS-SCAN ➔ FSTREE

FS-POPULATE ➔ empty FSTREE

Content profile ➔ Meta profile ➔ FS-IMPRESSION

/some/file/system

FS-PROFILE ➔
Initial File System Tree

/some/file/system

Using profile

/some/file/system

FS-SCAN ➔ FSTREE ➔ FS-POPULATE ➔ FS-IMPRESSION

Content profile ➔ Meta profile

empty FSTREE

FS-PHILE

EMC²

06/14/2012 Generating Realistic Datasets for Deduplication Analysis – ATC 2012
Initial File System Tree

/some/file/system

Using profile

Using profile

/some/file/system
Initial File System Tree

Using profile

/some/file/system

FS-SCAN → FSTREE

FS-PROFILE → Content profile
Meta profile → FS-IMPRESSION

empty FSTREE

/some/file/system
Initial File System Tree

/some/file/system

Using profile

/some/file/system
Outline

1. Deduplication and datasets survey
2. Framework design
3. Framework implementation
4. Evaluation
5. Conclusions & future work
Main Idea

- **Consecutive snapshots** of real datasets
  - Scan live file systems periodically
  - Use existing backup data
  - Software or data releases (e.g., Linux kernels)

- Observe **statistics of changes**
  - Markov Model
  - Multi-dimensional distribution
Markov Model: File States

- New
- Modified
- Unmodified
- Deleted
Markov Model: File States

New -> New (p(N))
New -> Deleted (p(DN))
New -> Modified (p(MM))
New -> Unmodified (p(UU))

Deleted -> New (p(ND))
Deleted -> Deleted (p(D))
Deleted -> Modified (p(MD))
Deleted -> Unmodified (p(DN))

Modified -> New (p(NM))
Modified -> Deleted (p(D))
Modified -> Modified (p(MM))
Modified -> Unmodified (p(UU))

Unmodified -> New (p(NU))
Unmodified -> Deleted (p(D))
Unmodified -> Modified (p(MU))
Unmodified -> Unmodified (p(UU))

p(ND) p(NM) p(N) p(NU) p(ND) p(NM) p(N) p(NU)
Markov Model: File States

- **New**
- **Modified**
- **Unmodified**
- **Deleted**

Transition Probabilities:
- \( p(N) \)
- \( p(ND) \)
- \( p(NM) \)
- \( p(NU) \)
- \( p(MM) \)
- \( p(MD) \)
- \( p(MU) \)
- \( p(UM) \)
- \( p(UU) \)
- \( p(NU) \)
- \( p(UD) \)
- \( p(DN) \)
- \( p(D) \)

- **10 files modified**
- **8 files modified**

**\( P(MM) = 80\% \)**
Markov Model: File States

Home
Directories

Modified

Unmodif.

New

Deleted

P(MM) = 80%

10 files modified

8 files modified
Markov Model: File States

- New
- Modified
- Unmodified
- Deleted

Transition Probabilities:
- \( p(N) = 4\% \)
- \( p(ND) = 2\% \)
- \( p(NM) = 20\% \)
- \( p(MM) = 36\% \)
- \( p(MD) = 2\% \)
- \( p(MU) = 4\% \)
- \( p(UM) = 10\% \)
- \( p(UU) = 54\% \)
- \( p(NU) = 6\% \)
- \( p(UD) = 99.51\% \)
- \( p(DN) = 0.14\% \)
- \( p(D) = 0.35\% \)

10 files modified
8 files modified

\[ P(MM) = 80\% \]
Markov Model: File States

New

Modified

Unmodif.

Deleted

Home Directories

10 files modified

8 files modified

P(MM) = 80%
Multi-dimensional Distribution: **New Files**

**Dimensions:**
- Directory Depth
- File Extension
- Size (in chunks)
- Unique chunks
- Chunks with 1 duplicate
- Chunks with 2 duplicates

Compared to previous snapshot
Multi-dimensional Distribution: New Files

Dimensions:
- Directory Depth
- File Extension
- Size (in chunks)
- Unique chunks
- Chunks with 1 duplicate
- Chunks with 2 duplicates

$M_{\text{new}}(\text{depth, ext, size\_chunks, uniq, dup1, dup2})$

$M_{\text{new}}$: number of new files with corresponding properties

Example: $M_{\text{new}}(2, \text{".c"}, 7, 3, 2, 2) = 10$
Outline

1. Deduplication and datasets survey
2. Framework design
3. Framework implementation
4. Evaluation
5. Conclusions & future work
## Datasets

<table>
<thead>
<tr>
<th>Name</th>
<th>Snapshot number</th>
<th>Files per snapshot (1000)</th>
<th>Snapshot size (GB)</th>
<th>Total Files (1000)</th>
<th>Total Size (GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kernels (2.6.*)</td>
<td>40</td>
<td>23</td>
<td>0.3</td>
<td>903</td>
<td>13</td>
</tr>
<tr>
<td>CentOS (5.*)</td>
<td>8</td>
<td>195</td>
<td>4.5</td>
<td>1,559</td>
<td>36</td>
</tr>
<tr>
<td>Home</td>
<td>15 weekly</td>
<td>1,023</td>
<td>227</td>
<td>15,352</td>
<td>3,482</td>
</tr>
<tr>
<td>MacOS</td>
<td>71 daily</td>
<td>1,173</td>
<td>59</td>
<td>83,220</td>
<td>4,080</td>
</tr>
<tr>
<td>System Logs</td>
<td>8 weekly</td>
<td>334</td>
<td>78</td>
<td>2,672</td>
<td>626</td>
</tr>
<tr>
<td>Sources</td>
<td>8 weekly</td>
<td>139</td>
<td>162</td>
<td>1,112</td>
<td>1,331</td>
</tr>
</tbody>
</table>
## Datasets

<table>
<thead>
<tr>
<th>Name</th>
<th>Snapshot number</th>
<th>Files per snapshot (1000)</th>
<th>Snapshot size (GB)</th>
<th>Total Files (1000)</th>
<th>Total Size (GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kernels (2.6.*)</td>
<td>40</td>
<td>23</td>
<td>0.3</td>
<td>903</td>
<td>13</td>
</tr>
<tr>
<td>CentOS (5.*)</td>
<td>8</td>
<td>195</td>
<td>4.5</td>
<td>1,559</td>
<td>36</td>
</tr>
<tr>
<td>Home</td>
<td>15 weekly</td>
<td>1,023</td>
<td>227</td>
<td>15,352</td>
<td>3,482</td>
</tr>
<tr>
<td>MacOS</td>
<td>71 daily</td>
<td>1,173</td>
<td>59</td>
<td>83,220</td>
<td>4,080</td>
</tr>
<tr>
<td>System Logs</td>
<td>8 weekly</td>
<td>334</td>
<td>78</td>
<td>2,672</td>
<td>626</td>
</tr>
<tr>
<td>Sources</td>
<td>8 weekly</td>
<td>139</td>
<td>162</td>
<td>1,112</td>
<td>1,331</td>
</tr>
</tbody>
</table>
## Datasets

<table>
<thead>
<tr>
<th>Name</th>
<th>Snapshot number</th>
<th>Files per snapshot (1000)</th>
<th>Snapshot size (GB)</th>
<th>Total Files (1000)</th>
<th>Total Size (GB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kernels (2.6.*)</td>
<td>40</td>
<td>23</td>
<td>0.3</td>
<td>903</td>
<td>13</td>
</tr>
<tr>
<td>CentOS (5.*)</td>
<td>8</td>
<td>195</td>
<td>4.5</td>
<td>1,559</td>
<td>36</td>
</tr>
<tr>
<td>Home</td>
<td>15 weekly</td>
<td>1,023</td>
<td>227</td>
<td>15,352</td>
<td>3,482</td>
</tr>
<tr>
<td>MacOS</td>
<td>71 daily</td>
<td>1,173</td>
<td>59</td>
<td>83,220</td>
<td>4,080</td>
</tr>
<tr>
<td>System Logs</td>
<td>8 weekly</td>
<td>334</td>
<td>78</td>
<td>2,672</td>
<td>626</td>
</tr>
<tr>
<td>Sources</td>
<td>8 weekly</td>
<td>139</td>
<td>162</td>
<td>1,112</td>
<td>1,331</td>
</tr>
</tbody>
</table>
Methodology

1. Scanned all snapshots
2. Created mutation profiles
3. Generated synthetic datasets with FS-MUTATE
4. Evaluated the error between emulated parameters and real parameters
File Number and Total Size

Kernels in the dataset (2.6.0 - 2.6.39)

- Files (thousands)
- Chunks (thousands)
File Number and Total Size

Error:
2% on average
6% maximum (for all datasets)
Chunk Duplicates: MacOS

Unique chunks

Chunks with 1 duplicate

Chunks with 2 duplicates
Chunk Duplicates: MacOS

Unique chunks

Chunks with 1 duplicate

Chunks with 2 duplicates

Error: 9% on average 15% maximum (for all datasets)
Chunk Duplicates: MacOS

Unique chunks

Chunks with 1 duplicate

Chunks with 2 duplicates

Error: 9% on average 15% maximum (for all datasets)

Profile sizes: 200,000× smaller!

6/14/2012

Generating Realistic Datasets for Deduplication Analysis – ATC 2012
Performance vs. Size

- Kernel
- CentOS
- System Logs
- Sources
- MacOS
- Homes

Mutation Time (minutes) vs. Dataset Size (TB)

Intel Xeon 3.3GHz
64GB of RAM
Outline

1. Deduplication and datasets survey
2. Framework design
3. Framework implementation
4. Evaluation
5. Conclusions & future work
Summary

- First technique for generating deduplication datasets
  - Realistic properties
  - Fair comparison of different deduplication techniques
  - Sharing among researchers
- Emulate file system changes
- Generic framework
  - Flexible, versatile, and extensible
- Statistical implementation
  - Markov Model
  - Multi-dimensional statistics
- High accuracy, small model size, high performance
Future work

- Study existing deduplication systems
- Create exhaustive list of parameters
  - E.g., local chunk compression control
- Initial file system generation
  - Convenient usage
- Detect global trend lines
  - Statistical clustering
- Analyze other datasets
Generating Realistic Datasets for Deduplication Analysis

Q&A

Download sources and profiles:
https://avatar.fsl.cs.sunysb.edu/groups/deduplicationpublic/

Vasily Tarasov¹, Amar Mudrankit¹, Will Buik², Philip Shilane³, Geoff Kuenning², Erez Zadok¹

¹Stony Brook University  ²Harvey Mudd College  ³EMC Corporation