InftyDedup: Scalable and Cost-Effective Cloud Tiering with Deduplication

Iwona Kotlarska, Andrzej Jackowski, Krzysztof Lichota, Michal Welnicki, Cezary Dubnicki, Konrad Iwanicki

9LivesData

UNIVERSITY OF WARSAW

USENIX FAST ‘23
Who we are?

► 9LivesData
- R&D company based in Warsaw, Poland
- Specializing in storage, distributed systems, cloud computing
- Provides services outsourced on a contract basis to clients from USA, Japan, EMEA

► Selected publications
- IEEE TPDS: ObjDedup: High-Throughput Object Storage Layer for Backup Systems with Block-Level Deduplication (accepted Feb '23)
- SYSTOR ‘15: Reducing fragmentation impact with forward knowledge in backup systems with deduplication
- FAST ‘13: Concurrent Deletion in a Distributed Content-Addressable Storage System with Global Deduplication
- FAST ‘09: HYDRAstor: A Scalable Secondary Storage
- And others in FAST, SYSTOR, ACM ToS, SRDS, HPDC...
Motivation for InftyDedup

- Tiering to cloud becoming popular for backup data
  - Duggal et al. ATC ‘19 (Data Domain Cloud Tier)
  - Other backup appliances and backup applications

- Cost effective storage for long term retention of older backups
  - Low probability of massive restore from cloud

- Our goal - exploit cloud capabilities to provide:
  - Limitless scalability
  - Major cost reductions

- NEC HYDRAstor as local tier
How Cloud Tiering with Deduplication is usually done?

Local Tier

File_A  File_B  File_C
How Cloud Tiering with Deduplication is usually done?

Local Tier

Cloud Tier
How Cloud Tiering with Deduplication is usually done?

Local Tier

Cloud Tier

DEDUPLICATED BLOCKS

File_A File_B File_C

File_Y File_Z File_C
Benefits of Cloud Tiering with Deduplication

+ Decreased cloud storage costs
+ Decreased network traffic between tiers
+ Data available in the cloud when local tier fails
Limitations of Existing Solutions

- **Local** tier processes metadata of cloud data
  - Local resources consumed for computations
  - Cloud tier size limited by local resources
  - No deduplication between multiple local tier systems
Limitations of Existing Solutions

- Unexploited potential of cost reduction
  - Lack of algorithms for mixing *hot* and *cold* cloud storage
  - No utilization of affordable in-cloud computing (e.g. *spot instances*)
InftyDedup Architecture

Local Tier

System A

System B
InftyDedup Architecture: Deduplication

Local Tier

System A

System B

Cloud Tier

Cloud Data & Metadata

Deduplication Batch Processing

File_Y

File_Z

META DATA

DEDUPLICATED BLOCKS

META DATA
InftyDedup Architecture: Restores

System A

System B

Local Tier

Cloud Tier

RESTORE TO LOCAL TIER

INDEPENDENT RESTORE

RESTORE TARGET

File_Y

File_Z

Cloud Data & Metadata
InftyDedup Architecture: Garbage Collection

Cloud Tier

Cloud Data & Metadata

Garbage Collection
Batch Processing

System A

System B

Local Tier
Deduplication Flow

Typical Use Case
Deduplication Flow: Filerecipes Upload

Cloud Tier

Local Tier

File_A
File_B
File_C

DEDUPLICATED BLOCKS

File_D
File_E
File_F

DEDUPLICATED BLOCKS

File_C
File_F

FILE RECIPES (METADATA)
Deduplication Flow: BatchDedup

Cloud Tier

Local Tier
Deduplication Flow: Metadata Download

FILE RECIPES (METADATA)

BLOCKS TO UPLOAD (METADATA)

Local Tier

Cloud Tier

DEDUPPLICATED BLOCKS

DEDUPLICATED BLOCKS
Deduplication Flow: Data Upload

Local Tier

Cloud Tier

DEDUPLICATED DATA BLOCKS

FILE RECIPES (METADATA)
Deduplication Flow: Finished Upload

Local Tier

Cloud Tier

USENIX FAST ‘23
Deduplication Batch Processing: BatchDedup

► Distributed algorithm for duplicate elimination
  - Spin-up many *cloud instances*
► Highly efficient batch processing
► Executed occasionally (e.g. once a week)
► Spot instances can be used
► Processes multi-petabyte data sets for a couple of dollars
Why Batch Processing?

- Processing fingerprints in batches is efficient
- Tiering to cloud is lengthy anyway
  - Upload can take up hours/days
  - Reclaiming space locally requires local garbage collection
- Typical use case: tiering of older backups
  - Life-cycle of files known in advance from backup policy
BatchDedup is scalable

![Bar chart showing wall clock time for different instance counts and data sizes.](chart.png)

- 8 instances: 156 TB
- 16 instances: 313 TB
- 32 instances: 626 TB
- 64 instances: 1.25 PB

Wall Clock Time [s]

USENIX FAST ‘23
Our in-cloud structures

- Data (blocks) are kept in containers
  - Example: 1000 blocks per container
  - Decreases PUT / GET request costs
Our in-cloud structures

► Most important metadata structures:
  - **Fingerprint Index** lists fingerprints of blocks in cloud
  - **File Recipes** references blocks of each file

► Fingerprint Index much smaller than sum of file recipes
Garbage Collection: BatchGC

- Finds blocks no longer referenced by any file recipe
- Distributed computation similar to BatchDedup
- Consumes more resources than BatchDedup
  - Processes file recipes of all files stored in the cloud
- Similar running time to other algorithms [Strzelczak et al. FAST '13; Douglis et al. FAST '17]
- Executed less frequently than BatchDedup
What to do with partially empty containers?

- After GC containers have both live and dead blocks
- Rewriting containers is non-free
- Backup expiration known upfront based on backup policy
  - WORM protection guarantees no early deletes
- Our batch algorithms extended to process per-block expiration dates
  - We only rewrite containers when profitable
## Clouds Offer Multiple Classes of Storage

<table>
<thead>
<tr>
<th>Hot Storage</th>
<th>Cold Storage</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ Cheaper PUT/GET requests</td>
<td>+ Lower GB/month costs (e.g., 5.25 times)</td>
</tr>
<tr>
<td>+ No minimal storage period</td>
<td>– Minimal storage period (e.g., 90-365 days)</td>
</tr>
<tr>
<td>+ No transfer fees (other than egress traffic)</td>
<td>– Additional transfer fees</td>
</tr>
<tr>
<td>– Higher GB/month costs</td>
<td>– More expensive requests (e.g., 25 times)</td>
</tr>
</tbody>
</table>

Many cold storage services offer the same **millisecond latency** as hot storage.
Mixing Storage Classes in InftyDedup

- Batch algorithms extended to chose between hot and cold storage
Mixing Storage Classes in InftyDedup

- Cold storage introduces minimal storage duration
  + We know expiration dates

- More expensive PUT requests
  + Data kept in containers

- More expensive GET requests and additional transfer fees

💡 Try to predict restore frequency of each block:
  - Administrator estimates restore frequency of backups
  - InftyDedup propagates the information to blocks
  - Heuristics forecast future references of block
Mixing Storage Classes Benefits

Almost no reads

Some reads

Many reads
Summary of InftyDedup

► Novel cloud-native architecture for tiering with deduplication
► Deduplication processed entirely in cloud
  - Deduplicates data of multiple local-tier systems
  - No scaling limit
► Deduplication of petabytes for a couple of dollars
► Further cost reductions by mixing hot and cold storage

Thank You!

jackowski@9livesdata.com