Reducing Replication Bandwidth for Distributed Document Databases

Lianghong Xu¹, Andy Pavlo¹, Sudipta Sengupta²
Jin Li², Greg Ganger¹
Carnegie Mellon University¹, Microsoft Research²
Document-oriented Databases

{  
  "_id" : '55ca4cf7bad4f75b8eb5c25c',  
  "pageId" : "46780",  
  "revId" : "41173",  
  "timestamp" : '2002-03-30T20:06:22',  
  "sha1" : "6i81h1zt22ulw4stxoofyzmxd",  
  "text" : "The Peer and the Peri is a comic [[Gilbert and Sullivan]] [[operetta ]] in two acts... just as predicting,...The fairy Queen, however, appears to ... all live happily ever after."
}

Update: Reading a recent doc and writing back a similar one

{  
  "_id" : '55ca4cf7bad4f75b8eb5c25d',  
  "pageId" : "46780",  
  "revId" : "128520",  
  "timestamp" : '2002-03-30T20:11:12',  
  "sha1" : "q08x58kbjmylj4bow3e903uz",  
  "text" : "The Peer and the Peri is a comic [[Gilbert and Sullivan]] [[operetta ]] in two acts... just as predicted, ...The fairy Queen, on the other hand, is "not" happy, and appears to ... all live happily ever after."
}
Replication Bandwidth

Goal: Reduce bandwidth for WAN geo-replication

The Peer and the Peri is a comic [[Gilbert and Sullivan]] [[operetta]] in two acts... just as predicted, ...The fairy Queen, on the other hand, is "not" happy, and appears... all live happily ever after. "

{ "_id" : "55ca4cf7bad4f75b8eb5c25d", "pageId" : "46780", "revId" : "128520", "timestamp" : "2002-03-30T20:11:12Z", "sha1" : "q08x58kbjmyljj4bow3e903uz", "text" : "The Peer and the Peri is a comic [[Gilbert and Sullivan]] [[operetta]] in two acts... just as predicted, ...The fairy Queen, on the other hand, is "not" happy, and appears... all live happily ever after. " }

{ "_id" : "55ca4cf7bad4f75b8eb5c25c", "pageId" : "46780", "revId" : "41173", "timestamp" : "2002-03-30T20:06:22Z", "sha1" : "6i81h1zt22u1w4sfxoofyzmxd", "text" : "The Peer and the Peri is a comic [[Gilbert and Sullivan]] [[operetta]] in two acts... just as predicting, ...The fairy Queen, however, appears to ... all live happily ever after. " }
Why Deduplication?

- Why not just **compress**?
  - Oplog batches are small and not enough overlap

- Why not just use **diff**?
  - Need application guidance to identify source

- **Dedup** finds and removes redundancies
  - *In the entire data corpus*
Traditional Dedup: Ideal

Incoming Data

{BYTE STREAM}

Duplicate Region

Modified Region

Chunk Boundary

Deduped Data

Send dedup’ed data to replicas
Traditional Dedup: Reality

Incoming Data

Chunk Boundary | Modified Region | Duplicate Region

Deduped Data

Send almost the entire document.
Similarity Dedup

Incoming Data

Dedup’ed Data

Chunk Boundary

Modified Region

Duplicate Region

Only send delta encoding.
Compress vs. Dedup

20GB sampled Wikipedia dataset
MongoDB v2.7 // 4MB Oplog batches
sDedup: Similarity Dedup

**Client**

**Database**

**Source documents**

**sDedup Encoder**

**Dedup’ed oplog entries**

**Oplog**

**Unsynchronized oplog entries**

**Oplog batch**

**sDedup Decoder**

**Reconstructed oplog entries**

**Oplog syncer**

**Source documents**

**Database**

**Replay**

**Primary Node**

**Secondary Node**
sDedup Encoding Steps

• Identify Similar Documents
• Select the Best Match
• Delta Compression
Identify Similar Documents

Rabin Chunking

Target Document

32 17 25 41 12

Consistent Sampling

Similarity Sketch

41 32

Feature Index Table

32 41

Candidate Documents

32 25 38 41 12

32 17 38 41 12

32 22 15

39 32

Similarity Score

1

Doc #1

2

Doc #2

2

Doc #3

Doc #1

Doc #2

Doc #3
Select the Best Match

<table>
<thead>
<tr>
<th>Rank</th>
<th>Candidates</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Doc #2</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>Doc #3</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>Doc #1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Initial Ranking**

<table>
<thead>
<tr>
<th>Rank</th>
<th>Candidates</th>
<th>Score</th>
<th>Cached?</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Doc #3</td>
<td>Yes</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Doc #1</td>
<td>Yes</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Doc #2</td>
<td>No</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

**Final Ranking**

*Is doc cached?*

*If yes, reward +2*
Evaluation

• MongoDB setup (v2.7)
  – 1 primary, 1 secondary node, 1 client
  – Node Config: 4 cores, 8GB RAM, 100GB HDD storage

• Datasets:
  – Wikipedia dump (20GB out of ~12TB)
  – Additional datasets evaluated in the paper
Compression

<table>
<thead>
<tr>
<th>Chunk Size</th>
<th>sDedup</th>
<th>trad-dedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>4KB</td>
<td>9.9</td>
<td>2.3</td>
</tr>
<tr>
<td>1KB</td>
<td>26.3</td>
<td>4.6</td>
</tr>
<tr>
<td>256B</td>
<td>38.4</td>
<td>9.1</td>
</tr>
<tr>
<td>64B</td>
<td>38.9</td>
<td>15.2</td>
</tr>
</tbody>
</table>

20GB sampled Wikipedia dataset
Memory

- Memory (MB): 34.1, 47.9, 57.3, 61.0, 80.2, 272.5, 780.5
- Chunk Size: 4KB, 1KB, 256B, 64B

Dedup methods:
- sDedup
- trad-dedup

20GB sampled Wikipedia dataset
Other Results (See Paper)

- Negligible client performance overhead
- Failure recovery is quick and easy
- Sharding does not hurt compression rate
- More datasets
  - Microsoft Exchange, Stack Exchange
Conclusion & Future Work

• **sDedup**: Similarity-based deduplication for replicated document databases.
  - Much greater data reduction than traditional dedup
  - Up to 38x compression ratio for Wikipedia
  - Resource-efficient design with negligible overhead

• **Future work**
  - More diverse datasets
  - Dedup for local database storage
  - Different similarity search schemes (e.g., super-fingerprints)
Backup Slides
Compression: StackExchange

Comparison of compression ratios for different chunk sizes:

- **sDedup**
- **trad-dedup**

**Chunk Size**:
- 4KB
- 1KB
- 256B
- 64B

**Compression Ratio**

1.0 1.0 1.2 1.0 1.3 1.1 1.8 1.2

10GB sampled StackExchange dataset
Memory: StackExchange

- **sDedup**
- **trad-dedup**

### 10GB sampled StackExchange dataset

<table>
<thead>
<tr>
<th>Chunk Size</th>
<th>Memory (MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4KB</td>
<td>83.9</td>
</tr>
<tr>
<td>1KB</td>
<td>115.4</td>
</tr>
<tr>
<td>256B</td>
<td>228.4</td>
</tr>
<tr>
<td>64B</td>
<td>414.3</td>
</tr>
<tr>
<td></td>
<td>3,082.5</td>
</tr>
</tbody>
</table>

- StackExchange dataset

---

**Note:** The diagram shows memory usage for different chunk sizes across two deduplication methods: sDedup and trad-dedup.
Throughput Overhead

![Bar graph showing insertion throughput (MB/s) for Wikipedia and Stack Exchange with and without sDedup.](image)

![Line graph showing run time (seconds) for Wikipedia and Stack Exchange with and without sDedup.](image)
Failure Recovery

20GB sampled Wikipedia dataset.
Dedup + Sharding

Compression Ratio

Number of Shards

20GB sampled Wikipedia dataset
Delta Compression

• Byte-level diff between source and target docs:
  – Based on the xDelta algorithm
  – Improved speed with minimal loss of compression

• Encoding:
  – Descriptors about duplicate/unique regions + unique bytes

• Decoding:
  – Use source doc + encoded output
  – Concatenate byte regions in order