A study of practical deduplication

Dutch T. Meyer
University of British Columbia
Microsoft Research Intern
William Bolosky
Microsoft Research
A study of practical deduplication

Dutch T. Meyer
University of British Columbia
Microsoft Research Intern

William Bolosky
Microsoft Research
Why study deduplication?

9ms per seek

$0.046 per GB
When do we exploit duplicates?

It Depends.

• How much can you get back from deduping?
• How does fragmenting files affect performance?
• How often will you access the data?
Outline

• Intro
• Methodology
• “There’s more here than dedup” teaser

(intermission)

• Deduplication Background
• Deplication Analysis
• Conclusion
Methodology

MD5(name)
Metadata
MD5(data)

MD5(name)
Metadata
MD5(data)

MD5(name)
Metadata
MD5(data)

MD5(name)
Metadata
MD5(data)

Once per week for 4 weeks.
~875 file systems
~40TB
~200M Files
There’s more here than dedup!

• We update and extend filesystem metadata findings from 2000 and 2004
• File system complexity is growing
• Read the paper to answer questions like:

Are my files bigger now than they used to be?
Teaser: Histogram of file size

File Size (bytes), power-of-two bins

- Since 1981!
There’s more here than dedup!

How fragmented are my files?
Teaser: Layout and Organization

• High linearity: only 4% of files fragmented in practice
  – Most windows machines defrag weekly
• One quarter of fragmented files have at least 170 fragments
Intermission

• Intro
• Methodology
• “There’s more here than dedup” teaser

(intermission)

• Deduplication Background
• Deplication Analysis
• Conclusion
Dedup Background

Whole file Deduplication

foo

01101010..... ....110010101

bar

01101010..... ....110010101
Dedup Background

Fixed Chunk Deduplication

foo

bar
Dedup Background

Rabin Fingerprinting

foo

bar

1 001000000.....

1 1 010100 

01101010.....

1101010010101010.....

110010101

=xxxxxx81

=xxxxxxx00

....110010101
The Deduplication Space

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Parameters</th>
<th>Cost</th>
<th>Deduplication effectiveness</th>
</tr>
</thead>
<tbody>
<tr>
<td>Whole-file</td>
<td></td>
<td>Low</td>
<td>Lowest</td>
</tr>
<tr>
<td>Fixed Chunk</td>
<td>Chunk Size</td>
<td>Seeks</td>
<td>Middle</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CPU</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Complexity</td>
<td></td>
</tr>
<tr>
<td>Rabin fingerprints</td>
<td>Average Chunk Size</td>
<td>Seeks More CPU</td>
<td>Highest</td>
</tr>
<tr>
<td></td>
<td></td>
<td>More Complexity</td>
<td></td>
</tr>
</tbody>
</table>
What is the relative deduplication rate of the algorithms?
Dedup by method and chunk size

The graph shows the percentage of space deduplicated across different chunk sizes for three methods: Whole File, Fixed-Chunk, and Rabin. As the chunk size decreases from 64K to 8K, the percentage of space deduplicated generally increases for all methods. The Rabin method consistently shows the highest percentage of space deduplicated across all chunk sizes, followed by Fixed-Chunk, and then Whole File.
What if I was doing full weekly backups?
Backup dedup over 4 weeks

- 8K rabin: 80%
- Whole File + Sparse: 70%
- Whole File: 65%
How does the number of filesystems influence deduplication?
Dedup by filesystem count

![Graph showing deduplication percentages for different domain sizes and deduplication methods.](image)
So what is filling up all this space?
Bytes by containing file size

Percentage of Total Bytes

Containing File Size (Bytes), Power-of-2 bins

- 2000
- 2004
- 2009
What types of files take up disk space?
Disk consumption by file type
Disk consumption by file type

- 2000:
  - dll: 60%
  - pdb: 20%
  - lib: 20%

- 2004:
  - dll: 60%
  - pdb: 20%
  - vhd: 20%

- 2009:
  - vhd: 60%
  - dll: 40%
Which of these types deduplicate well?
## Whole-file duplicates

<table>
<thead>
<tr>
<th>Extension</th>
<th>% of Duplicate Space</th>
<th>Mean File Size (bytes)</th>
<th>% of Total Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>dll</td>
<td>20%</td>
<td>521K</td>
<td>10%</td>
</tr>
<tr>
<td>lib</td>
<td>11%</td>
<td>1080K</td>
<td>7%</td>
</tr>
<tr>
<td>pdb</td>
<td>11%</td>
<td>2M</td>
<td>7%</td>
</tr>
<tr>
<td>&lt;none&gt;</td>
<td>7%</td>
<td>277K</td>
<td>13%</td>
</tr>
<tr>
<td>exe</td>
<td>6%</td>
<td>572K</td>
<td>4%</td>
</tr>
<tr>
<td>cab</td>
<td>4%</td>
<td>4M</td>
<td>2%</td>
</tr>
<tr>
<td>msp</td>
<td>3%</td>
<td>15M</td>
<td>2%</td>
</tr>
<tr>
<td>msi</td>
<td>3%</td>
<td>5M</td>
<td>1%</td>
</tr>
<tr>
<td>iso</td>
<td>2%</td>
<td>436M</td>
<td>2%</td>
</tr>
<tr>
<td>&lt;a guid&gt;</td>
<td>1%</td>
<td>604K</td>
<td>&lt;1%</td>
</tr>
</tbody>
</table>
What files make up the 20% difference between whole file dedup and sparse file, as compared to more aggressive deduplication?
Where does fine granularity help?

<table>
<thead>
<tr>
<th>Percentage of difference vs. whole file + sparse</th>
</tr>
</thead>
<tbody>
<tr>
<td>8K Fixed</td>
</tr>
<tr>
<td>vhd</td>
</tr>
<tr>
<td>lib</td>
</tr>
<tr>
<td>pdb</td>
</tr>
<tr>
<td>pdb</td>
</tr>
<tr>
<td>iso</td>
</tr>
<tr>
<td>dwl</td>
</tr>
<tr>
<td>avhd</td>
</tr>
<tr>
<td>ø</td>
</tr>
<tr>
<td>pst</td>
</tr>
<tr>
<td>wma</td>
</tr>
<tr>
<td>vhd</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>8K Rabin</th>
</tr>
</thead>
<tbody>
<tr>
<td>vhd</td>
</tr>
<tr>
<td>pdb</td>
</tr>
<tr>
<td>pdb</td>
</tr>
<tr>
<td>pdb</td>
</tr>
<tr>
<td>iso</td>
</tr>
<tr>
<td>dwl</td>
</tr>
<tr>
<td>avhd</td>
</tr>
<tr>
<td>ø</td>
</tr>
<tr>
<td>pst</td>
</tr>
<tr>
<td>wma</td>
</tr>
</tbody>
</table>

0% 10% 20% 30% 40% 50% 60% 70%
Last plea to read the whole paper

• ~4x more results in paper!
• Real world filesystem analysis is hard
  – Eight machines months in query processing
  – Requires careful simplifying assumptions
  – Requires heavy optimization
Conclusion

• The benefit of fine grained dedup is < 20%
  – Potentially just a fraction of that.
• Fragmentation is a manageable problem
• Read the paper for more metadata results

We’re releasing this dataset