Evaluation of Efficient Archival Storage Techniques

Lawrence You

University of California, Santa Cruz
Jack Baskin School of Engineering
1156 High Street, Santa Cruz, California 95064
Tel: +1-831-459-4458
you@cs.ucsc.edu

NASA/IEEE MSST 2004
12th NASA Goddard/21st IEEE Conference on
Mass Storage Systems & Technologies
The Inn and Conference Center
University of Maryland University College
Adelphi MD USA
April 13-16, 2004

Motivation

- An increasing need to store immutable data
- Disk-based archival storage satisfies some needs, bandwidth and latency, but not cost
- Two different strategies are used today to eliminate or reduce redundancy that exists across files
  - **chunking**: sub-file content-addressable storage (LBFS, Avamar, Venti, HP Labs ElephantStore)
  - **resemblance + delta**: detect similar files, store delta compressed files (UCSC Deep Store, DERD)
- Two strategies improve storage efficiency when similar files are stored

What is the most space-efficient way to store immutable data?

Investigation for the UCSC **Deep Store** Project

Evaluation Metric

Storage efficiency (%) =

compressed size ÷ original size

Account for all data, not just incremental

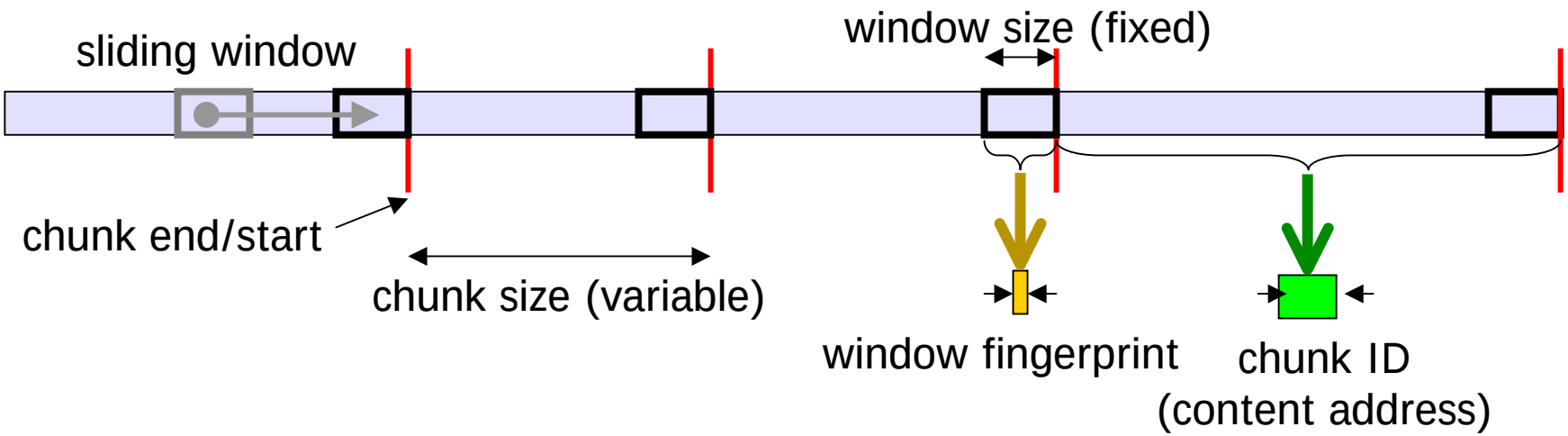
Account for overhead

Inter-file vs. intra-file redundancy:

- We aim to eliminate redundancy across files, or “inter-file compression”
- We compare against a baseline: **gzip** (**zlib**), or “intra-file compression”

Chunking

1. Divide files into content-addressable chunks



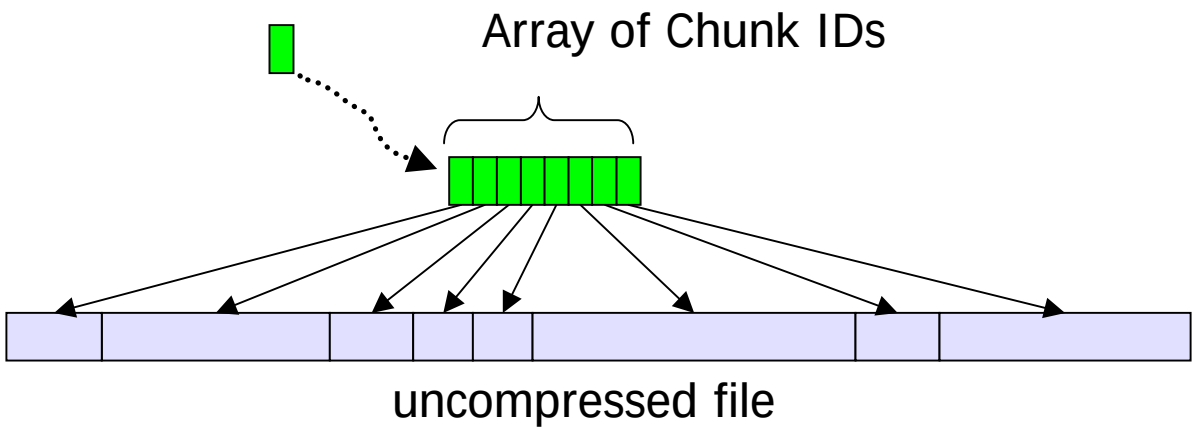
Divide files into variable-sized chunks described by boundaries within the data

A chunk division (red “breakpoint”) is created when the fingerprint for the sliding window meets a criteria

Muthitacharoen, Chen, and Mazières. **A low-bandwidth network file system**. (LBFS) SOSP ’01

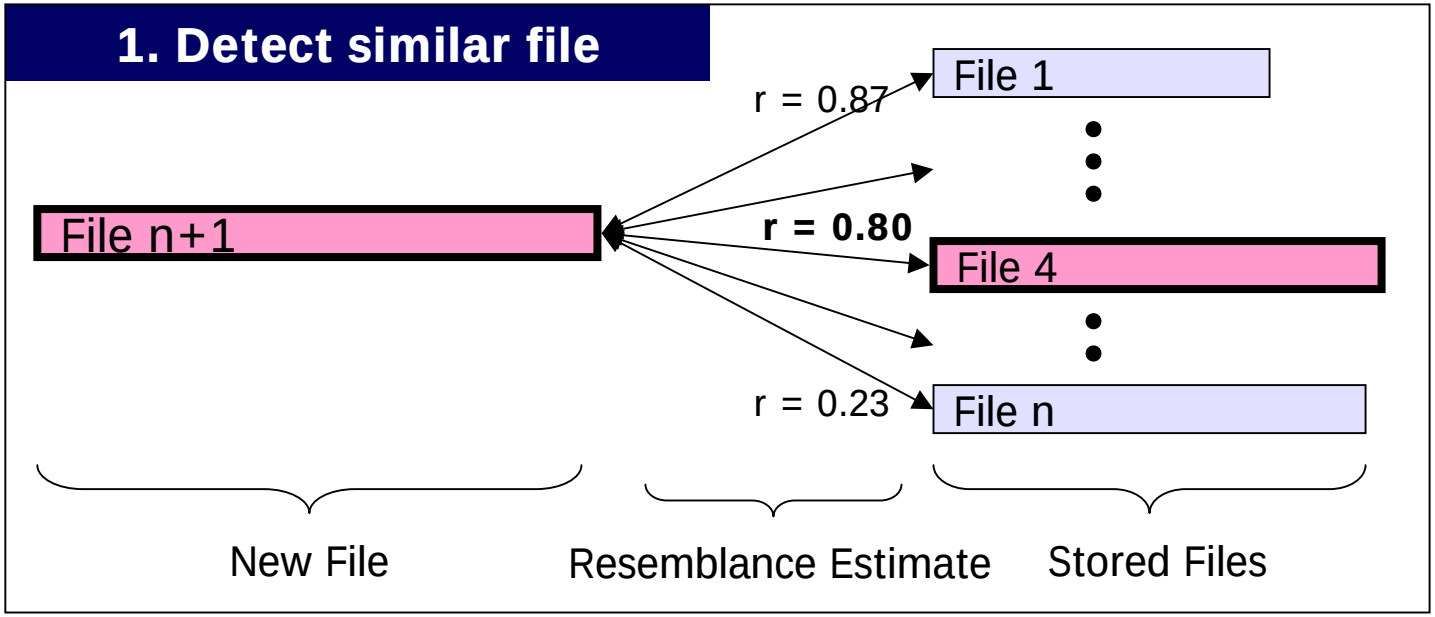
Addressing and Storing Chunks

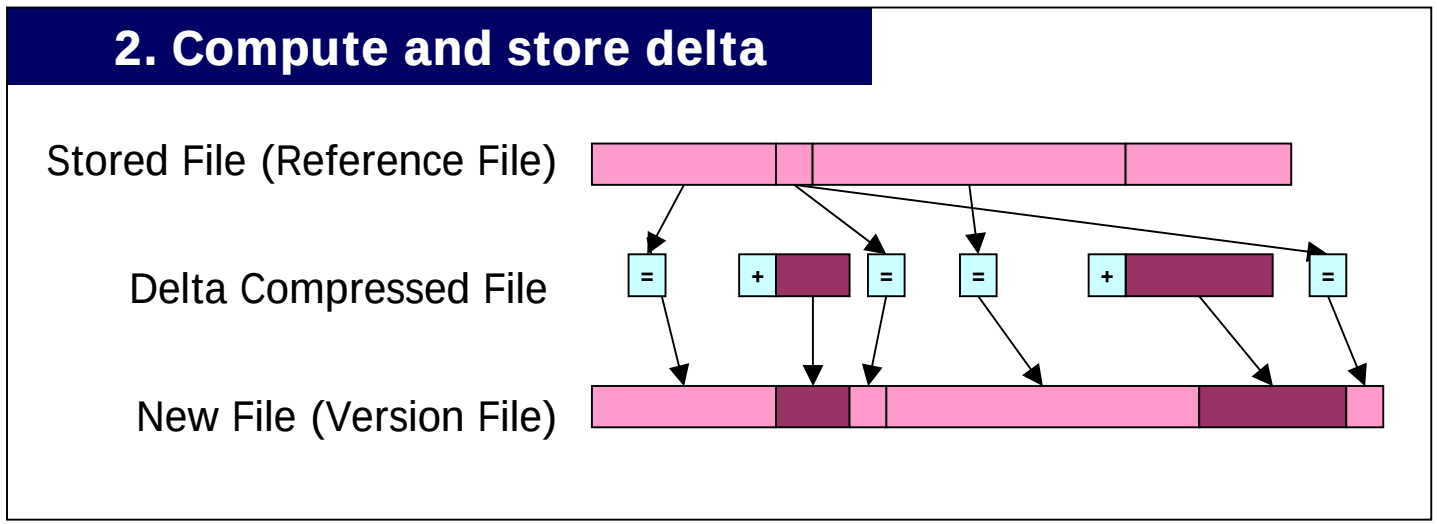
2. Store chunk data



- Store:
  - per chunk: **chunk ID, chunk data**
  - per file: **list of chunk IDs**

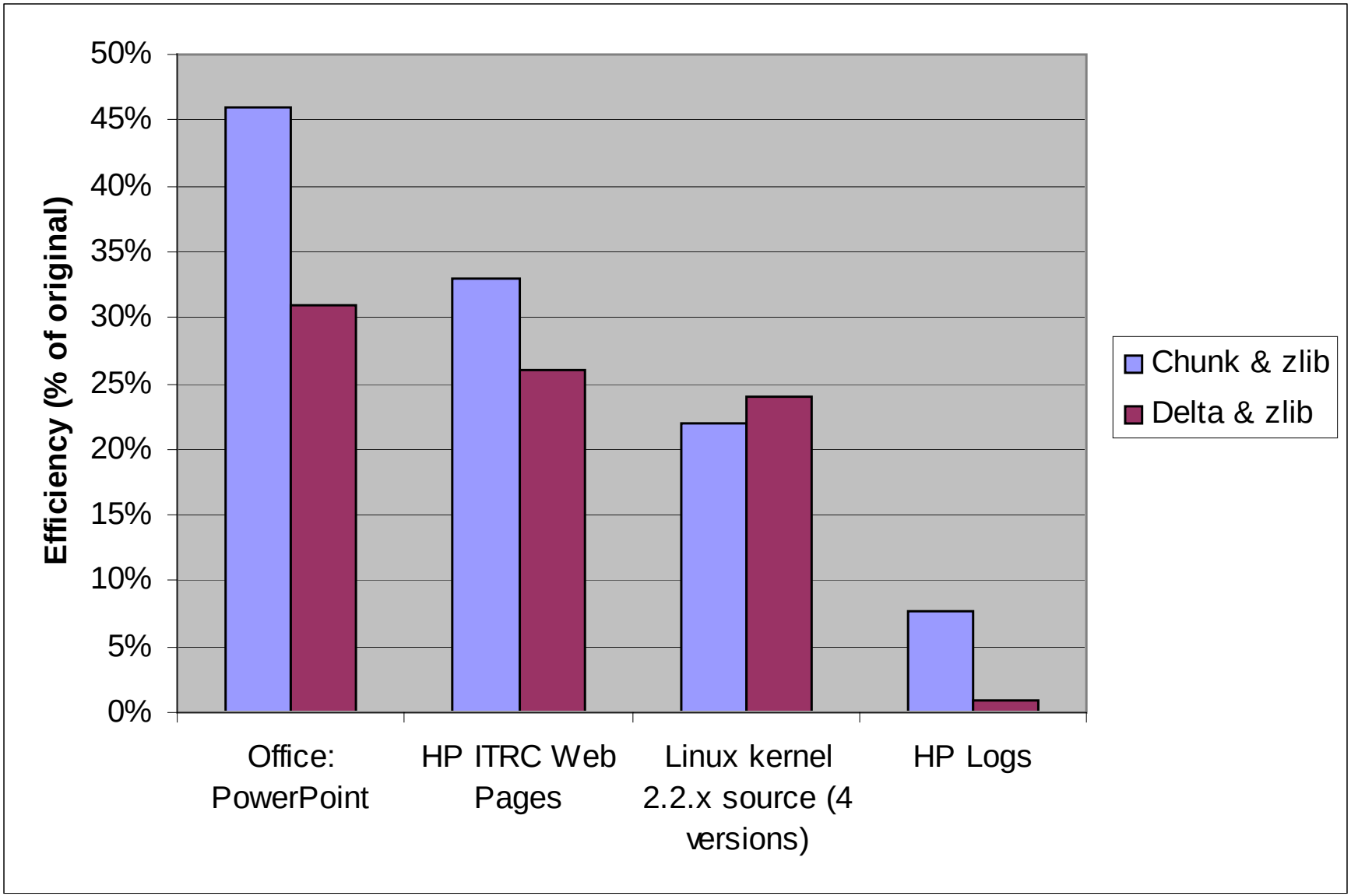
Detect Similar Files/Delta Compression





Douglis and Iyengar, **Application-specific Delta-encoding via Resemblance Detection**, USENIX 2003 Annual Technical Conference.

Chunking vs. Delta



Storage Efficiency Results

- **Chunking** and **zlib** is best for highly similar (versioned) data: some types of versioned data
  - 22% of original size for four versions of Linux source
- **Delta** and **zlib** is best overall for collections of similar binary data and machine-generated text data
  - Under 1% of the original size (**100:1** compression) for computer-generated log data

Conclusions

- Experimental data compares the two methods using the same data
- Chunking offers direct access to chunk storage, but must contend with overhead
- Delta compression between similar files provides improved storage efficiency in most data sets, and very high compression with highly similar data sets
- Opportunities for hybrid techniques using both sub-file CAS and delta compression

Acknowledgements

Christos Karamanolis, co-author

Darrell Long, advisor

Project funding provided by HP Labs, Microsoft Research and the NSF