



Fingerdiff: Improved Duplicate Elimination for Storage Systems

Deepak Bobbarjung
Cezary Dubnicki
Suresh Jagannathan



Problem statement

- Improved duplicate elimination can reduce storage and bandwidth utilization in content-addressable storage networks.
- Can we improve the duplicate elimination that can be obtained in storage systems?.



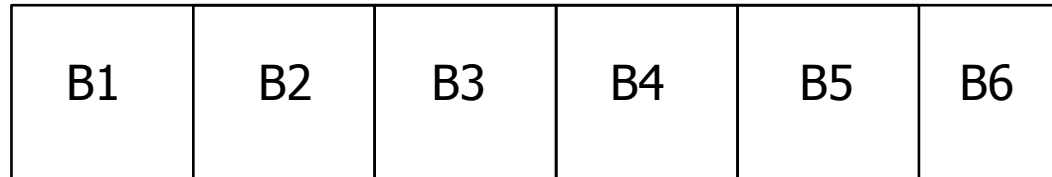
Fingerdiff

- A data partitioning technique that provides better duplicate elimination in storage system while introducing fewer overheads.

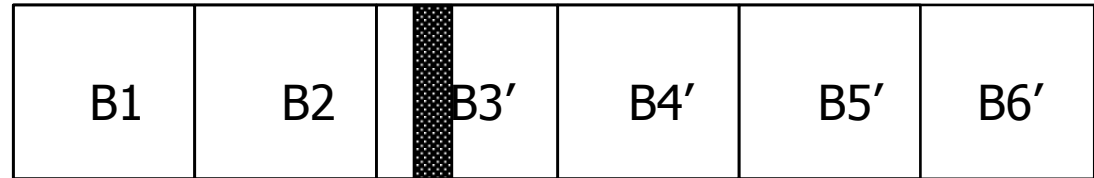


Fixed size chunking (*FSC*)

File F Version 1

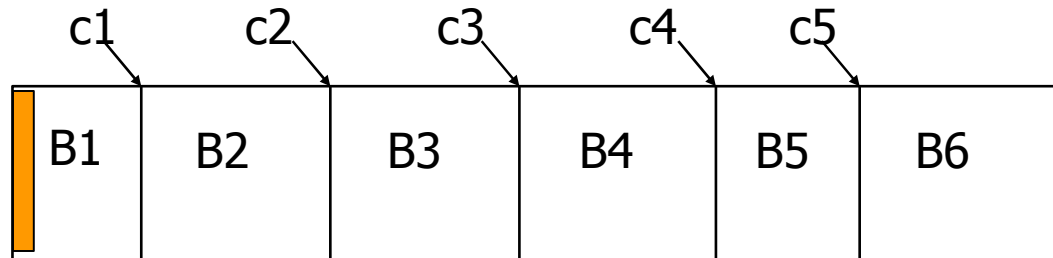


File F Version 2

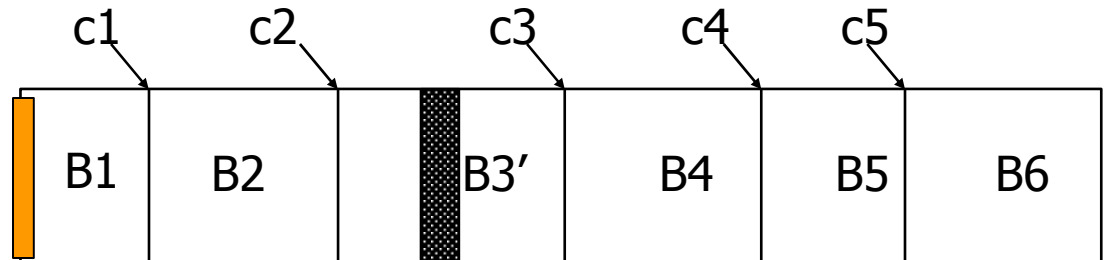


Content determined chunking (*CDC*)

File f version 1



File f version 2





Avenues for improvement of *CDC*

- Variable sized chunks but with a small margin of error.
- Size of any new chunk is the expected chunk size with a small margin of error.
- Large expected chunk size implies large sizes for all new chunks
- Small expected chunk size implies many small sized chunks resulting in large metadata overheads.

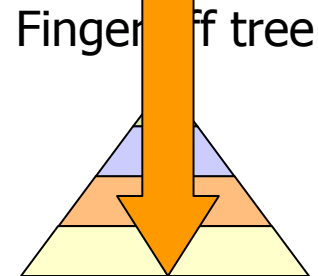
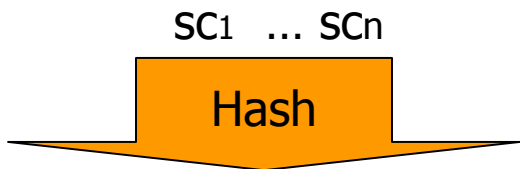
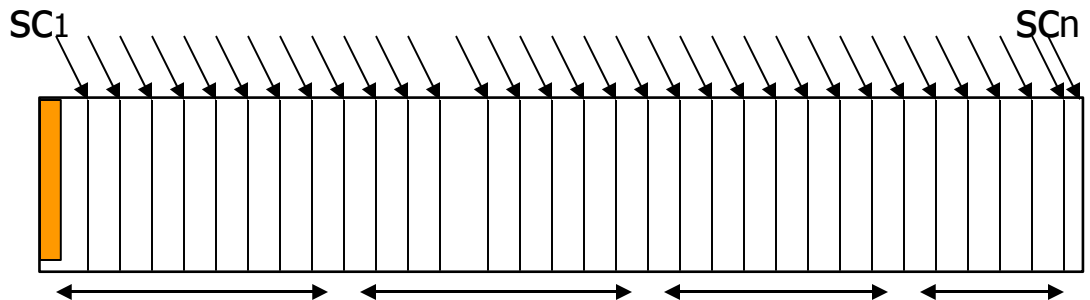


fingerdiff

- Goal:
 - Be able to partition objects into very small chunks without imposing overheads of too many chunks.
- Solution:
 - Partition into small chunks only when necessary. (i.e. in regions of change).
 - Coalesce small chunks into larger chunks wherever possible. (i.e. in regions of no change).

fingerdiff

File f version 1



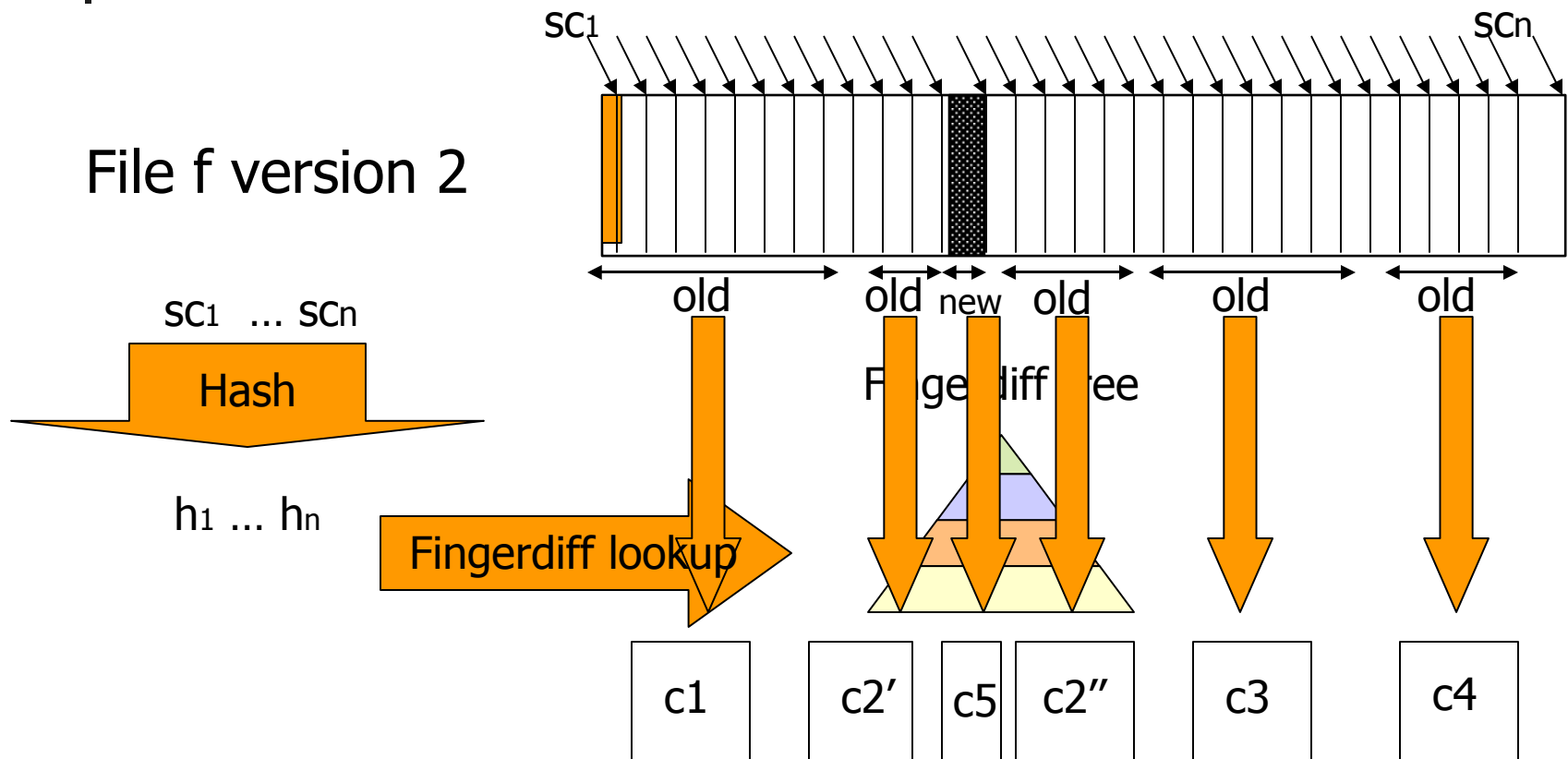
c1

c2

c3

c4

fingerdiff



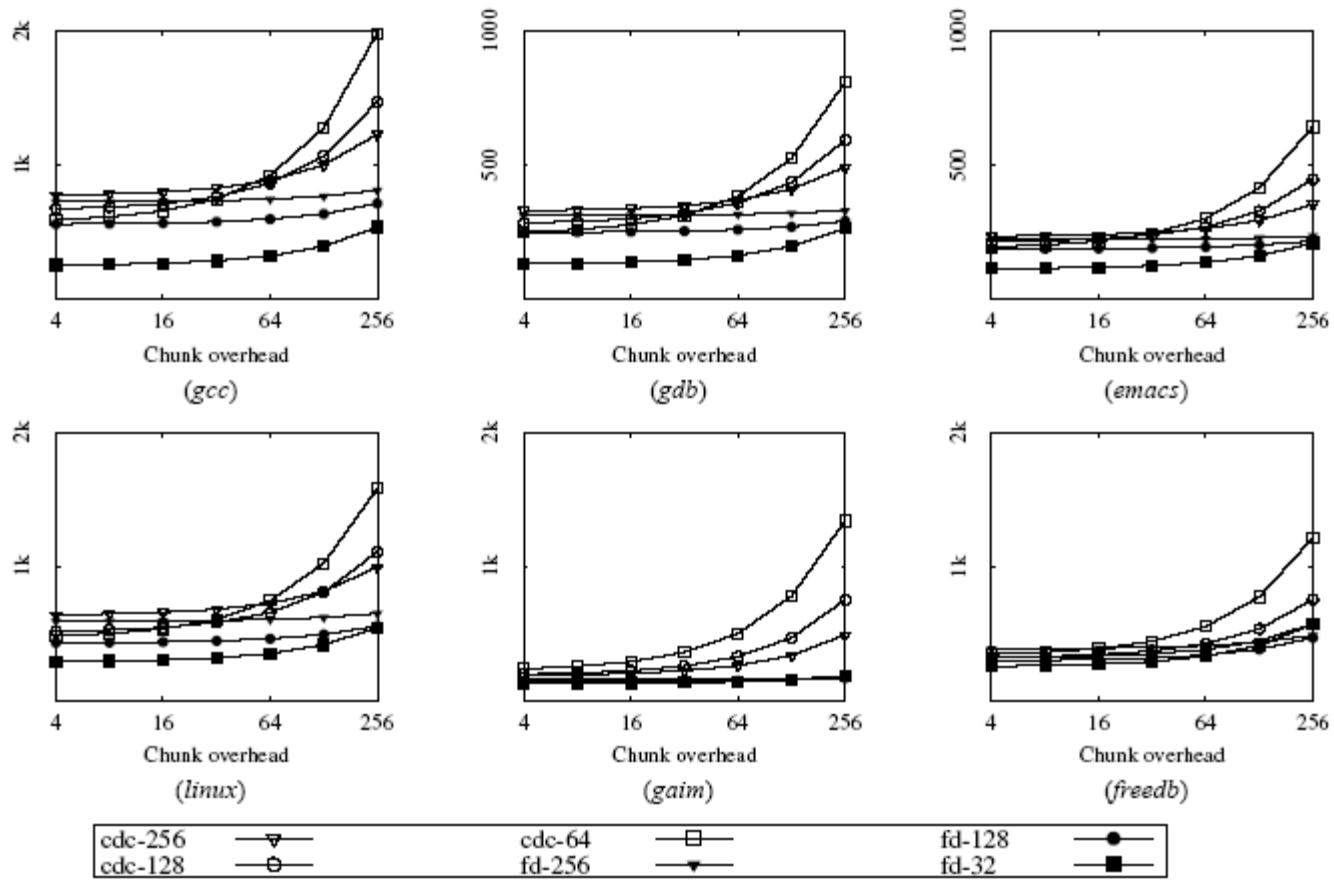
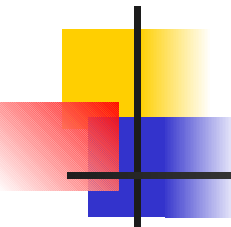


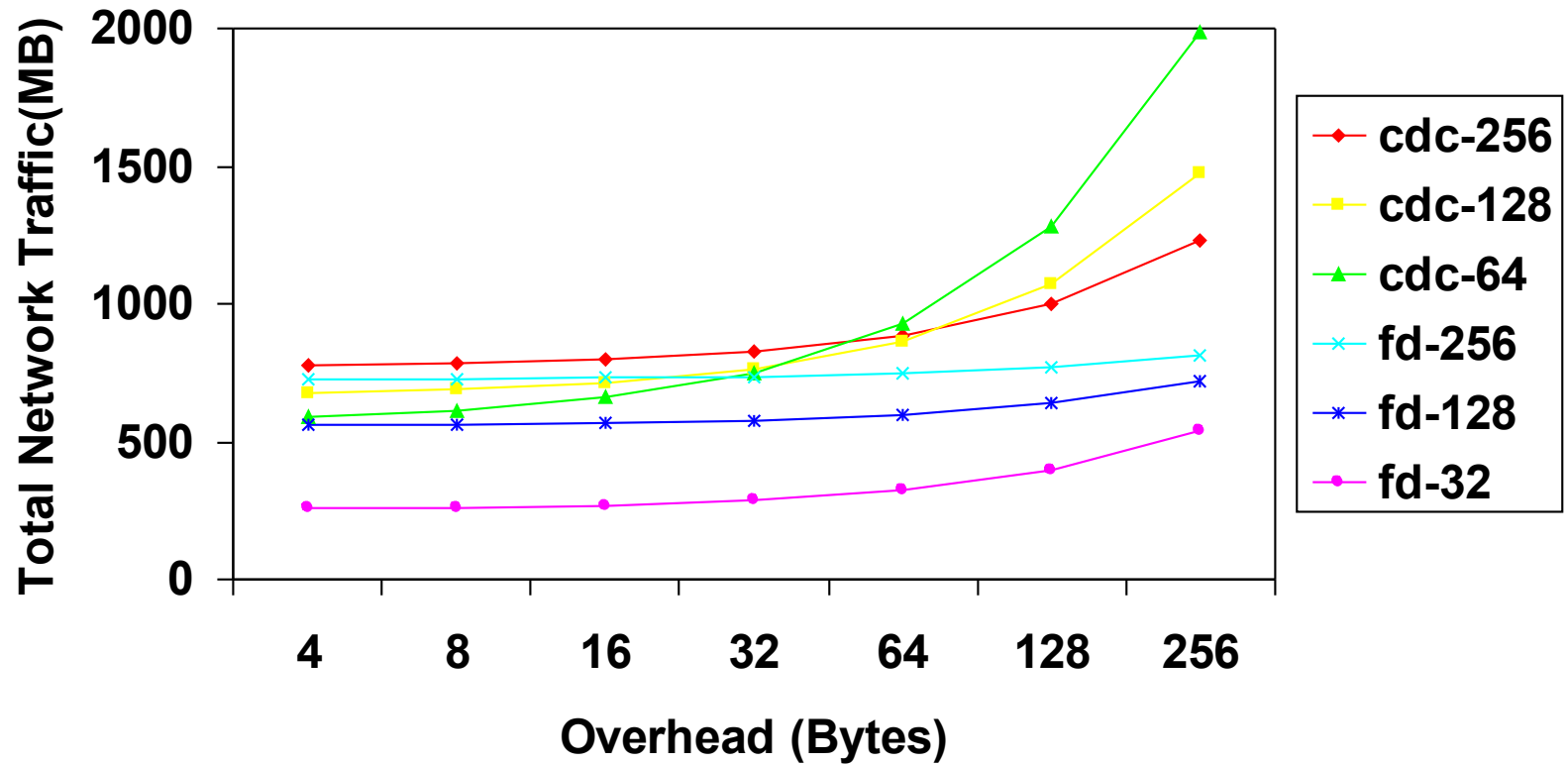
Figure 1. Comparison of the total network traffic (in MB) consumed by six of the ten chunking technique instantiations after writing each benchmark on a content addressable chunk store. The X-axis of each graph is a log plot which gives the chunk overhead; i.e the overhead in bytes associated with transferring one chunk of data from the driver to the chunk store. The network traffic measured is between the object server and the chunk store. The Y-axis gives the total network traffic generated in MB after writing each benchmark to the chunk store.



Storage space consumption

benchmark	<i>cde-2k</i>	<i>cde-256</i>	<i>cde-128</i>	<i>cde-64</i>	<i>cde-32</i>	<i>fd-2k</i>	<i>fd-256</i>	<i>fd-128</i>	<i>fd-64</i>	<i>fd-32</i>	% saving
Sources											
<i>gcc</i>	1414	866	828	859	979	1400	799	680	579	498	40
<i>gdb</i>	501	363	344	358	500	498	336	293	255	255	26
<i>emacs</i>	327	258	259	281	457	324	239	220	199	221	25
<i>linux</i>	1204	708	629	692	985	1195	644	520	469	543	23
Databases											
<i>freedb</i>	396	348	369	442	644	370	396	317	291	290	17
Binaries											
<i>gaim</i>	225	245	301	447	527	213	196	208	244	246	13
Total	4067	2788	2731	3079	4090	3999	2611	2238	2038	2052	25

Total Network Traffic Consumption for *gcc*





Conclusions

- Existing object partitioning techniques cannot improve the duplicate elimination without increasing the metadata management overheads imposed on the system.
- *Fingerdiff* enabled duplicate elimination provides better storage and network utilization over existing techniques.