BUILDING A BETTER CONNETED WORLD

Leap-based Content Defined Chunking --- Theory and Implementation

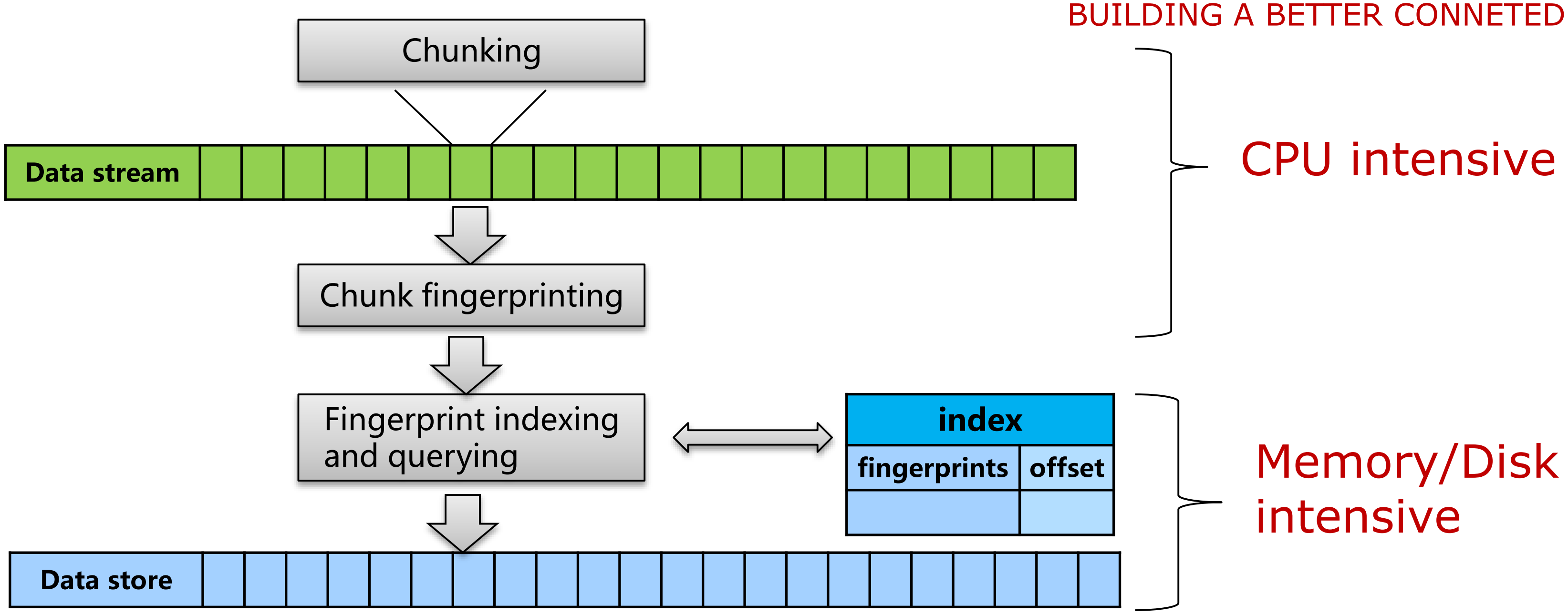
Chuanshuai Yu, Chengwei Zhang, Yiping Mao, Fulu Li

Huawei Technologies

Agenda

- Introduction & motivation
- Proposed algorithm
- Algorithm validation
- Conclusion

Issue and Motivation



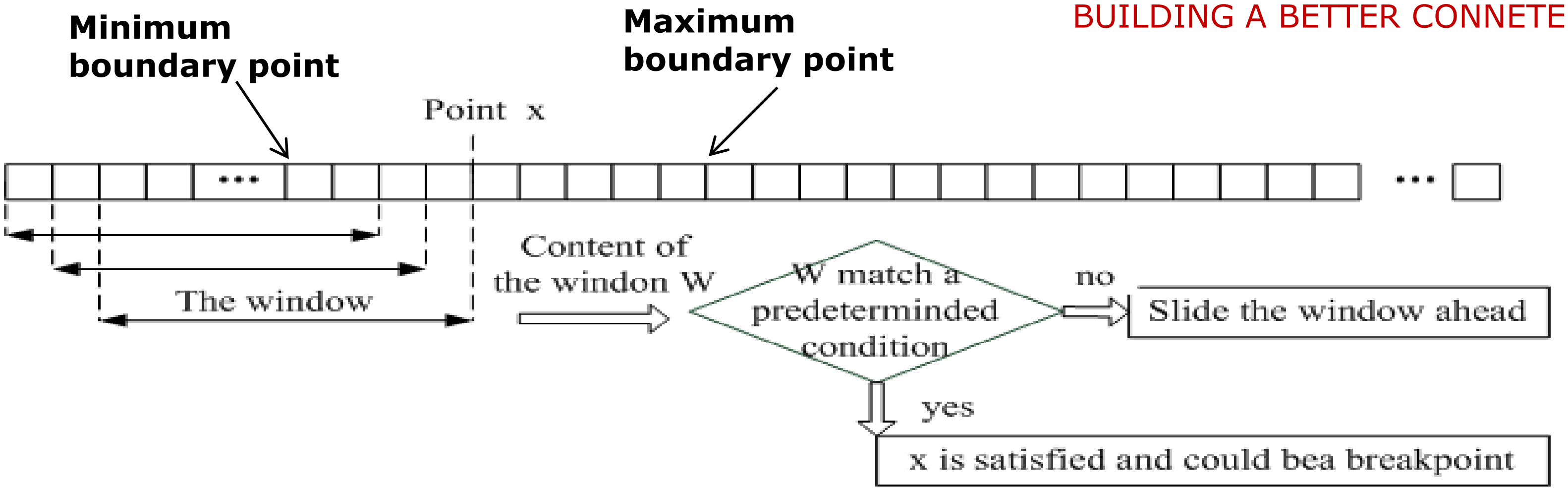
A "good" chunking algorithm should satisfy the following criteria:

1. outputting right data chunks so that the dedup can approach to the best inherent de-duplication ratio.
2. fast chunking speed with a low CPU overhead.

We will focus on both (1) and (2) in this work

Sliding-window-based Content Defined Chunking

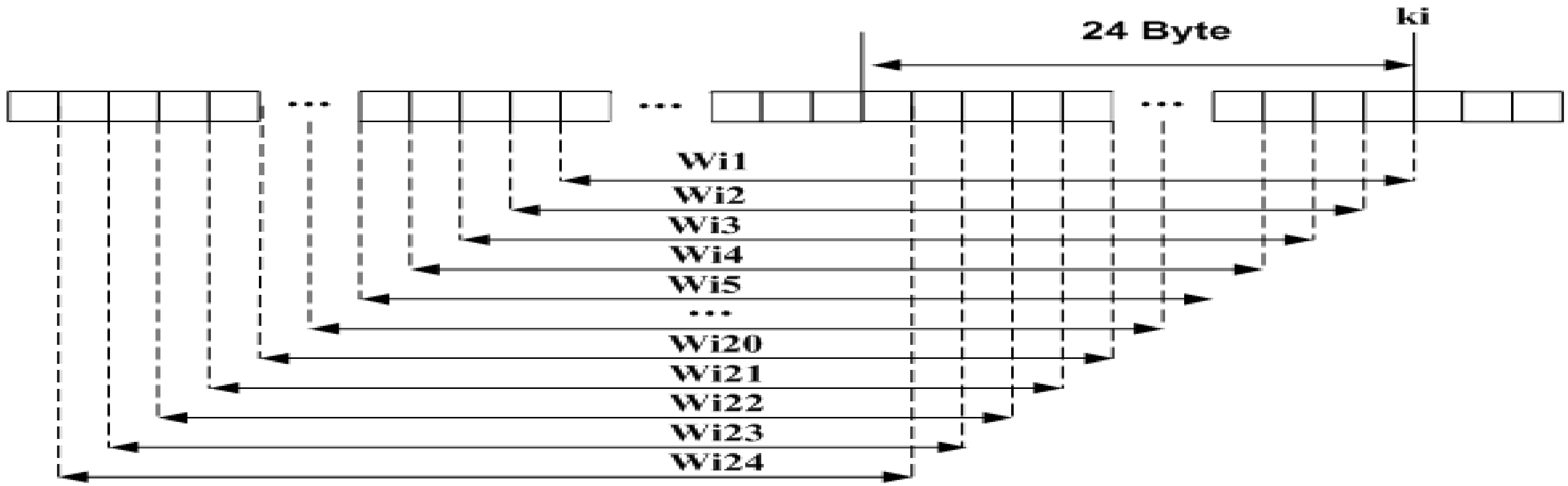
BUILDING A BETTER CONNETED WORLD



1. Calculate the fingerprint of each window of pre-defined size and with the end-point lying between the minimum and the maximum boundary points.
2. If the fingerprint satisfies a given condition, then the chunk end-point is set at the end-point of the window.
3. If not, then the window is slid forward by one byte and repeat the step 1 and 2, until reach to the maximum boundary point and in this case, set the chunk end-point at the maximum boundary point.

Issue: Heavy computation due to byte sliding mechanism.

Proposed Leap-based Content Defined Chunking

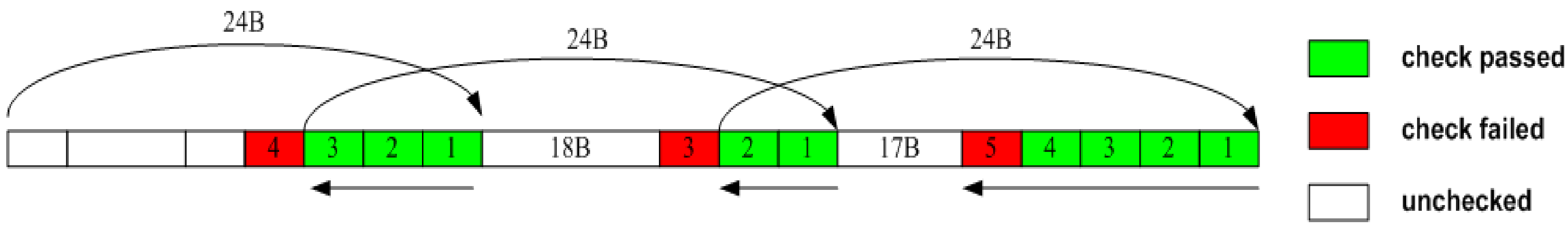


Similar setting as in sliding-window-based CDC, except that:

1. Instead of checking the fingerprint of one window, the proposed algorithm check k windows (say, k=24) to set chunk end-point.
2. Only when all of k windows' fingerprints meet the given conditions the chunk is set.
3. It will leap forward k bytes as soon as one window fails the check condition (The detailed leap procedure is addressed in next slide).

Leap Procedure

BUILDING A BETTER CONNETED WORLD

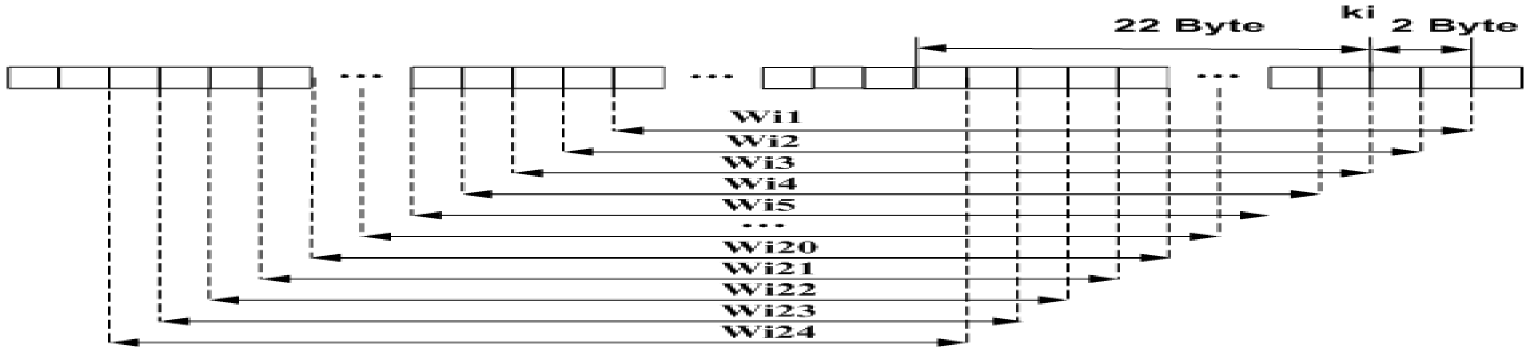


1. Starting from the right-most one among k windows, backward check the given condition for each of k windows.
2. If all k windows satisfy given condition, then a chunk end-point is set at the most-right window's end-point.
3. Once encounter the first window that fails the check condition, leap 24 bytes forward from the failed point.
4. Repeat step 1 – step 3 until reach to or jump over the maximum boundary point and in this case, set the chunk end-point at the maximum boundary point.

Since the probability of the failed condition for each window is high(= ¼ in our typical design), leap forwarding will take place before check out all k windows in most time (averagely check 3 windows) and hence speed up the chunking.

The Secondary Conditions

- To reduce the probability of forcing set of a chunk, the well-known TTTD algorithm introduces the secondary conditions.



- Similarly, we can also introduce the secondary conditions in proposed leap-based chunking algorithm.

Theoretical Analysis

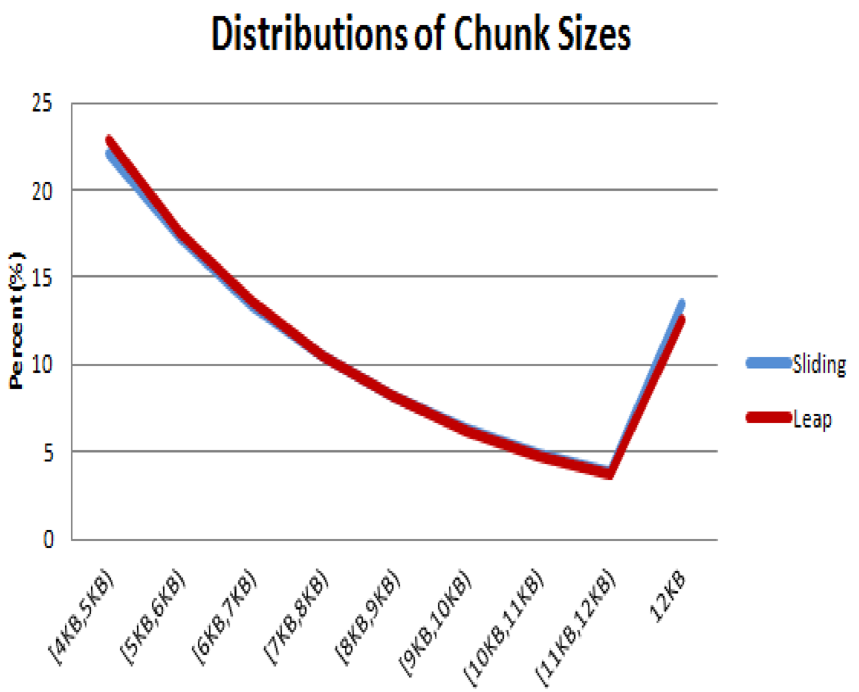
BUILDING A BETTER CONNETED WORLD

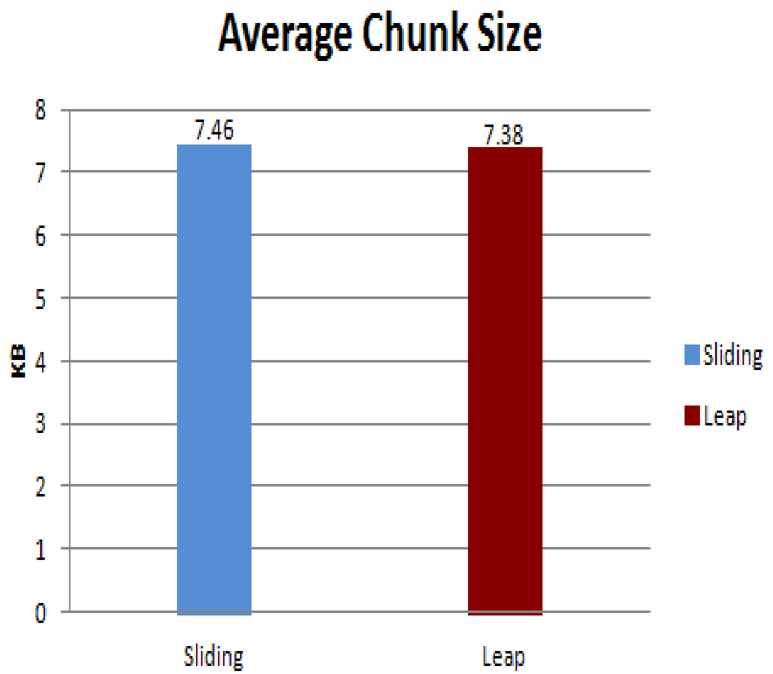
	Probability of one failed window check	Action after failed check	Average amount checked	Average cost of each check
Sliding window based	1- 1/4096	Sliding 1 byte	1X	1X
Leap based	1/4	Leap 24 bytes	1/5 X	2.5X

Theoretically, the computation cost of one chunking by the leap-based algorithm is about half of the one by the sliding-window-based algorithm .

Theoretical Analysis on Chunk Sizes

BUILDING A BETTER CONNETED WORLD

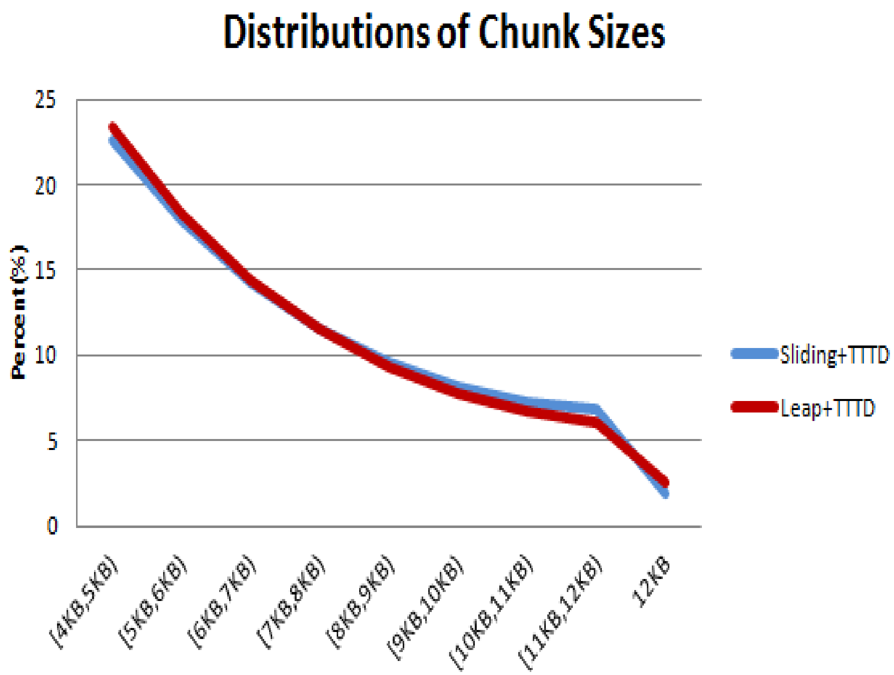


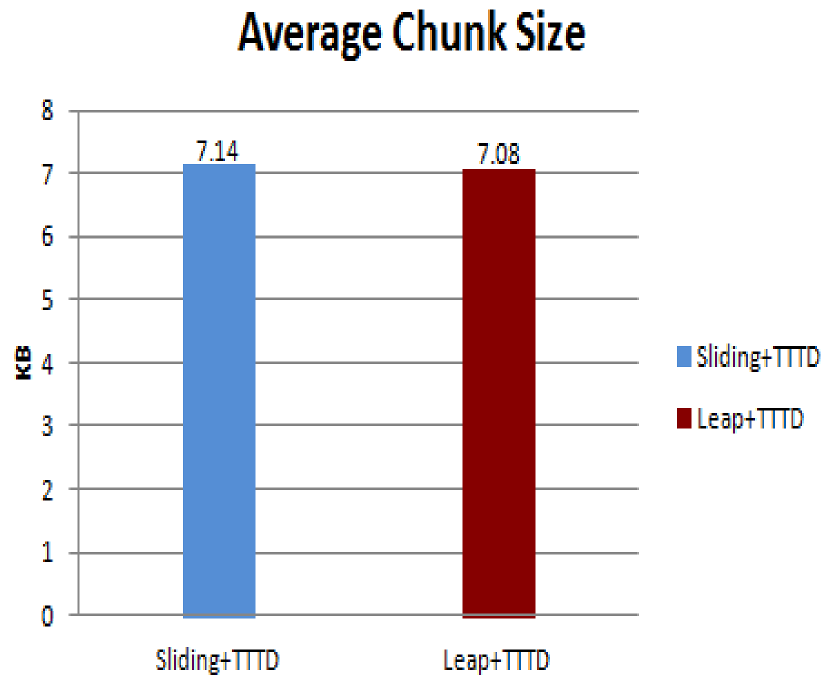


The distributions of chunk sizes outputted from two algorithms are similar

Theoretical Analysis on Chunk Sizes – with TTTD

BUILDING A BETTER CONNETED WORLD





The distributions of chunk sizes outputted from two algorithms with TTTD conditions are similar

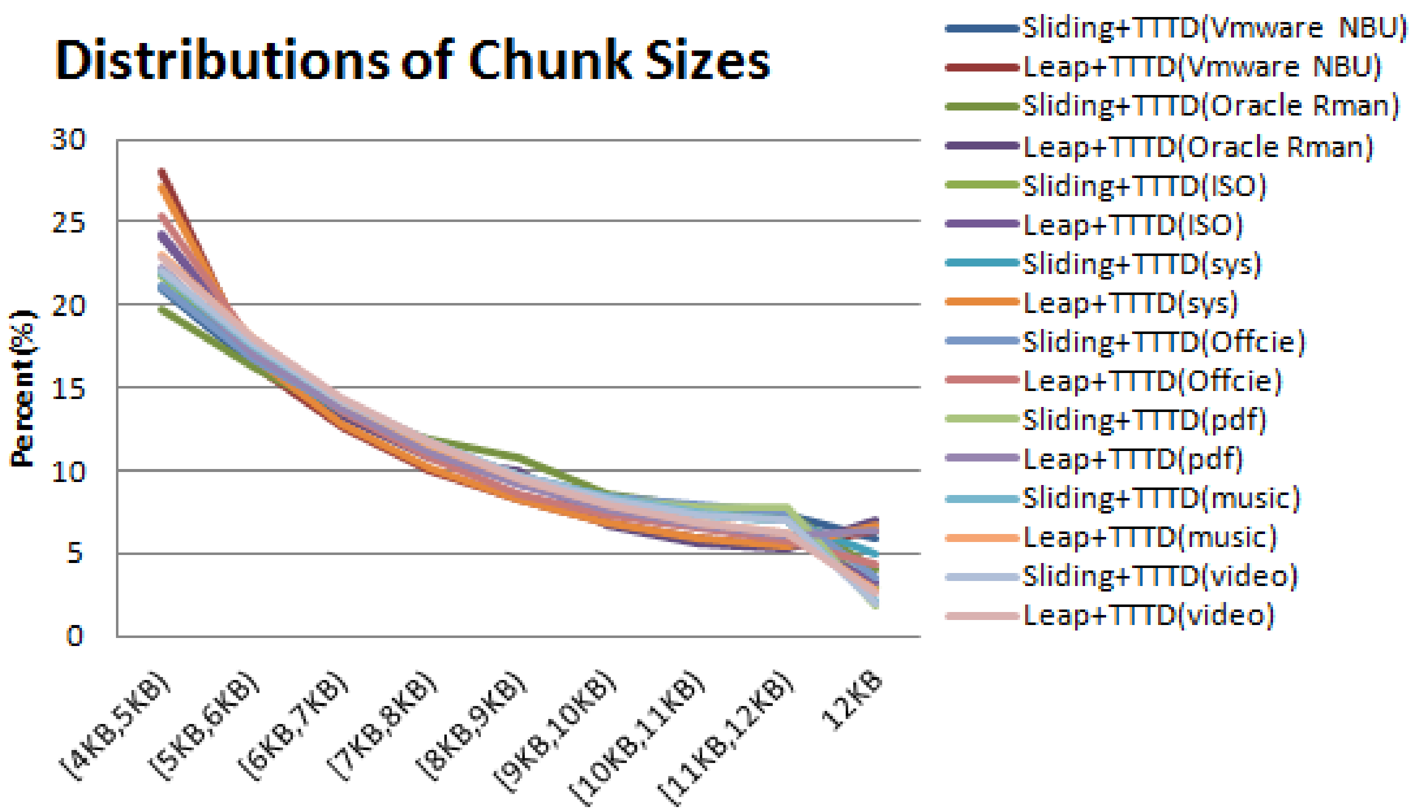
Datasets Used in Experiments

Type	Size(KB)	the way we generated them
vmware	81300750	This dataset is gotten by backuping 10 VMvare files of Windows7 system by NetBackup software.
oracle_tbs_rman	14427720	This dataset is gotten by backuping a real database by RMAN interface.
oracle_tbs_dmp	10602880	This is the dmp file of a real database.
oracle_tbs_dbf	15990792	This is the dbf file of a real database.
sys	153871100	This dataset collects data of 20 C disks. The data is packed together without compression.
ISO	45486080	This dataset collects 20 ISO install files different versions of Windows operating system.
office	18114600	This dataset collects all kinds of office files, including doc, xls, ppt and so on. These files are packed together without compression.
music	4556260	This dataset collects all kinds of music files. These files are packed together without compression.
video	11327510	This dataset collects all kinds of video files. These files are packed together without compression.
pdf	4714870	This dataset collects all kinds of pdf files. These files are packed together without compression.

All datasets were collected from real production environments

Distributions of Chunk Sizes from Experiments.

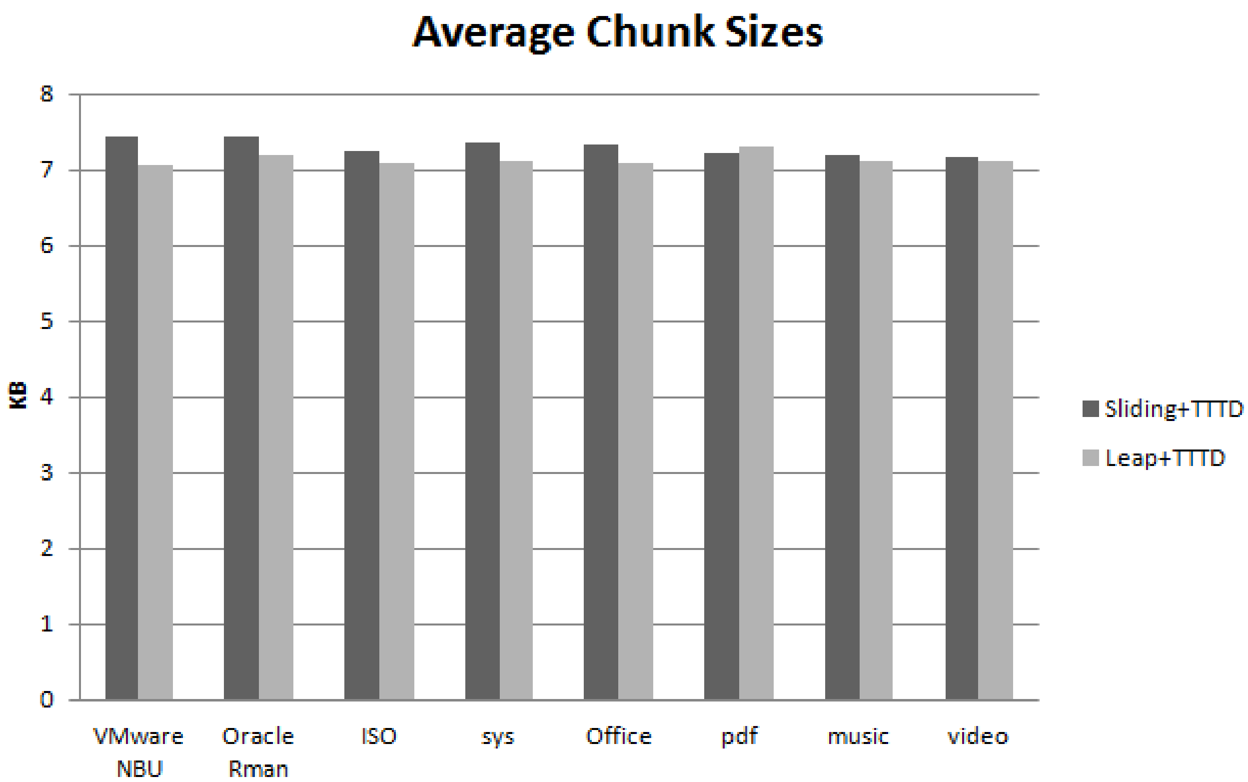
BUILDING A BETTER CONNETED WORLD



The distributions of chunk sizes outputted from two algorithms with a secondary condition agree with the theoretical analysis.

Average Chunk Size from Experiments

BUILDING A BETTER CONNETED WORLD



The average chunk sizes outputted from the two algorithms with a secondary condition agree with the theoretical analysis.

De-duplication Ratio from Experiments

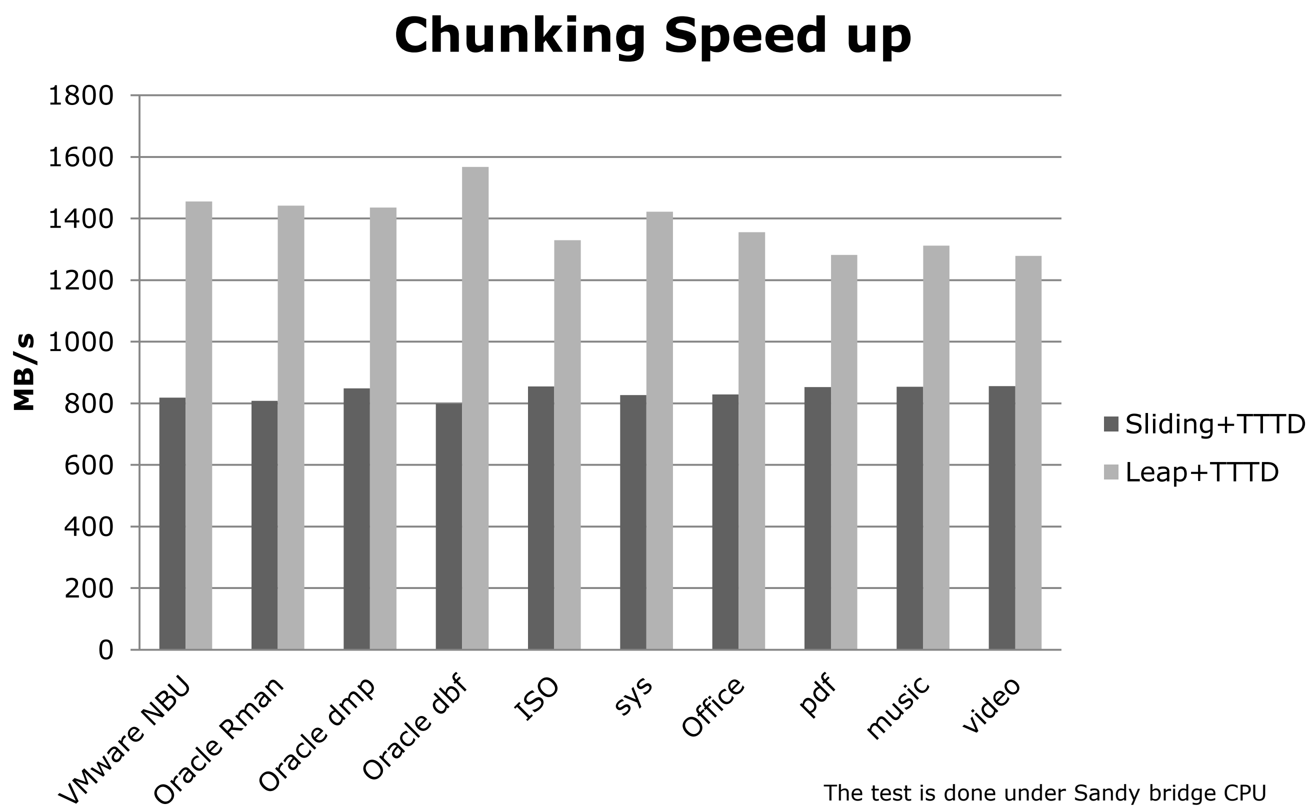
BUILDING A BETTER CONNETED WORLD

	VMware NBU	Oracle Rman	ISO	sys	Office	pdf	music	video
Sliding+ TTTD	7.84640	1.00025	2.01905	3.56711	1.28722	1.05202	1.10806	1.00009
Leap+ TTTD	7.81146	1.00058	2.01983	3.55052	1.28828	1.05269	1.10852	1.00001

The deduplication ratio delivered by two algorithms are almost the same.

Chunking Speed Comparison

BUILDING A BETTER CONNETED WORLD



The test is done under Sandy bridge CPU

Leap-based CDC algorithm with the secondary condition speeds up the chunking by 50%~100%.

Conclusion

- The leap-based CDC algorithm can speed up chunking in 50% ~ 100% range.

- The leap-based algorithms outputs the similar distribution of chunk sizes with sliding-window-based algorithm and hence delivers the similar de-duplication ratio.

Thank You