# CSCI 5980
# Content Defined Chunking in Data Deduplication

Professor: David HC Du
Wei-Yu Chen    Chai-Wen Hsieh

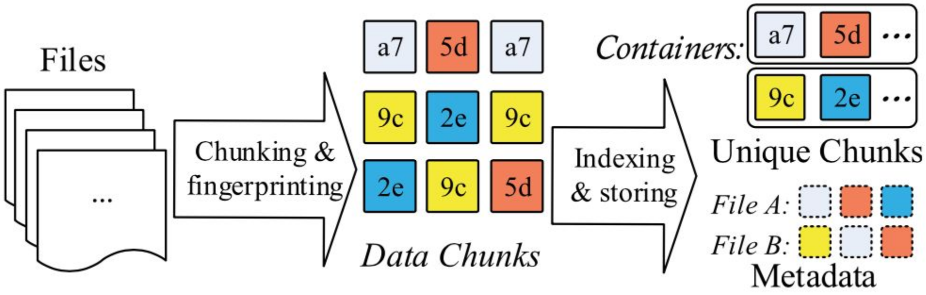
Apr 27 2020

# Abstract

- Deduplication Process
- Fixed-size (FSC) & content defined chunking (CDC)
- Three Rolling Hash Algorithm
- Rabin-based CDC
- Problems of Rabin-based CDC
  - BSW
  - TTTD
  - Subchunk
- FastCDC
- Can We Do Better?

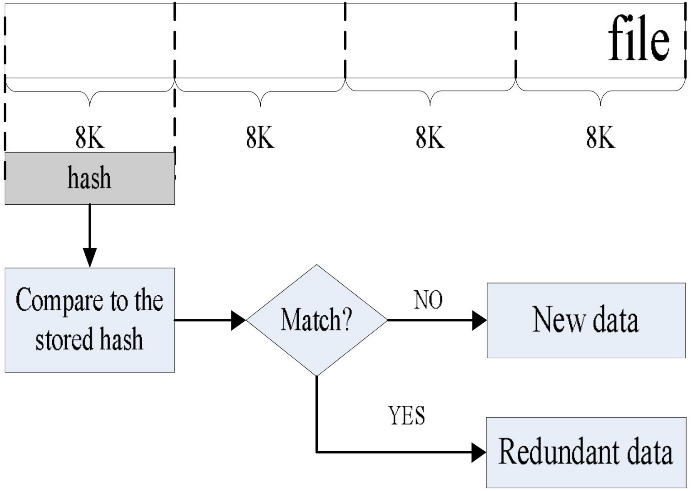
# Deduplication Process

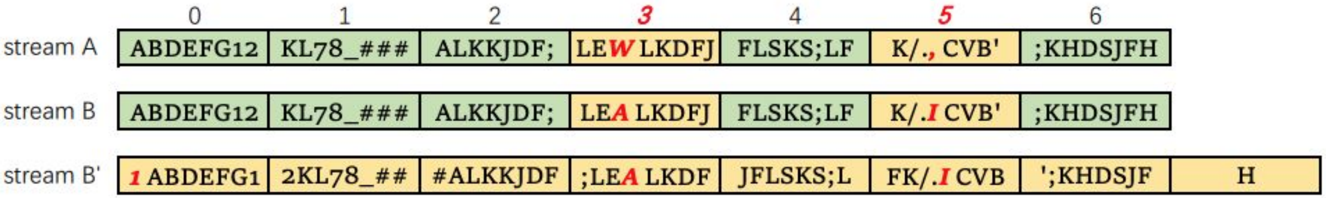
- Chunking
  - FSC and CDC
- Fingerprinting
  - SHA-1, SHA-256
- Indexing
  - Deduplicate identical chunks
- Storing



# Fixed-Size Chunking (FSC)

- Breakpoint
  - Fixed-Size
- Simple and Fast
- Low deduplication ratio
  - Boundary-shift problem

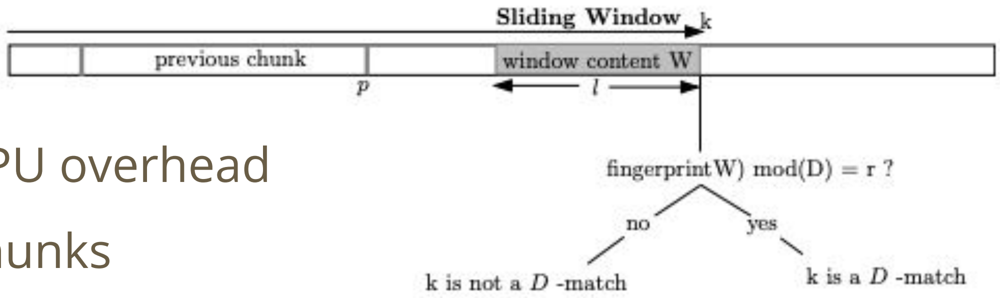


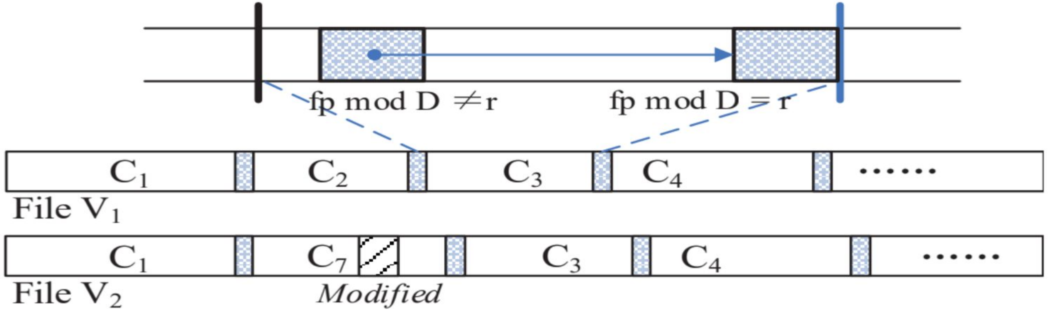


| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | |
|---|---|---|---|---|---|---|---|---|
| stream A | ABDEFG12 | KL78_### | ALKKJDF; | LEW LKDFJ | FLSKS;LF | K/., CVB' | ;KHDSJFH | |
| stream B | ABDEFG12 | KL78_### | ALKKJDF; | LEA LKDFJ | FLSKS;LF | K/.I CVB' | ;KHDSJFH | |
| stream B' | 1 ABDEFG1 | 2KL78_## | #ALKKJDF | ;LEA LKDF | JFLSKS;L | FK/.I CVB | ';KHDSJF | H |

# Content-Defined Chunking (CDC)

- Breakpoint
  - Content-Defined
- Time consuming and heavy CPU overhead
- Updating only the modified chunks
  - Boundary-shift problem solved



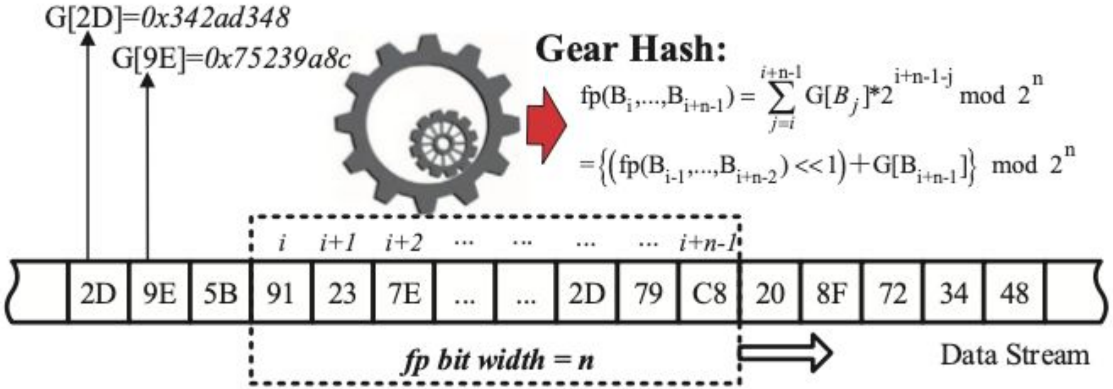


# Three Rolling Hash Algorithm

- Rabin

$$Rabin(B_1, B_2, ..., B_\alpha) = A(p) = \{\sum_{x=1}^{\alpha} B_x p^{\alpha - x}\} mod\ D$$

- Adler
- Gear



Gear Hash:

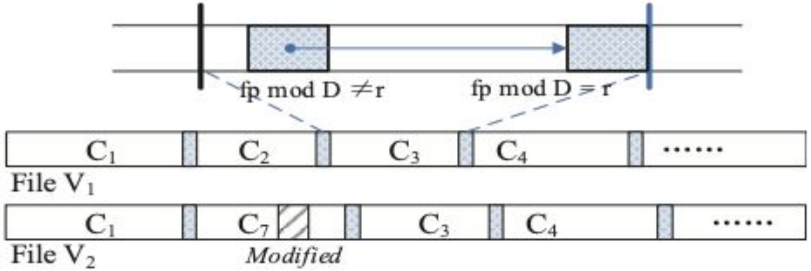
$$fp(B_i,...,B_{i+n-1}) = \sum_{j=i}^{i+n-1} G[B_j]*2^{i+n-1-j} \text{ mod } 2^n$$

$$= \left\{ \left( fp(B_{i-1},...,B_{i+n-2}) << 1 \right) + G[B_{i+n-1}] \right\} \text{ mod } 2^n$$

# Rabin-based CDC

- Rolling hash algorithm
  - Random polynomial
  - Compute Incrementally
- Basic Sliding Window (BSW) algorithm
  - Rabin-based CDC
  - Byte-by-byte
  - D: sliding window size
  - Usually D and r are 0x02000 and 0x78
  - Chunks are 8KB

$$
\begin{aligned}
&\mathrm{Rabin}(B_{i+1}, B_{i+2}, \ldots, B_{i+\alpha}) \\
&\quad = \left\{ \sum_{x=i+1}^{i+\alpha} B_x p^{\alpha-x+i} \right\} \bmod D \\
&\quad = \left\{ \left[ \sum_{x=i}^{i+\alpha-1} B_x p^{\alpha-x+i-1} - B_i p^{\alpha-1} \right] p + B_{i+\alpha} \right\} \bmod D \\
&\quad = \Big\{ \left[ \mathrm{Rabin}(B_i, B_{i+1}, \ldots, B_{i+\alpha-1}) - B_i p^{\alpha-1} \right] p \\
&\qquad\quad + B_{i+\alpha} \Big\} \bmod D.
\end{aligned}
$$



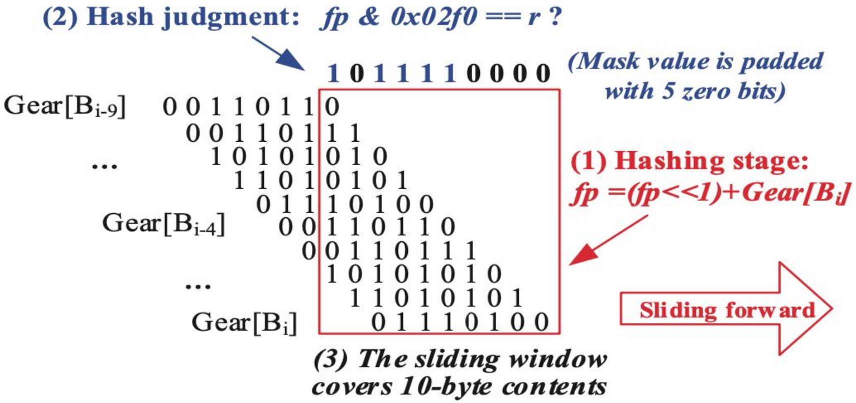
# Problems of Rabin-based CDC

- BSW with rabin-based
  - Size : High chunk size variance
  - Speed : Time consuming and heavy CPU overhead
  - Deduplication ratio : Inaccuracy of duplicate detection
- Two Thresholds Two Divisors (TTTD)
  - max/min chunk size threshold
- Gear
  - Improve speed
  - Small sliding window size
  - Reducing hash calculation by a pre-defined random integer table
- Subchunk
  - Re-chunking unique chunks

# FastCDC by Xia et al.[ATC’16]

- 3 observations of Gear-based CDC
  - Fast hashing  (sliding window size is small)
  - Hash judgement becomes new bottleneck   “*fp mod D==r*”
  - Skipping cut-points can speed up chunking process at the cost of decreasing dedup ratio
- FastCDC techniques
  - Simplified but enhanced hash judgment

    “*!fp & Mask*”.
  - Sub-minimum chunk cut-point skipping
  - Normalized chunking



# Can We Do Better?

Does the CDC really cut at the “perfect” cut-point? What is the “ideal” way to do CDC?

1. Identify large duplicate chunks
   a. Less metadata in indexing table
   b. Faster restore speed
2. Identify smaller chunks with high number of duplicates
3. Unique chunks that rarely appears

# Question?

Let’s take it offline.