Fast SIMD-Based Chunking Algorithm

Yehonatan Dude, Michael Hirsch, Yair Toaff

toganetworks a HUAWEI company

PSC 2019

Outline

1. Background
2. Chunking Problem
3. Traditional Solutions
4. Our Solution
5. Future Work

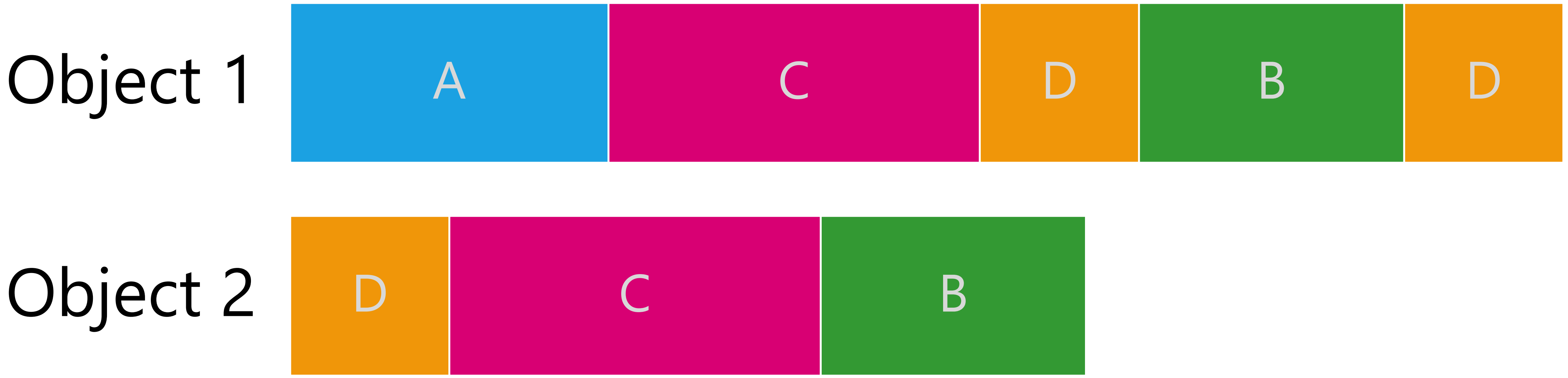
Background - Deduplication

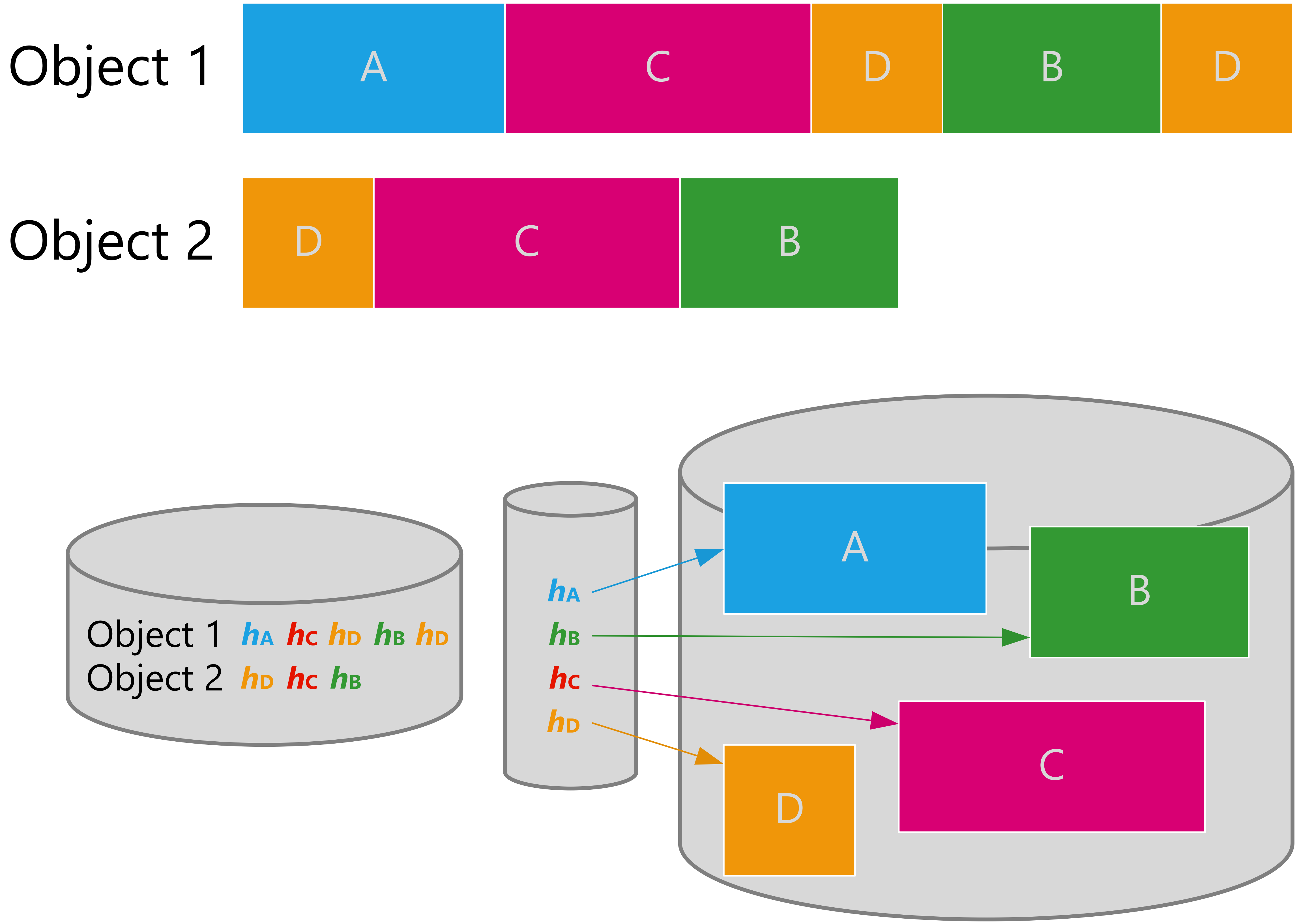
Deduplication is a technique for eliminating duplicate copies of repeating data.

Deduplication process in a nutshell

1. Divide into chunks
2. Calculate the chunks' hashes
3. Store chunks uniquely


Object 1
Object 2


Object 1
A
C
D
B
D
Object 2
D
C
B



Object 1
A
C
D
B
D
Object 2
D
C
B
Object 1 hA hC hD hB hD
Object 2 hD hC hB
hA
hB
hC
hD
A
B
C
D

Background - Chunking Methods

How to chunk the input data

1. Simple - fixed size.
2. Content aware - files, objects, applications.
3. Content sensitive - rolling hash.

The Opera ghost really existed._	He was not, as was long believed
, a creature of the imagination_	of the artists, the superstition
of the managers, or a product o	f the *joy* and impressionable bra
ins of the young ladies of the b	allet, their mothers, the box-ke
epers, the cloak-room attendants	or the concierge. Yes, he exist
ed in flesh and blood, although_	he assumed the complete appearan
ce of a real phantom; that is to	say, of a spectral shade.

The Opera ghost really existed._	He was not, as was long believed
, a creature of the imagination_	of the artists, the superstition
of the managers, or a product o	f the *absurd* and impressionable
brains of the young ladies of th	e ballet, their mothers, the box
-keepers, the cloak-room attenda	nts or the concierge. Yes, he ex
isted in flesh and blood, althou	gh he assumed the complete appea
rance of a real phantom; that is	to say, of a spectral shade.

The Opera ghost really existed.
He was not, as was long believed, a creature of the imagination of the artists, the superstition of the managers, or a product of the *joy* and impressionable brains of the young ladies of the ballet, their mothers, the box-keepers, the cloak-room attendants or the concierge.
Yes, he existed in flesh and blood, although he assumed the complete appearance of a real phantom; that is to say, of a spectral shade.

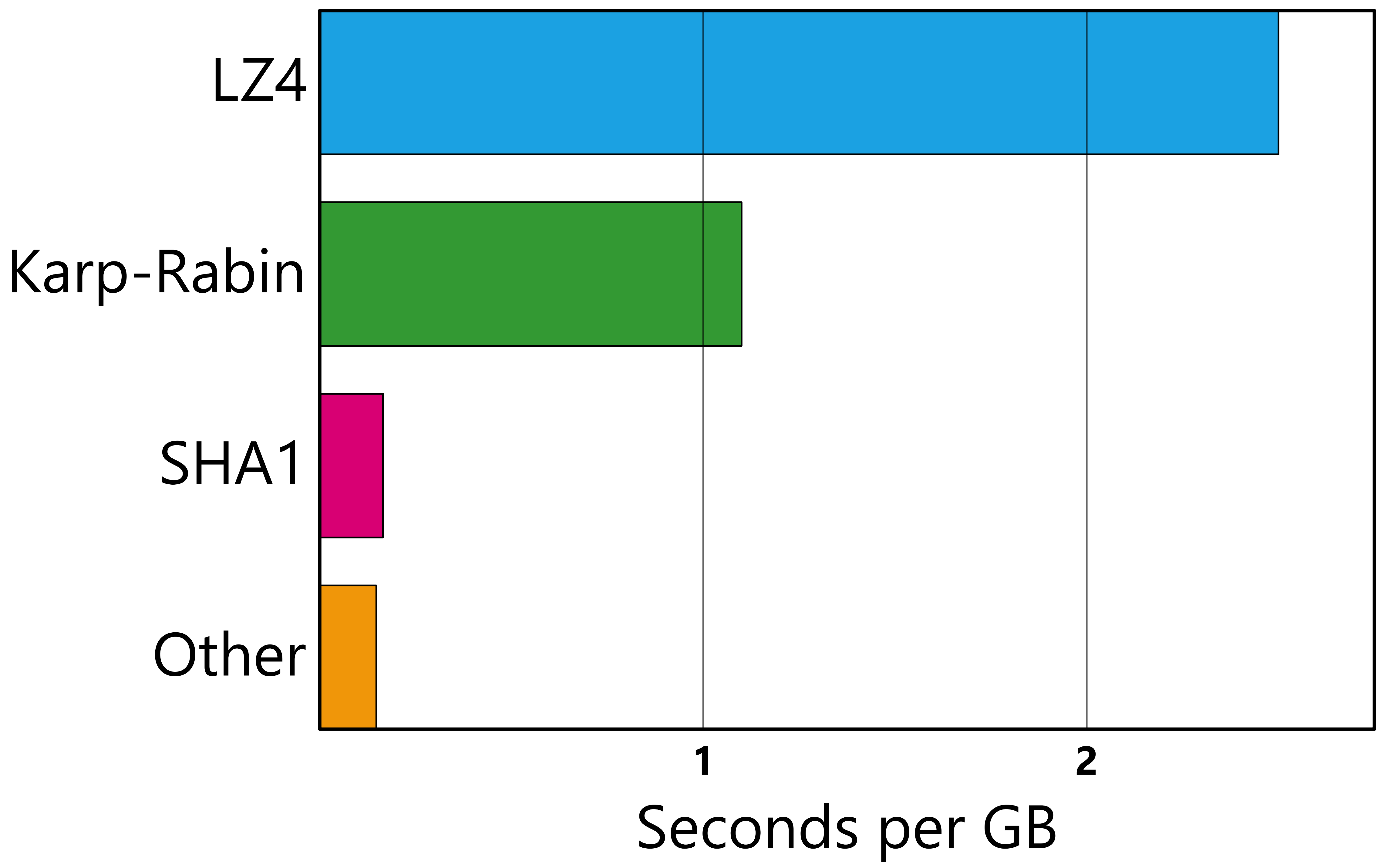
The Opera ghost really existed.
He was not, as was long believed, a creature of the imagination of the artists, the superstition of the managers, or a product of the *absurd* and impressionable brains of the young ladies of the ballet, their mothers, the box-keepers, the cloak-room attendants or the concierge.
Yes, he existed in flesh and blood, although he assumed the complete appearance of a real phantom; that is to say, of a spectral shade.

The Opera ghost really existed. He was not, as was long believed
, a creature of the imagination of the artists, the superstition
of the managers, or a product of the *joy* and impressionable bra
ins of the young ladies of the ballet, their mothers, the box-ke
epers, the cloak-room attendants or the concierge. Yes, he exist
ed in flesh and blood, although he assumed the complete appearan
ce of a real phantom; that is to say, of a spectral shade.

The Opera ghost really existed. He was not, as was long believed
, a creature of the imagination of the artists, the superstition
of the managers, or a product of the *absurd* and impressionable
brains of the young ladies of the ballet, their mothers, the box
-keepers, the cloak-room attendants or the concierge. Yes, he ex
isted in flesh and blood, although he assumed the complete appea
rance of a real phantom; that is to say, of a spectral shade.

Background - Deduplication Performance

In 2017 we worked on a deduplication engine,
and we tried to improve its performance.


LZ4
Karp-Rabin
SHA1
Other
1
2
Seconds per GB

Chunking Problem

Given a stream of bytes, divide it into chunks for deduplication.

1. Output identical chunks for identical data
2. Good chunk size distribution.
3. Good performance.
4. Works for any input (photo, DB, text, random, etc...)

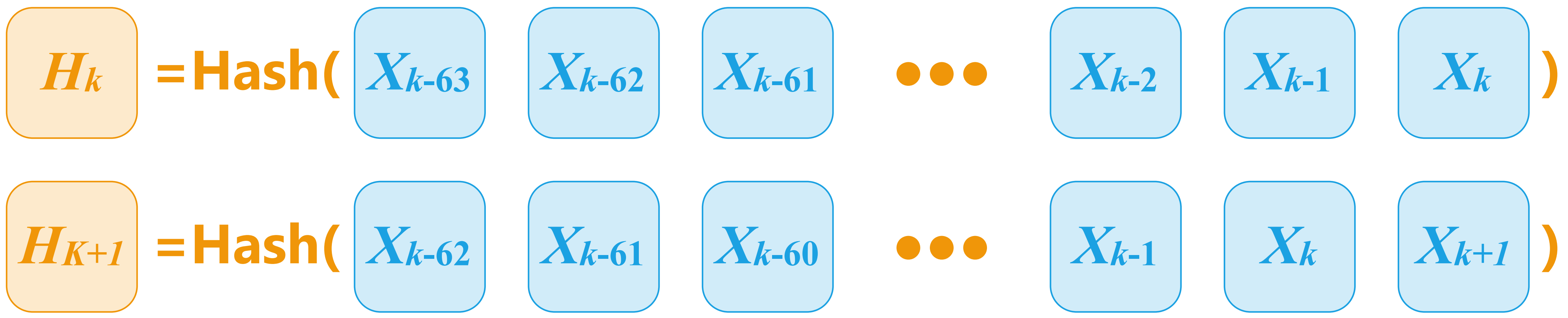
Traditional Solutions

Karp-Rabin

Cyclic Polynomial

$\boxed{H_k} = \mathrm{Hash}(\boxed{X_{k-63}}\ \boxed{X_{k-62}}\ \boxed{X_{k-61}}\ \cdots\ \boxed{X_{k-2}}\ \boxed{X_{k-1}}\ \boxed{X_k})$

If Criteria($\boxed{H_k}$) holds then mark a boundary after X_k



H_k =Hash(X_{k-63} X_{k-62} X_{k-61} ••• X_{k-2} X_{k-1} X_k)
H_{K+1} =Hash(X_{k-62} X_{k-61} X_{k-60} ••• X_{k-1} X_k X_{k+1})

$$H_k = \mathrm{Hash}(X_{k-63}\ X_{k-62}\ X_{k-61}\ \cdots\ X_{k-2}\ X_{k-1}\ X_k)$$

$$H_{K+1} = \mathrm{Hash}(X_{k-62}\ X_{k-61}\ X_{k-60}\ \cdots\ X_{k-1}\ X_k\ X_{k+1})$$

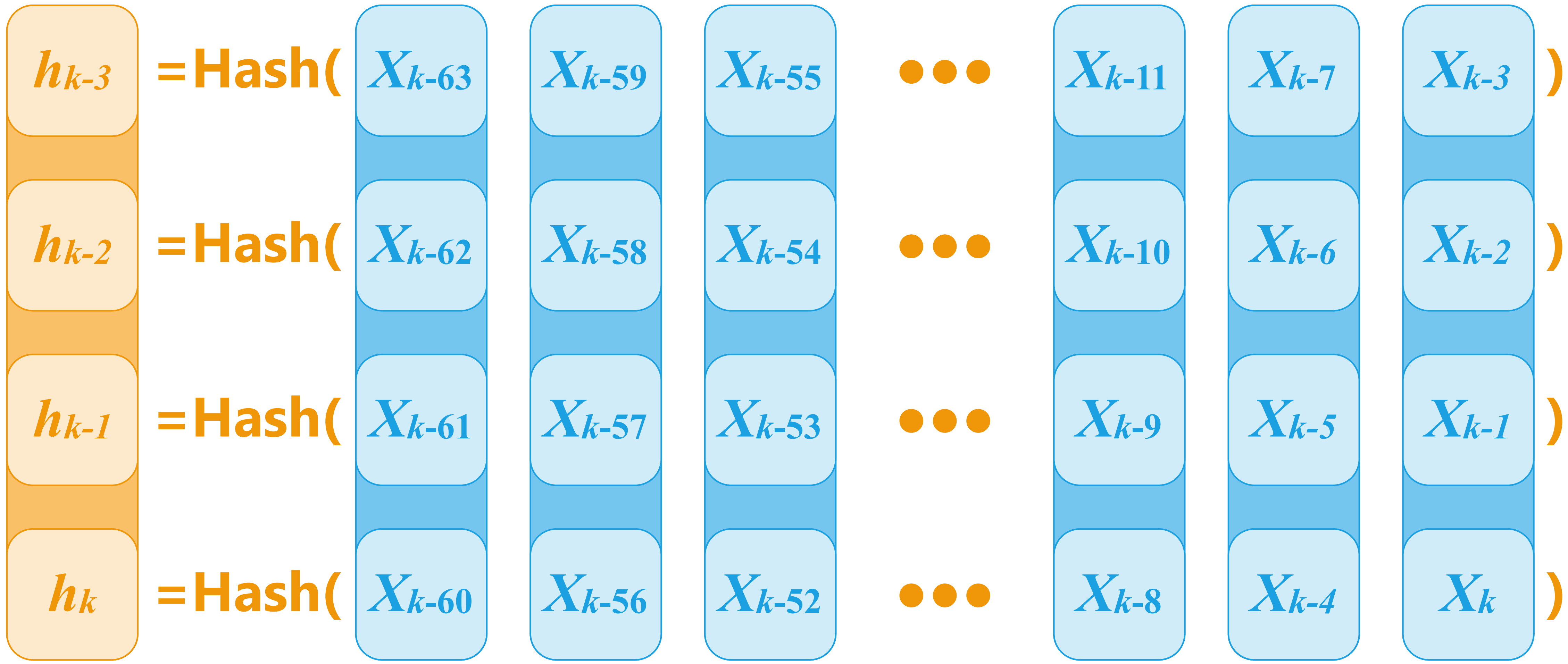
$$= \mathrm{RollHash}(X_{k-63}, H_k, X_{k+1})$$

Karp-Rabin	Cyclic Polynomial
$h_i = \sum_{j=0}^{j=63} p^j x_{i-j} \mod N$	$h_i = \bigoplus_{j=0}^{j=63} \mathrm{rotate}(x_{i-j}, j)$
$h_{i+1} = p^{64} x_{i-64} + p h_i + x_i \mod N$	$h_{i+1} = x_{i-64} + \mathrm{rotate}(h_i, 1) + x_i$

Proposed Solution

How does it work:

1. Work with rolling vectors
2. Calculate a hash of byte size
3. Calculate the criteria, in a way that:
   - Number of calculations are constant unrelated to the vector size
   - Can find a cutting point at a byte offset



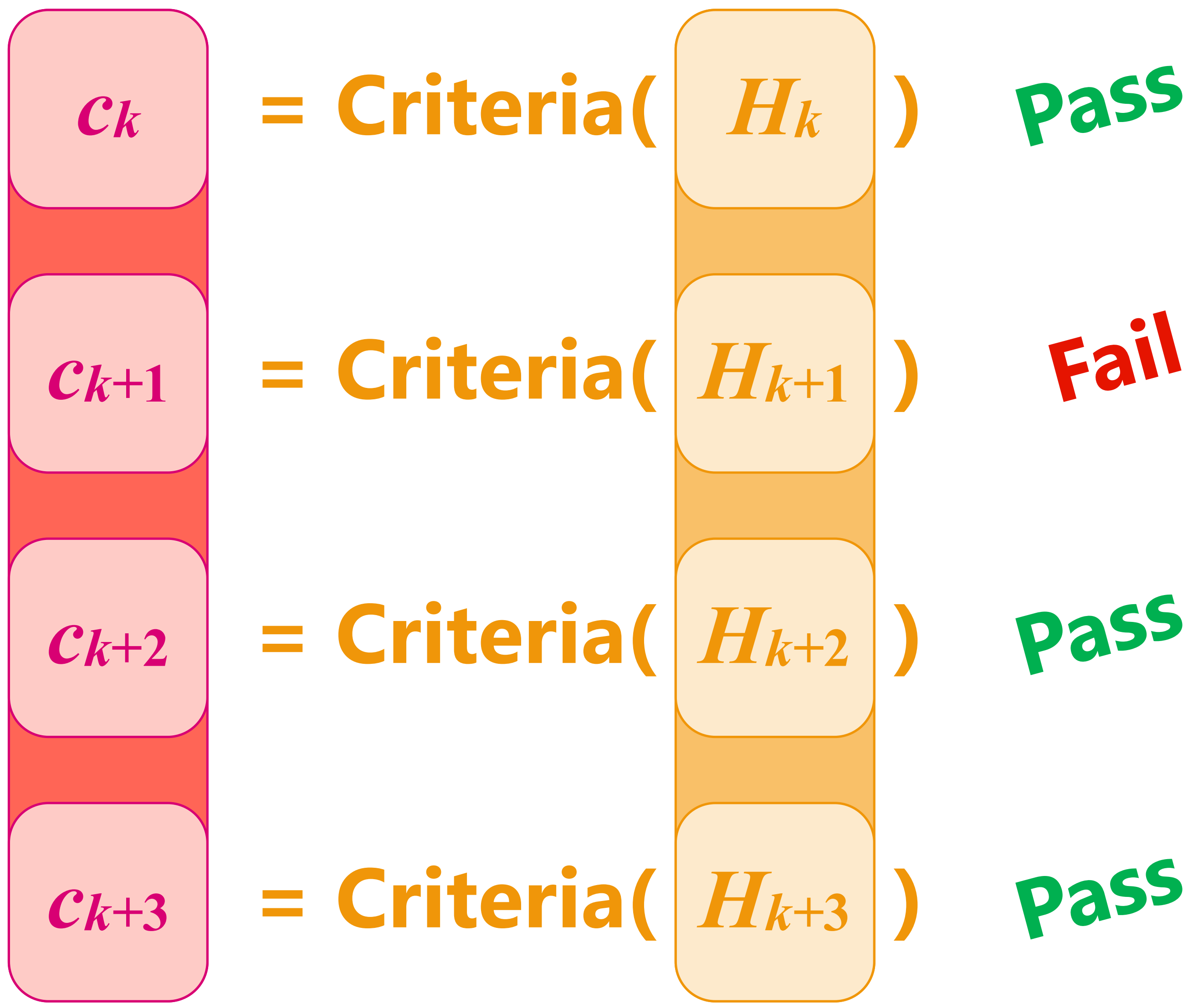
h_{k-3} =Hash(X_{k-63} X_{k-59} X_{k-55} ••• X_{k-11} X_{k-7} X_{k-3})
h_{k-2} =Hash(X_{k-62} X_{k-58} X_{k-54} ••• X_{k-10} X_{k-6} X_{k-2})
h_{k-1} =Hash(X_{k-61} X_{k-57} X_{k-53} ••• X_{k-9} X_{k-5} X_{k-1})
h_{k} =Hash(X_{k-60} X_{k-56} X_{k-52} ••• X_{k-8} X_{k-4} X_{k})

$$h_{k+1} = \mathrm{Hash}(X_{k-59}\ \ X_{k-55}\ \cdots\ X_{k-3}\ \ X_{k+1}) = \mathrm{RollHash}(X_{k-63},\ h_{k-3},\ X_{k+1})$$

$$h_{k+2} = \mathrm{Hash}(X_{k-58}\ \ X_{k-54}\ \cdots\ X_{k-2}\ \ X_{k+2}) = \mathrm{RollHash}(X_{k-62},\ h_{k-2},\ X_{k+2})$$

$$h_{k+3} = \mathrm{Hash}(X_{k-57}\ \ X_{k-53}\ \cdots\ X_{k-1}\ \ X_{k+3}) = \mathrm{RollHash}(X_{k-61},\ h_{k-1},\ X_{k+3})$$

$$h_{k+4} = \mathrm{Hash}(X_{k-56}\ \ X_{k-52}\ \cdots\ X_{k}\ \ X_{k+4}) = \mathrm{RollHash}(X_{k-60},\ h_{k},\ X_{k+4})$$



c_k = Criteria(H_k) Pass
c_{k+1} = Criteria(H_{k+1}) Fail
c_{k+2} = Criteria(H_{k+2}) Pass
c_{k+3} = Criteria(H_{k+3}) Pass

c_k = Criteria(H_k) Pass

c_{k+1} = Criteria(H_{k+1}) Fail

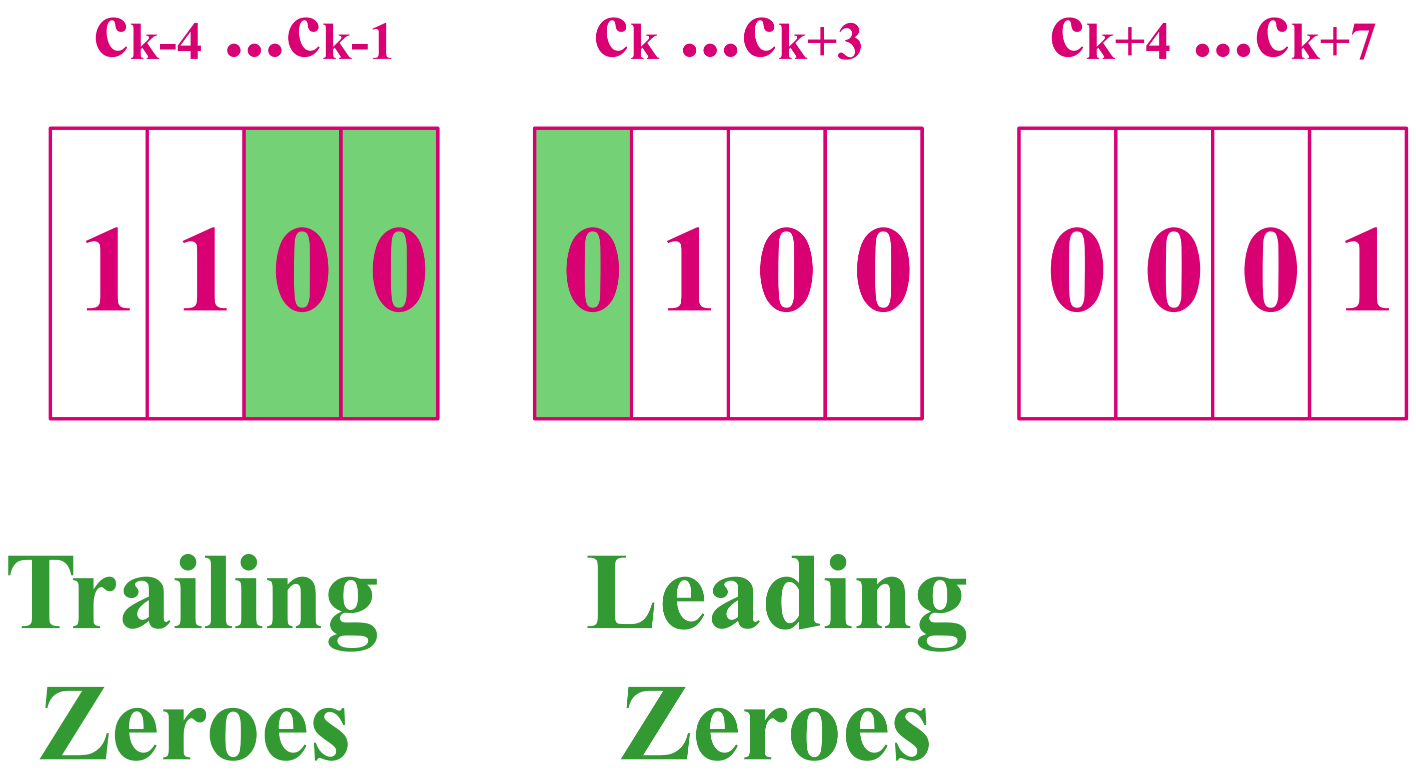
c_{k+2} = Criteria(H_{k+2}) Pass

c_{k+3} = Criteria(H_{k+3}) Pass

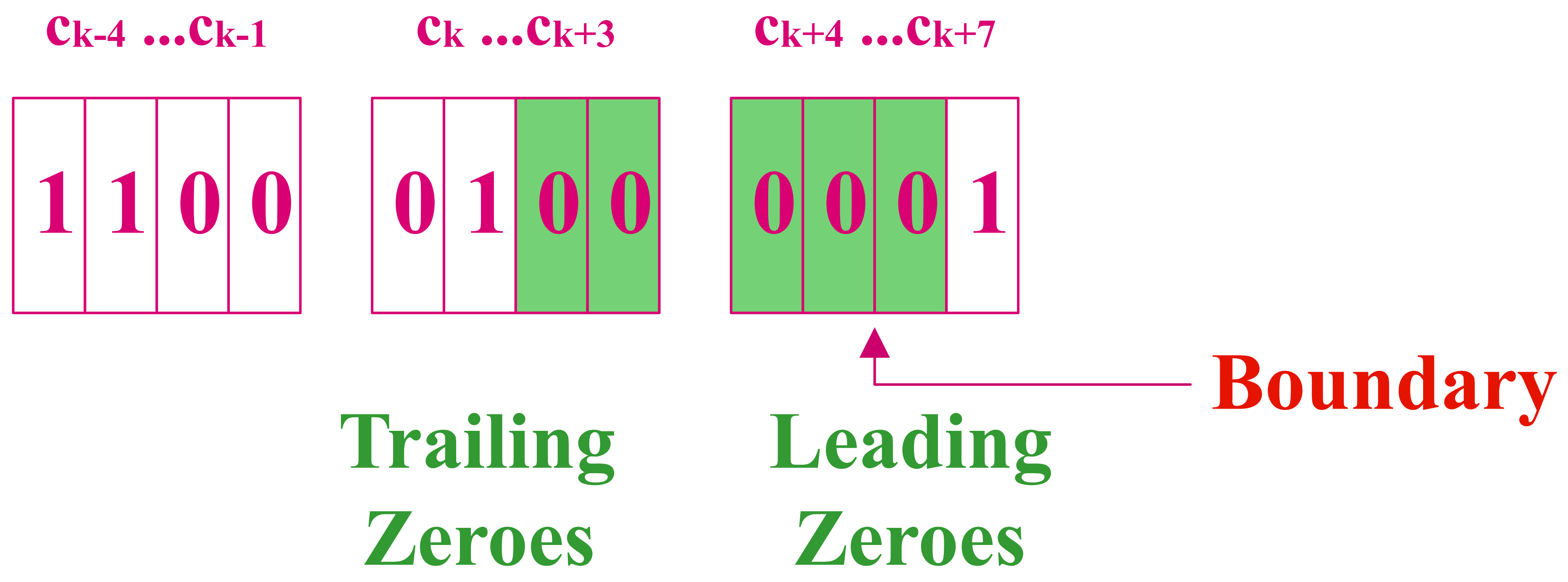
$c_k \ldots c_{k+3}$

0	1	0	0

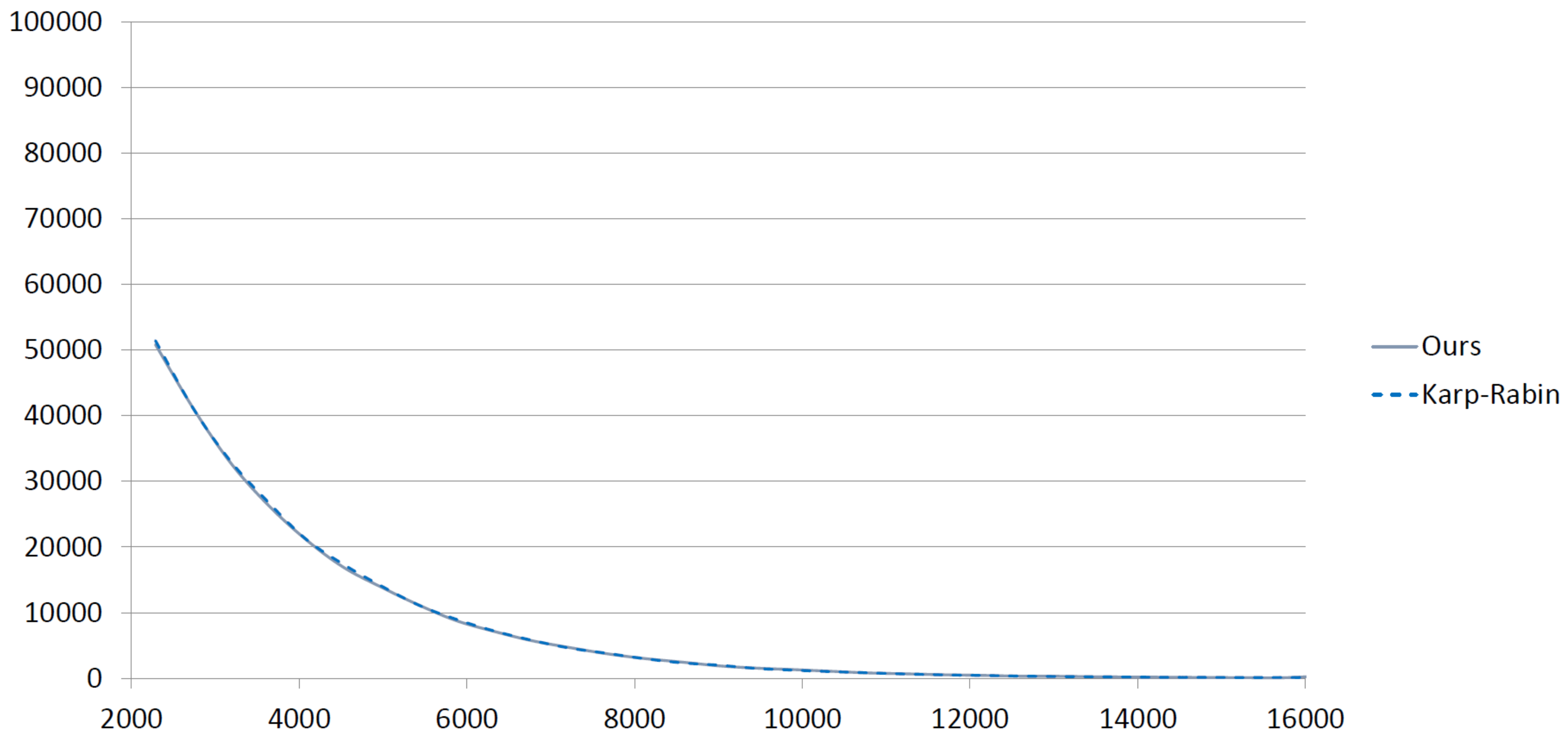
$c_{k-4} \ldots c_{k-1}$	$c_k \ldots c_{k+3}$	$c_{k+4} \ldots c_{k+7}$
1 1 0 0	0 1 0 0	0 0 0 1



c_{k-4} ...c_{k-1}
c_k ...c_{k+3}
c_{k+4} ...c_{k+7}
1 1 0 0
0 1 0 0
0 0 0 1
Trailing Zeroes
Leading Zeroes


ck-4 ...ck-1
ck ...ck+3
ck+4 ...ck+7
1 1 0 0
0 1 0 0
0 0 0 1
Boundary
Trailing Zeroes
Leading Zeroes

Measured Results



Algorithm	Random Data	Corpus Data
Karp-Rabin	975 MB/s	927 MB/s
Cyclic-Polynomial	1675 MB/s	1676 MB/s
Ours	6715 MB/s	7136 MB/s

Chunking Alg.	*Dedup Perf.*	LZ4	SHA1	Other	Chunking
Karp-Rabin	*262 MB/s*	63.9%	4.1%	3.8%	28.1%
Ours	*345 MB/s*	84.5%	5.4%	5.1%	4.7%

- Same Distribution
- Faster Chunking Performance
- Faster Overall Performance

Future Work

A chunking algorithm that is

*	Past	Presented	Future
Backward Compatible	N/A	$f(k, x) \neq f(l, x)$	$f(k, x) = f(l, x)$
Work	cn	cn	cn
Speed	cn	c(n log k) / k	c(n log k) / k

Thanks

https://github.com/dudejohnny/PSC2019