Fast SIMD-Based Chunking Algorithm

Yehonatan Dude, Michael Hirsch, Yair Toaff

toganetworks

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
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 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 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 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.
In 2017 we worked on a deduplication engine, and we tried to improve its performance.
Given a stream of bytes, divide it into chunks for deduplication.

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

**Chunking Problem**
Traditional Solutions

Karp-Rabin
Cyclic Polynomial
Fast SIMD-Based Chunking Algorithm

Johnny Dude
$H_k = \text{Hash}(X_{k-63}, X_{k-62}, X_{k-61}, \ldots, X_{k-2}, X_{k-1}, X_k)$

If \text{Criteria}(H_k) holds then mark a boundary after $X_k$
\[ H_k = \text{Hash}(X_{k-63}, X_{k-62}, X_{k-61}, \ldots, X_{k-2}, X_{k-1}, X_k) \]

\[ H_{K+1} = \text{Hash}(X_{k-62}, X_{k-61}, X_{k-60}, \ldots, X_{k-1}, X_k, X_{k+1}) \]
$H_k = \text{Hash}(X_{k-63}, X_{k-62}, X_{k-61}, \ldots, X_{k-2}, X_{k-1}, X_k)$

$H_{K+1} = \text{Hash}(X_{k-62}, X_{k-61}, X_{k-60}, \ldots, X_{k-1}, X_k, X_{k+1})$

$= \text{RollHash}(X_{k-63}, H_k, X_{k+1})$
### Karp-Rabin

\[ h_i = \sum_{j=0}^{j=63} p^j x_{i-j} \mod N \]

\[ h_{i+1} = p^{64} x_{i-64} + ph_i + x_i \mod N \]

### Cyclic Polynomial

\[ h_i = \bigoplus_{j=0}^{j=63} \text{rotate}(x_{i-j}, j) \]

\[ h_{i+1} = x_{i-64} + \text{rotate}(h_i, 1) + x_i \]
**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
Fast SIMD-Based Chunking Algorithm

Johnny Dude
$h_{k-3} = \text{Hash}(X_{k-63} \ X_{k-59} \ X_{k-55} \ \cdots \ X_{k-11} \ X_{k-7} \ X_{k-3})$

$h_{k-2} = \text{Hash}(X_{k-62} \ X_{k-58} \ X_{k-54} \ \cdots \ X_{k-10} \ X_{k-6} \ X_{k-2})$

$h_{k-1} = \text{Hash}(X_{k-61} \ X_{k-57} \ X_{k-53} \ \cdots \ X_{k-9} \ X_{k-5} \ X_{k-1})$

$h_k = \text{Hash}(X_{k-60} \ X_{k-56} \ X_{k-52} \ \cdots \ X_{k-8} \ X_{k-4} \ X_k)$
$h_{k+1} = \text{Hash}(X_{k-59}, X_{k-55}, \ldots, X_{k-3}, X_{k+1}) = \text{RollHash}(X_{k-63}, h_{k-3}, X_{k+1})$

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

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

$h_{k+4} = \text{Hash}(X_{k-56}, X_{k-52}, \ldots, X_{k}, X_{k+4}) = \text{RollHash}(X_{k-60}, h_{k}, X_{k+4})$
\[ C_k = \text{Criteria}(H_k) \quad \text{Pass} \]
\[ C_{k+1} = \text{Criteria}(H_{k+1}) \quad \text{Fail} \]
\[ C_{k+2} = \text{Criteria}(H_{k+2}) \quad \text{Pass} \]
\[ C_{k+3} = \text{Criteria}(H_{k+3}) \quad \text{Pass} \]
\[ C_k = \text{Criteria}(H_k) \quad \text{Pass} \]
\[ C_{k+1} = \text{Criteria}(H_{k+1}) \quad \text{Fail} \]
\[ C_{k+2} = \text{Criteria}(H_{k+2}) \quad \text{Pass} \]
\[ C_{k+3} = \text{Criteria}(H_{k+3}) \quad \text{Pass} \]
Fast SIMD-Based Chunking Algorithm

Trailing Zeroes

1100

Leading Zeroes

0100

0001

C_{k-4} ... C_{k-1}  C_k ... C_{k+3}  C_{k+4} ... C_{k+7}
Trailing Zeroes

Leading Zeroes

Boundary
Measured Results
<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Random Data</th>
<th>Corpus Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Karp-Rabin</td>
<td>975 MB/s</td>
<td>927 MB/s</td>
</tr>
<tr>
<td>Cyclic-Polynomial</td>
<td>1675 MB/s</td>
<td>1676 MB/s</td>
</tr>
<tr>
<td>Ours</td>
<td>6715 MB/s</td>
<td>7136 MB/s</td>
</tr>
<tr>
<td>Chunking Alg.</td>
<td>Dedup Perf.</td>
<td>LZ4</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------</td>
<td>-----</td>
</tr>
<tr>
<td>Karp-Rabin</td>
<td>262 MB/s</td>
<td>63.9%</td>
</tr>
<tr>
<td>Ours</td>
<td>345 MB/s</td>
<td>84.5%</td>
</tr>
</tbody>
</table>
• Same Distribution
• Faster Chunking Performance
• Faster Overall Performance
## Future Work

*A chunking algorithm that is*

<table>
<thead>
<tr>
<th></th>
<th>Past</th>
<th>Presented</th>
<th>Future</th>
</tr>
</thead>
<tbody>
<tr>
<td>Backward Compatible</td>
<td>N/A</td>
<td>$f(k, x) \neq f(l, x)$</td>
<td>$f(k, x) = f(l, x)$</td>
</tr>
<tr>
<td>Work</td>
<td>$cn$</td>
<td>$cn$</td>
<td>$cn$</td>
</tr>
<tr>
<td>Speed</td>
<td>$cn$</td>
<td>$c(n \log k) / k$</td>
<td>$c(n \log k) / k$</td>
</tr>
</tbody>
</table>
Thanks

https://github.com/dudejohnny/PSC2019