EndRE: An End-System Redundancy Elimination Service For Enterprises

Bhavish Aggarwal, Aditya Akella, Ashok Anand1, Athula Balachandran1, Pushkar Chitnis, Chitra Muthukrishnan, Ramachandran Ramjee and George Varghese

Microsoft Research India; University of Wisconsin-Madison; CMU; UCSD

SLIDES BY SHIMON AZULAY & AYAL MITTELMAN
Introduction

- Nowadays, network services have reached a global scale in enterprise space
- Data between clients is transferred over WAN
- Data should be transferred from end to end clients quickly and efficiently for better user experience
Globalization And Network Service Example

- In a global scale corporation, branch offices can be found all over the globe
- Where should they locate their servers?

What is better?
Example – cont.

Tradeoff:

- Servers that are located near the clients are much more efficient in data exchange, but the operational costs for them are high.
- Servers at a small number of locations can lower administration costs, but increase network costs and latency.
Middlebox & Protocol Independent redundancy elimination

We use Middleboxes

- Performance in WAN communication
- One box detects chunks of data that match entries in its cache (by computing fingerprints), it encodes matches using tokens
- Box at the far end reconstructs original data using its own cache and the tokens
Middle box Drawbacks

- **Encrypted data:**
  - Encrypted data could not be found in the middlebox cache, although the decrypted data exists in the cache.
  - Data encrypted, then was sent to middlebox, which need to decrypt it - Not safe and redundant.

- **Usage of mobile devices:**
  - Token reached the middlebox, found in the cache and data and was reconstructed. Now the bottleneck is between the router and the mobile phone.
End-System Redundancy Elimination

In this presentation we will explore end-system redundancy elimination service called EndRE. EndRE could supplement or supplant middle box-based techniques while addressing their drawbacks. We will examine the changes in design and implementation in order to support EndRE.
EndRE—Design Goals

We will examine five design goals for the new approach:

- Transparent operation
- Fine-grained operation
- Simple decoding at clients
- Fast and adaptive encoding at servers
- Limited memory footprint at servers and clients
EndRE Design is divided into two modules:

- **Server**
  - Encoding the redundant data with shorter meta-data
  - Meta-data is essentially a set of `<offset, length>`
  - The meta-data computed with respect to the client-side cache
EndRE Design is divided into two modules:

- **Client**
  - Consist simple logic to decode the meta-data by “de-referencing” the offsets sent by the server.
EndRE design - Handle Redundancy

For handling the redundancy we need to do two steps:

- **Fingerprinting (4 Approaches):**
  - MODP
  - MAXP
  - FIXED
  - SAMPLEBYTE

- **Matching and Encoding (2 Approaches):**
  - Chunk-Match
  - Max-Match
EndRE-Terminology

Data Block

Marker

Minimum 32 Bytes

Chunk

Marker

Byte

1010...0010010111010110101000001101110010110111010101110101011011111101010111011101
EndRE-Terminology

Chunk

DATA

WHAT IF I TOLD YOU
I AM A HASH FUNCTION
EndRE design-MODP Fingerprinting

Data Block

Fingerprints

X Bytes  X Bytes  X Bytes  X Bytes

1 Byte

RabinHash

Over window size
W = 32 Bytes
EndRE design-
MODP Fingerprinting

Fingerprints

Is this fingerprint mod p == 0 ?

Original Data Block
EndRE design-
MODP Fingerprinting – Cont.

- Content based
- Expensive computational operations
- Over/under sampling
EndRE design-
MAXP Fingerprinting

Data Block

Maximum Byte size?

1 Byte

W = 32 Bytes
EndRE design - MAXP Fingerprinting – Cont.

- Content based
- No over/under sampling
- Expensive computational operations
EndRE design-FIXED Fingerprinting

Data Block

Markers

1 Byte

FIXED SIZE
EndRE design-
MODP Fingerprinting – Cont.

- Cheap computational operations
- No over/under sampling

- Not robust to small changes
EndRE design - SAMPLEBYTE Fingerprinting

Data Block  Skip p/2 Bytes  Data Block

<table>
<thead>
<tr>
<th>Byte</th>
<th>bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000000</td>
<td>1</td>
</tr>
<tr>
<td>00000001</td>
<td>1</td>
</tr>
<tr>
<td>00000010</td>
<td>0</td>
</tr>
<tr>
<td>00000100</td>
<td>0</td>
</tr>
</tbody>
</table>

11111101  11111111  00000001
EndRE design - SAMPLEBYTE Fingerprinting – Cont.

- Content-Based
- Computationally efficient

WHAT ABOUT OVER/UNDER SAMPLING?
SAMPLEBYTE Fingerprinting
Over/Under sampling

- Skips $P/2$ bytes after match
- Table is built in a way that match will be occur every $1/P$ bytes
We build static lookup table:

- Use Network traces from one of the enterprise sites
- Run MAXP to identify redundant content
- Sort characters in descending order of their presence in the identified redundant content
- Set the first x to 1
EndRE design-Matching And Encoding

We examine 2 Approaches:

- Chunk-Match
- Max-Match

For both approaches we will try to:

- Move computationally operations & memory management tasks to the server
- Exploit inherent structure within the data to optimize memory usage
EndRE design-Matching Overview

Server

Index

chunk

chunk

chunk

Client
Matching And Encoding –

In other systems:
- Client saves hash chunk mapping
- Server sends the hash to the client
- Server holds chunks

In EndRE:
- Client hold simple circular FIFO cache. Doesn’t hold the hash function
- Server hold hash - <offset, length> table
- Server sends <offset, length> tuple
Matching And Encoding – Chunk-Match
Matching And Encoding – Chunk-Match

<table>
<thead>
<tr>
<th>Index</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>&lt;Offset, length&gt;</td>
</tr>
<tr>
<td>#</td>
<td>&lt;Offset, length&gt;</td>
</tr>
<tr>
<td>#</td>
<td>&lt;Offset, length&gt;</td>
</tr>
</tbody>
</table>

Client

First In, First Out

FIFO

chunk

chunk

chunk

Cache
Chunk Match - Optimization

Assume:

- $P = 64$
- Cache Size (client) = 16MB = $2^{24}$ Bytes
- Maximum Chunk Size = 256 Bytes

We only need to store:

$$< \text{SHA1}, \text{Offset}, \text{Length} >$$

Server Holds 38% of the client cache size
Matching And Encoding – Max-Match

Server

Client

Index Fingerprint | Value Offset
-- | --
--- | ---
--- | ---
--- | ---

Compute <Offset, length>

Server Memory

---
Matching And Encoding – Max-Match

Server

Client

Index Fingerprint | Value Offset
--- | ---

Server Memory

Compute <Offset, length>
Max-Match - Optimization

- Client cache size of 16MB = $2^{24}$
- $P = 64 = 2^6$ bytes
- $2^{18}$ fingerprints
- Add additional 8 bits to fingerprint index column
- Server holds table of size 6% of the client cache size
- Server holds in total 106% of the client cache size

![Figure 5: Max-Match: matched region is expanded]
Implementation - Socket Layer Above TCP

Benefits of implementing EndRE at the socket layer above TCP:

- **Latency** – Reduce the number of packets
- **Encryption** – Can be compressed before encryption
- **Cache Synchronization**: TCP ensure reliable in-order delivery. However, TCP connections may get reset in the middle of a transfer. 2 Solutions:
  - Pessimistic
  - Optimistic
Evaluation

- 11 corporate enterprise locations (classified as small, medium or large)
- Small pilot deployment (15 laptops) in their lab

<table>
<thead>
<tr>
<th>Trace Name</th>
<th>Unique Client IPs</th>
<th>Dates (Total Days)</th>
<th>Size (TB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small Enterprise</td>
<td>29-39</td>
<td>07/28/08 - 08/08/08 (11) 11/07/08 - 12/10/08 (33)</td>
<td>0.5</td>
</tr>
<tr>
<td>(Sites 1-2)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Medium Enterprise</td>
<td>62-91</td>
<td>07/28/08 - 08/08/08 (11) 11/07/08 - 12/10/08 (33)</td>
<td>1.5</td>
</tr>
<tr>
<td>(Sites 3-6)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Large Enterprise</td>
<td>101-210</td>
<td>07/28/08 - 08/08/08 (11) 11/07/08 - 12/10/08 (33)</td>
<td>3</td>
</tr>
<tr>
<td>(Sites 7-10)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Large Research Lab</td>
<td>125</td>
<td>06/23/08 - 07/03/08 (11)</td>
<td>1</td>
</tr>
<tr>
<td>(Site 11, training trace)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Server CPU And Bandwidth Costs
### CPU Costs - Server

<table>
<thead>
<tr>
<th></th>
<th>Max-Match</th>
<th>Fingerprint</th>
<th>InlineMatch</th>
<th>Admin</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$p \rightarrow$</td>
<td>32</td>
<td>512</td>
<td>32</td>
</tr>
<tr>
<td>MODP</td>
<td>526.7</td>
<td>496.7</td>
<td>9.6</td>
<td>6.8</td>
</tr>
<tr>
<td>MAXP</td>
<td>306.3</td>
<td>118.8</td>
<td>10.1</td>
<td>7.7</td>
</tr>
<tr>
<td>FIXED</td>
<td>69.4</td>
<td>14.2</td>
<td>7.1</td>
<td>4.7</td>
</tr>
<tr>
<td>SAMPLEBYTE(SB)</td>
<td>76.8</td>
<td>20.2</td>
<td>9.5</td>
<td>6.1</td>
</tr>
</tbody>
</table>

**CPU Time(s) for different algorithms**
Memory Costs

Two key questions:

1. What is the cache size limit between a single client – server pair?

2. Given the cache size limit for one pair, what is the cumulative memory requirement at clients & sever

- We will examine the trade-off between cache sizes and bandwidth savings
Memory Costs

- SMB
- NetBiosFS
- HTTP
- LDAP
- OTHERS

Bandwidth Savings (%) vs. Cache Size (KB), log-scale
Memory Costs

Figure 10: Cache scalability
## Bandwidth Savings

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>173</td>
<td>9</td>
<td>47</td>
<td>47</td>
<td>48</td>
<td>36</td>
<td>71</td>
<td>72</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>14</td>
<td>24</td>
<td>20</td>
<td>24</td>
<td>19</td>
<td>33</td>
<td>33</td>
</tr>
<tr>
<td>3</td>
<td>71</td>
<td>17</td>
<td>25</td>
<td>23</td>
<td>26</td>
<td>22</td>
<td>34</td>
<td>35</td>
</tr>
<tr>
<td>4</td>
<td>58</td>
<td>17</td>
<td>23</td>
<td>20</td>
<td>24</td>
<td>21</td>
<td>45</td>
<td>47</td>
</tr>
<tr>
<td>5</td>
<td>69</td>
<td>15</td>
<td>26</td>
<td>22</td>
<td>27</td>
<td>21</td>
<td>39</td>
<td>42</td>
</tr>
<tr>
<td>6</td>
<td>80</td>
<td>12</td>
<td>21</td>
<td>18</td>
<td>22</td>
<td>17</td>
<td>34</td>
<td>36</td>
</tr>
<tr>
<td>7</td>
<td>80</td>
<td>12</td>
<td>21</td>
<td>18</td>
<td>22</td>
<td>17</td>
<td>34</td>
<td>36</td>
</tr>
<tr>
<td>8</td>
<td>142</td>
<td>14</td>
<td>22</td>
<td>18</td>
<td>22</td>
<td>17</td>
<td>34</td>
<td>36</td>
</tr>
<tr>
<td>9</td>
<td>198</td>
<td>9</td>
<td>16</td>
<td>16</td>
<td>16</td>
<td>15</td>
<td>44</td>
<td>46</td>
</tr>
<tr>
<td>10</td>
<td>117</td>
<td>13</td>
<td>20</td>
<td>17</td>
<td>21</td>
<td>17</td>
<td>27</td>
<td>30</td>
</tr>
<tr>
<td>Avg/site</td>
<td>100</td>
<td>13</td>
<td>25</td>
<td>19</td>
<td>26</td>
<td>22</td>
<td>34</td>
<td>39</td>
</tr>
</tbody>
</table>

**Note:** Server bandwidth savings are shown in the table above, indicating the efficiency of different compression methods and server-client interactions.
Energy Savings

(a) Compression Savings

(b) Energy
Questions?