Cloud Assisted Traffic Redundancy Elimination for Power Efficiency in Smartphones

Shenghua He*, Haiying Shen**, VivekGautham Soundararaj† and Lei Yu ++

*Department of Computer Science and Engineering, Washing University in St. Louis, MO, USA
** Department of CS, University of Virginia, Charlottesville, VA, USA
† Department of ECE, Clemson University, Clemson, SC, USA
++ College of Computing, Georgia Institute of Technology, Atlanta, GA, USA
Outline

• Background and contributions
• System design
• Performance evaluation
• Conclusions
Background: Exceptional increase in smartphone usage

A massive increase in data traffic from application servers to smartphones

- Strains the computation capacity of smartphones
- **Strains the batteries** of smartphones
- Bog down the last hop in data transmission

Traffic redundancy elimination (TRE)
Background: Traffic redundancy elimination (TRE)

Data transmission

Sender side

$C_A, C_B, C_C, C_D, C_X, C_Y, C_Z$: data chunks
A, B, C, D, X, Y, Z: hash values of chunks

Encoding

Compressed data

Decoding

Recovered data

Cache

Data transmission

11/28/2018
Background: Existing TRE methods

Existing TRE methods:

- **Receiver-based TRE**
  + Receiver has cache
    - Large amount of up-link synchronization overhead
    - Low cache hit rate
  *i.e.* *Asymmetric caching (AC)*, *MobiCom 2012*

- **In-network TRE**
  + caching is deployed in network (routers, switches)
  - No TRE at the last hop
  *i.e.* *WAN optimization*, *Riverbed Networks*

- **Sender-based TRE**
  - Both sender and receiver have caches
  - Inefficient cache usage
  - huge workload at the server side
  *i.e.* *EndRE, NSDI 2010*

Shortcomings:

- **Existing methods conduct TRE against any application**, however, due to the limitation of memory resource at smartphones, such TRE will result in the **low cache hit ratio, and also less power conservation at smartphone**.
Contributions

To solve these challenges, we propose a novel TRE system, called TailoredRE, that incorporates three components:

- To reduce the computing and up-link bandwidth overhead on smartphones, the proposed TRE is cloud clone assisted.

- To enhance the cache hit rate, so that more power consumption needed for data transmission can be reduced, the proposed method conduct TRE by tailoring to individual user’s activities.

- To conserve the cache cost at the cloud, the clones in the cloud are grouped and their caches are shared.
System design

• Cloud clone assisted TRE

• Personalized and application TRE

• Cache sharing among clones
Could clone assisted TRE: overview

- **Clone**: Each smartphone has a *clone in the cloud*, which is a program running in a virtual machine (VM) to conduct TRE for its belonged smartphone.
- **Data flow path**: server <-> clone (in cloud) <-> network <-> smartphone TRE client

![Diagram showing clone assisted TRE process]

Parses Traffic, detects and eliminates Redundant Traffic
Cloud clone assisted TRE: Clone side

**Chunking**: find the local maximum byte value in the to-be-sent data as chunk boundary.

**Encoding**: use SHA1 to compute the hash for each chunk; if a chunk is found in the cache, then encode chunk with its corresponding hash in the data, otherwise, put the chunk and its hash value in the cache.

Data (application data) received by the clone

Chunked data

Redundancy-eliminated data

Cache at the clone

\[ C_A, C_B, C_C, C_D, C_X, C_Y, C_Z: \text{data chunks} \]

\[ A, B, C, D, X, Y, Z: 20\text{-byte hashes of chunks} \]
Could clone assisted TRE: Smartphone side

- **Search chunks**: if chunk is found in the cache, get the chunk; otherwise, insert the chunk and its hash in the cache.
- **Decoding**: replace the hashes with the found chunks.

The cache at the smartphone side

Data transmission from the clone

Decoding

Recovered data

Applications: YouTube, Facebook, ...

Redundancy-eliminated data

Search chunks
Personalized and application-adaptive TRE

- **Personalized TRE**: The clone performs TRE only on applications that have historically presented higher cache hit rate and larger accumulative transmission volume, which are related to user’s smartphone usage behaviors.

The clone performs TRE against only App1, while neglecting App 2 and App 3.

<table>
<thead>
<tr>
<th>Application</th>
<th>$r_i$ (cache hit ratio)</th>
<th>$a_i$ (relative volume)</th>
<th>Benefit factor $r_i \cdot a_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>App 1</td>
<td>0.9</td>
<td>0.4</td>
<td>0.36</td>
</tr>
<tr>
<td>App X</td>
<td>0.3</td>
<td>0.2</td>
<td>0.06</td>
</tr>
<tr>
<td>App Y</td>
<td>0.5</td>
<td>0.25</td>
<td>0.125</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>APP 2</td>
<td>0.2</td>
<td>0.001</td>
<td>0.002</td>
</tr>
<tr>
<td>APP 3</td>
<td>0.3</td>
<td>0.001</td>
<td>0.003</td>
</tr>
</tbody>
</table>

**Benefit factor** $= r \cdot a$

$r_i$: the average cache hit ratio of application $i$ for the user;

$a_i$: relative data volume of application $i$ for the user;

$r_i, a_i$ are the statistic information of application $i$ computed for each clone in the cloud;
Cache sharing among clones

• **grouping**: The clones on the same VM in the cloud are grouped according to the similarity of user’s application interests by use of locality sensitive hash (LSH).

• **Cache sharing**: The grouped clones share the caches with 2-layer cache sharing strategy.
  - **Logical cache**: each clone has its own logical cache, which is logically independent with other clones. It is used for the synchronization between the individual clone cache and the cache at its corresponding smartphone.
  - **Physical cache**: the grouped clones share a common physical cache, where the chunks are virtually stored. Only one copy of a chunk is stored in the physical cache, and only if all a chunk is removed from all the logical caches, the chunk will be removed from the physical cache.
Performance evaluation

• Data trace collection and analysis

• Trace-driven simulation

• Prototype-based simulation
Data trace collection and analysis

<table>
<thead>
<tr>
<th>Number</th>
<th>Traces</th>
<th>Size (GB)</th>
<th>Redundancy hit ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>WebBrowser1</td>
<td>0.85</td>
<td>90.18%</td>
</tr>
<tr>
<td>2</td>
<td>Techcrunch1</td>
<td>1.11</td>
<td>76.10%</td>
</tr>
<tr>
<td>3</td>
<td>Techcrunch2</td>
<td>1.09</td>
<td>74.09%</td>
</tr>
<tr>
<td>4</td>
<td>Bloomberg1</td>
<td>1.28</td>
<td>69.47%</td>
</tr>
<tr>
<td>5</td>
<td>NYTimes1</td>
<td>1.40</td>
<td>52.60%</td>
</tr>
<tr>
<td>6</td>
<td>Bloomberg2</td>
<td>0.90</td>
<td>40.67%</td>
</tr>
<tr>
<td>7</td>
<td>Quora1</td>
<td>1.36</td>
<td>26.73%</td>
</tr>
<tr>
<td>8</td>
<td>Quora2</td>
<td>0.81</td>
<td>22.26%</td>
</tr>
<tr>
<td>9</td>
<td>Quora3</td>
<td>1.12</td>
<td>20.41%</td>
</tr>
<tr>
<td>10</td>
<td>Twitter1</td>
<td>1.03</td>
<td>20.10%</td>
</tr>
<tr>
<td>11</td>
<td>CNN1</td>
<td>0.98</td>
<td>20.01%</td>
</tr>
<tr>
<td>12</td>
<td>Instagram1</td>
<td>1.40</td>
<td>19.57%</td>
</tr>
<tr>
<td>13</td>
<td>Spotify1</td>
<td>1.01</td>
<td>18.17%</td>
</tr>
<tr>
<td>14</td>
<td>Twitter2</td>
<td>1.13</td>
<td>17.91%</td>
</tr>
<tr>
<td>15</td>
<td>Instagram2</td>
<td>0.84</td>
<td>17.68%</td>
</tr>
<tr>
<td>16</td>
<td>YouTube1</td>
<td>1.58</td>
<td>17.58%</td>
</tr>
<tr>
<td>17</td>
<td>YouTube2</td>
<td>1.02</td>
<td>16.31%</td>
</tr>
<tr>
<td>18</td>
<td>Facebook1</td>
<td>0.94</td>
<td>9.80%</td>
</tr>
</tbody>
</table>

**Traces Collection:** use Wireshark to sniff traffic from servers to the smartphone.

**Redundancy analysis:** The redundancy ratio was analyzed.

Wireshark based traffic sniffing on windows 10
Trace-driven simulation: cache hit rate

**Cache hit ratio:** \( \frac{V_{hit}}{V_{total}} \)

- \( V_{total} \): the total number of bytes of chunks received by the client;
- \( V_{hit} \): the total number of bytes of chunks that hit the cache at the client.

**Compare methods:**
- **EndRE**: a sender-based TRE method that conducts TRE against any application;
- **AC**: a receiver-based TRE method where the caches at the sender and receiver are asymmetric.

- **Cache hit rate**: The proposed method has superior cache hit ratio compared with EndRE and AC, because it always select the applications that have higher rate and most frequently used by users.
- **Stability**: The proposed method has more stable hit ratio over time.
Trace-driven simulation: cache hit rate

- **Cache hit rate**: The cache hit ratio increases with the increase of the cache size at the receiver side.
Trace-driven simulation: efficiency of cache sharing

- **Cache saving**: Profiling grouping (the proposed cache sharing method) conserve half of the cache resource, which is superior efficient compared with random grouping.

Cache size: 2MB
Number of clones (clients): 1-1000
Similarity: 0.4
Max cluster size:
Prototype based simulation: power efficiency

User application preference

<table>
<thead>
<tr>
<th>Applications</th>
<th>Time in one week (h)</th>
</tr>
</thead>
<tbody>
<tr>
<td>YouTube</td>
<td>0.5-2</td>
</tr>
<tr>
<td>Facebook</td>
<td>0</td>
</tr>
<tr>
<td>CNN</td>
<td>2.0-4.0</td>
</tr>
<tr>
<td>Quora</td>
<td>0.5-2.0</td>
</tr>
<tr>
<td>Techcrunch</td>
<td>2.0-4.0</td>
</tr>
<tr>
<td>NYT Times</td>
<td>0.5-2.0</td>
</tr>
<tr>
<td>Twitter</td>
<td>2.0-4.0</td>
</tr>
<tr>
<td>Instagram</td>
<td>2.0-4.0</td>
</tr>
<tr>
<td>Spotify</td>
<td>2.0-4.0</td>
</tr>
<tr>
<td>Bloomberg</td>
<td>0.5-2.0</td>
</tr>
<tr>
<td>Browser</td>
<td>0.5</td>
</tr>
</tbody>
</table>

A survey about user application preference for 12 users
Python-based clone prototype
Android-based smartphone TRE client

- **Power consumption:** the proposed method result in the lowest power consumption across the 12 users, because it has the highest cache hit ratio and needs the least data transmission.
Conclusions

A novel TRE system, TailoredRE, is proposed for power efficiency in smartphones, which has the following properties:

• **Cloud clone assisted TRE**: the TRE computing is performed in the cloud;

• **Personalized and application adaptive TRE**: TRE is performed against selected applications that a user more frequently access and have high redundancy hit ratios for each individual user;

• **Cache sharing among clones**: TailoredRE groups the clones in one VM based on user interests, and let the clustered clones share contents with each other.
Thank you!
Questions & Comments?

Shenghua He
shenghuahe@wustl.edu