Applying Practical Parallel Grammar Compression to Large-scale Data

Masaki Matsushita and Yasushi Inoguchi
Japan Advanced Institute of Science and Technology
Introduction

- Grammar compression algorithm
  - Generates CFG deriving the source text
- Re-pair
  - Representative grammar compression algorithm
  - Achieves high compression rate for text, graph and tree
  - Slower than general compression algorithm in practice
    → We addressed this issue with parallel processing.
- Parallel Re-pair: Parallel variant of Re-pair
  - Basic implementation was shown in DCC 2021
  - In this session, we propose a practical implementation
### Sample Application of Re-Pair

- The most frequent pair is replaced by a new variable
- Output a dictionary and a compressed sequence

<table>
<thead>
<tr>
<th>step</th>
<th>sequence</th>
<th>dictionary</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>abracadabra</td>
<td>$X_1 \rightarrow ab$</td>
</tr>
<tr>
<td>1</td>
<td>$X_1\text{racad}X_1\text{ra}$</td>
<td>$X_2 \rightarrow ra$</td>
</tr>
<tr>
<td>2</td>
<td>$X_1X_2\text{cad}X_1X_2$</td>
<td>$X_3 \rightarrow X_1X_2$</td>
</tr>
<tr>
<td>3</td>
<td>$X_3\text{cad}X_3$</td>
<td></td>
</tr>
</tbody>
</table>
Parallel Re-Pair: a parallel variant of Re-Pair

1. Breaks the input text into chunks
2. Feed chunks into Re-Pair workers
3. Workers share the dictionary synchronized with lock
4. Workers share the dictionary synchronized with lock

- Data to Compress
- Block 1 → CPU 1 → add new pair → Shared Dictionary
- Block 2 → CPU 2 → ...
- Block t → CPU t → compressed strings

$t$ is number of processors

compressed file
Experiments

- Implemented Parallel Re-pair with Intel Threading Building Blocks
- On quad 18-core Intel Xeon G-6240M 2.6GHz with 12TB RAM
- Compared with pbzip2 (parallel version of bzip2)

<table>
<thead>
<tr>
<th>Texts</th>
<th>Size (MB)</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>wikimedia</td>
<td>20,442</td>
<td>wikimedia dump</td>
</tr>
<tr>
<td>genome</td>
<td>59,765</td>
<td>genome variants</td>
</tr>
<tr>
<td>repository</td>
<td>19,466</td>
<td>repository metadata</td>
</tr>
</tbody>
</table>

Table 1: Texts used in our experiments
Experimental Results: Compression Time

- 7.9 to 10.4 times faster than sequential one with 32 CPUs

Figure 1: Compression Time for each text

(a) wikimedia  (b) genome  (c) repository
Experimental Results: Compression Ratio

- Compression ratio of Parallel Re-pair slightly worsened as number of processors increased
- Parallel Re-pair doesn’t consider block boundaries

<table>
<thead>
<tr>
<th>Texts</th>
<th>pbzip2</th>
<th>ours (t=1)</th>
<th>ours (t=2)</th>
<th>ours (t=4)</th>
<th>ours (t=8)</th>
<th>ours (t=16)</th>
<th>ours (t=32)</th>
</tr>
</thead>
<tbody>
<tr>
<td>wikipediam</td>
<td>8.07</td>
<td>6.43</td>
<td>6.66</td>
<td>6.90</td>
<td>7.18</td>
<td>7.49</td>
<td>7.84</td>
</tr>
<tr>
<td>genome</td>
<td>0.71</td>
<td>0.46</td>
<td>0.48</td>
<td>0.49</td>
<td>0.50</td>
<td>0.51</td>
<td>0.53</td>
</tr>
<tr>
<td>repository</td>
<td>8.86</td>
<td>11.05</td>
<td>11.19</td>
<td>11.29</td>
<td>11.51</td>
<td>11.70</td>
<td>11.91</td>
</tr>
</tbody>
</table>

Table 2: Compression Ratio (t is number of processors)
Experimental Results: Memory Usage

- Measured resident set size with “ps” command on all threads
- Almost constant as number of processors increases

<table>
<thead>
<tr>
<th>Texts</th>
<th>pbzip2 (t=1)</th>
<th>pzbip2 (t=2)</th>
<th>pzbip2 (t=8)</th>
<th>pzbip2 (t=32)</th>
<th>ours (t=1)</th>
<th>ours (t=2)</th>
<th>ours (t=8)</th>
<th>ours (t=32)</th>
</tr>
</thead>
<tbody>
<tr>
<td>wikimedia</td>
<td>11.4</td>
<td>23.0</td>
<td>76.2</td>
<td>290.9</td>
<td>571,594.8</td>
<td>571,586.0</td>
<td>571,607.2</td>
<td>571,674.2</td>
</tr>
<tr>
<td>genome</td>
<td>12.1</td>
<td>24.0</td>
<td>80.5</td>
<td>311.9</td>
<td>1,684,568.0</td>
<td>1,684,566.7</td>
<td>1,684,571.5</td>
<td>1,684,631.1</td>
</tr>
<tr>
<td>repository</td>
<td>11.2</td>
<td>23.2</td>
<td>78.2</td>
<td>291.9</td>
<td>544,904.2</td>
<td>544,897.6</td>
<td>544,914.3</td>
<td>544,964.6</td>
</tr>
</tbody>
</table>

Table 3: Memory Usage (t is number of processors)
Conclusion

● We proposed a parallel variant of Re-Pair
● Our experimental results shows
  ○ It is 7.9 to 10.4 times faster than sequential one with 32 CPUs
  ○ Compression ratios are slightly worsened as number of processors increased
● Future work
  ○ Improve compression ratio by considering block boundaries