Fully Parallelized Lossless LZW decompression for CUDA-enabled GPUs

Koji Nakano, Shunji Funasaka, Yasuaki Ito
Department of Information Engineering, Hiroshima University
Kagamiyama 1-4-1, Higashi-Hiroshima, 739-8527, JAPAN

Abstract
We propose a parallel algorithm for LZW decompression and implement it in GeForce GTX 980. It achieves up to 40 times speedup over the sequential implementation using Intel Core i7-4790 (3.66GHz) processor.

LZW data compression
- Developed by Lempel, Ziv, and Welch
- Lossless: original data can be perfectly recovered
- Dictionary-base: dictionary is created sequentially
- Widely used: used in Unix "compress" command, and GIF/TIFF image formats.
- Original input: a string of 8-bit integers such as pixel values/characters
- Compressed output: a sequence of 9 to 12-bit codes

LZW compression algorithm
Initialization: Create dictionary with initial codes
Repeat the following steps until all input are read
1. Find the longest substring $W$ from the beginning of the input that is in the dictionary
2. Output the code corresponding to $W$ and remove $W$ from the input.
3. Read the next character $x$ and add substring $Wx$ to the dictionary

LZW decompression algorithm
Similar to LZW compression
The dictionary is created from compressed codes

Destination
Koji Nakano: nakano@hiroshima-u.ac.jp

Experimental Results
Evaluated the time for LZW-compressed TIFF images of 4096x3072 pixels (%: compression ratio)

<table>
<thead>
<tr>
<th>Scenario</th>
<th>A</th>
<th>A-2</th>
<th>B-2</th>
<th>B-3</th>
<th>B-1</th>
<th>C-2</th>
<th>C-3</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Crafts</td>
<td>6.38</td>
<td>3.53</td>
<td>9.91</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Flowers</td>
<td>6.43</td>
<td>3.52</td>
<td>5.56</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Graph</td>
<td>6.50</td>
<td>3.50</td>
<td>10.05</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Conclusion
Our GPU implementation for the LZW decompression is up to 50.00/1.25=40 times faster than the CPU implementation. Also Scenario C is up to 10.05/2.08=4.8 times faster than Scenario A. These scenarios can be applied to big data applications such as training phase of deep learning using the GPU, because the GPU stores the original uncompressed image at the end of scenarios.