Compression Schemes with Data Reordering for Ordered Data

Chun-Hee Lee, Department of Computer Science, KAIST (Korea Advanced Institute of Science and Technology), Daejeon, Korea
Chin-Wan Chung, Department of Computer Science, KAIST (Korea Advanced Institute of Science and Technology), Daejeon, Korea

ABSTRACT

Although there have been many compression schemes for reducing data effectively, most schemes do not consider the reordering of data. In the case of unordered data, if the users change the data order in a given data set, the compression ratio may be improved compared to the original compression before reordering data. However, in the case of ordered data, the users need a mapping table that maps the original position to the changed position in order to recover the original order. Therefore, reordering ordered data may be disadvantageous in terms of space. In this paper, the authors consider two compression schemes, run-length encoding and bucketing scheme as bases for showing the impact of data reordering in compression schemes. Also, the authors propose various optimization techniques related to data reordering. Finally, the authors show that the compression schemes with data reordering are better than the original compression schemes in terms of the compression ratio.

Keywords: Bucketing Scheme, Compression Scheme, Data Reordering, Ordered Data, Run-Length Encoding

INTRODUCTION

Currently, a large volume of data in various environments is generated. Such a large volume of data consumes valuable resources such as space, network bandwidth, and CPU. In order to save the resources, data compression schemes have been developed and applied in many applications.

However, they do not consider the effect of data reordering. If we reorganize data, the compression ratio for the reorganized data may be improved compared to that for the original data. Some papers deal with data reordering problems in very limited environments (Apaydin, Tosun & Ferhatosmanoglu, 2008; Blandford & Blelloch, 2002; Johnson, Krishnan, & Chhugani, 2004; Pinar, Tao & Ferhatosmanoglu, 2005). The work of Apaydin et al. (2008), Blandford and Blelloch (2002), Johnson et al. (2004), and Pinar et al (2005) assumes that the order of data does not have to be preserved, that is, the input data is unordered data. However, in general, the order of data should be preserved and has the important information. For example, time series data should be ordered by the time. If we change the order of the time series data, it will lose much information. Therefore, the approaches in Apaydin et al. (2008), Blandford and Blelloch (2002), Johnson et al. (2004), and

DOI: 10.4018/jdm.2014010101
Pinar et al. (2005) cannot be applied to the ordered data such as time series data (Chen, Dong, Han, Wah, & Wan, 2002; Korn, Jagadish, & Faloutsos, 1997; Reeves, Liu, Nath, & Zhao, 2009; Elmeleegy, Elmagarmid, Cecchet, Aref, & Zwaenepoel, 2009).

Consider the run-length encoding with the ordered data \( D = \{1, 1, 1, 3, 3, 1, 1, 4, 4, 4\} \). The run-length encoding is one of the widely used lossless compression schemes. It replaces repeated values with \( <\text{value, count}> \), where count is the number of repeated values. We can represent \( D \) by the run-length encoding as follows. Note that \( \text{RLE}(D) \) is the compressed data for \( D \) using the run-length encoding. See Table 1 for the detailed notational convention.

\[
\text{RLE}(D) = \{<1,3>,<3,2>,<1,2>,4,3>\},
\]

where in the pair \( <a,b> \), \( a \) is value and \( b \) is count.

We can reduce the number of elements in \( \text{RLE}(D) \) by reordering elements in \( D \). Consider \( D' = \{1, 1, 1, 1, 3, 3, 4, 4, 4\} \) which is the data after reordering elements in \( D \). Then, \( \text{RLE}(D') = \{<1,5>,<3,2>,<4,3>\} \). Since \( |\text{RLE}(D')| = 3 \) is less than \( |\text{RLE}(D)| = 4 \), we can improve the compression ratio by reorganizing data if the data is unordered data. However, if the data is ordered data, we should keep the following mapping table (presented in Box 1) to reconstruct the original data from \( \text{RLE}(D') \).

The space benefit by data reordering may be less than the space overhead for storing the mapping table. That is, the compression ratio by the compression scheme with data reordering may be worse than that by the original compression scheme without data reordering. In this case, data reordering is useless. Therefore, we should carefully consider how to store the mapping table effectively in order to apply data reordering techniques for ordered data.

In this paper, we first investigate general principles to improve compression schemes by data reordering. We do not keep the total mapping table since the size of the mapping table is too big. Instead, we keep the movement information for the portion of a whole data set. The movement information is represented by \( \ll \text{start, end, newStart} \gg \). It means that the

<table>
<thead>
<tr>
<th>Box 1.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original position: 1 2 3 4 5 6 7 8 9 10</td>
</tr>
<tr>
<td>Changed position: 1 2 3 4 5 6 7 8 9 10</td>
</tr>
</tbody>
</table>

---

Table 1. Notational convention

<table>
<thead>
<tr>
<th>Notation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{RLE}(D) )</td>
<td>Compressed data for ( D ) using the run-length encoding</td>
</tr>
<tr>
<td>( \text{BUCKET}(D, \varepsilon) )</td>
<td>Compressed data for ( D ) using the bucketing scheme with an error bound ( \varepsilon )</td>
</tr>
<tr>
<td>( d_i )</td>
<td>the ( i )-th element in data set ( D )</td>
</tr>
<tr>
<td>( d_{ij} )</td>
<td>( {d_i, d_{i+1}, \ldots, d_j} ), where ( i \leq j )</td>
</tr>
<tr>
<td>( &lt;X&gt; )</td>
<td>token ( X ) in the run-length encoding or the bucketing scheme</td>
</tr>
<tr>
<td>( \ll X \gg )</td>
<td>movement information ( X )</td>
</tr>
</tbody>
</table>

Copyright © 2014, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.
Related Content

Evaluating the Dynamic Behavior of Database Applications
Zhen He and Jérôme Darmont (2005). Journal of Database Management (pp. 21-45).
www.irma-international.org/article/evaluating-dynamic-behavior-database-applications/3330/

Managing Temporal Data
www.irma-international.org/chapter/managing-temporal-data/20685/

On Negative Information in Deductive Databases
www.irma-international.org/article/negative-information-deductive-databases/51076/

Social Network Structures in Open Source Software Development Teams
www.irma-international.org/article/social-network-structures-open-source/3369/

Information Search Patterns in E-Commerce Product Comparison Services
www.irma-international.org/chapter/information-search-patterns-commerce-product/63665/