Template-based Delta Compression of Large Scale Web Pages

Kai Lei*, Guangyu Sun, Lian'en Huang
Shenzhen Key Lab for Cloud Computing Technology & Applications (SPCCTA),
Shenzhen Graduate School, Peking University, Shenzhen 518055, P.R. China
*leik@pkusz.edu.cn, sun120409@gmail.com, hle@net.pku.edu.cn

Abstract—**Delta compression techniques are commonly used in the context of version control systems and the World Wide Web. They are used to compactly encode the differences between two files or strings in order to reduce communication or storage costs. In this paper, we study the use of delta compression in compressing massive web pages according to the similarity of their templates. We propose a framework for template-based delta compression which uses template-based clustering techniques to find the web pages that have similar templates and then encode their differences with delta compression techniques to reduce the storage cost. We also propose a filter-based optimization of Diff algorithm to improve the efficiency of the delta compression approach. To demonstrate the efficiency of our approach, we present experimental results on massive web pages. Our experiments show that template-based delta compression achieves significant improvements in compression ratio as compared to individually compressing each web page.**

Keywords- LCS, Diff, Delta compression, template

I. INTRODUCTION

Delta compression is a main field in data compression research which is concerned with compressing *target data set* in terms of *reference data set* by computing a *delta* which is viewed as an encoding of difference between them. It plays a more and more important role in information retrieval systems and other web applications because they have to crawl and store more and more data with the rapid growth of information on the World Wide Web.

Delta compression is firstly used in revision control systems[1]. By storing deltas of different versions, instead of the actual data, these systems are able to reduce storage cost significantly. However, most delta compression methods need to be performed with respect to a previous version of the same file, or some others that are easy to be identified as the reference data set, or they will be quite inefficient. As a result they don't perform well in large scale web pages where it is difficult to efficiently identify target and reference data set.

A feasible way to compressing large scale web pages is to use clustering algorithms. By clustering methods, the large data set is first divided into subsets of similar web pages which have large compressible content, and then delta compression is performed to compress the web pages of each subset respectively. And how to cluster similar web pages which have large compressible content in a fast and accurate way is important to delta compression. Gibson et al.[2] pointed that templates represent between 40% and 50% of data on the Web and this volume has been growing at a rate of approximately 6% per year, and the template proportion of portals is higher than that of other websites. That indicates using web pages' structural information to find similar templates might be an efficient way to indentify similar pages that have large compressible content.

In this paper, we propose a delta compression framework based on template for efficiently compressing large scale web pages. In the framework, we propose a random sampling clustering algorithm which can cluster the pages that have the same or similar templates together efficiently. We propose an optimization on Myers' diff algorithm[3] based on sliding window to improve its efficiency and use the optimized diff algorithm to compute the difference between web pages. We have evaluated our compression approach on the 19 million web pages extracted from a repository of 2.4 billion web pages collected by Web Infomall[4]. Compared to the results of Gzip[5] on the same data set, our template-based approach performed much better in compression ratio.

The rest of the paper is organized as follows: the related works of delta compression is introduced in Section 2. Our template-based delta compression approach is described in Section 3. The experiments results are reported in Section 4 and finally we conclude our work in Section 5.

II. RELATED WORK

The main delta compression algorithms in use today are diff and vdelta[6]. Using diff to find difference between two files and then applying Gzip[5] to compress the difference is a simple and widely used way to perform delta compression, but it does not provide good compression on files that are only slightly similar. Vdelta is a relatively new technique that integrates both data compression and data differencing.

The problem statement of determining the differences between two sequences of symbols is to find the minimum "script" of symbol deletions and insertions that transform one sequence into the other. Among numerous algorithms for the problem, Myers' difference algorithm[3] performs best. Myers solved the problem by transforming it into an equivalent problem of finding a shortest path in an edit graph and developing a greedy algorithm to solve the single-source shortest path problem.

The clustering techniques are studied extensively. K-means and its variants have a time complexity that is linear in the number of documents, but it suffers from the serious drawback that its performance heavily depends on the initial starting conditions. Unlike the previous method, hierarchical clustering methods do not require to fix the number of clusters priori as a parameter. It starts with all instances each in its own cluster, and then repeatedly merges the two clusters that are most similar in each iteration. The single linkage approach[7] is always merging those two clusters for which the distance between two of the documents from the two clusters is minimal over all inter-cluster distances. It is often portrayed as the better quality clustering approach, but is limited because of its quadratic time complexity.

There are several algorithms[8-10] that can identify structural similarity between templates in the template-based clustering process. Several authors proposed algorithms for detecting structural similarity based on tree-editing-distance[11]. Flesca et al.[12] introduced the Fourier transform technique as a mechanism to compute similarity between documents. Buttler[13] reduces paths in a document to hash values, which can be compared to those of other documents using set union and set intersection operators.

III. Template-based Delta Compression

The basic idea of our framework is to divide the set of massive pages into subsets of template-similar pages so that there is large space for compression within each subset. Clustering algorithm is first applied in the division of massive pages. The pages that have same or similar templates will be clustered together as they are considered to have large common content. Then we perform a delta computation for the pages of each cluster using Myers' diff algorithm and then save the delta and reference data set.

A. Clustering based on template-similarity

The first step of our framework is to divide massive web pages into subsets of pages that have similar templates, in order to constrain the computation of delta occurring only between pages that have large common content in which Myers' $O(ND)$ difference algorithm[3] works efficiently. The algorithm consists of two phases: computing a sketch for each pages and clustering the pages based on their sketches.

```
Procedure Cluster(D, sim)
Begin
1. compute a sketch s foreach document d in D, D is the set of document d
   S ← Compute_sketch(D), S is the set of sketch s
2. set T ← {}, T is the set of sketch of initial template t
3. set C ← {}, C is the set of cluster $C_t$, $C_t$ is the set of document d
4. choose k templates randomly in D, sim is threshold of the similarity
   between templates
   T ← Select_Template(S,sim,k)
5.  for each sketch s in S do
6.      for each template t in T do
7.          if similarity of s and t is large than sim
8.              if $C_t$ exist in C
9.                  $C_t \leftarrow C_t \cup \{d\}$
10.             else
11.                 $C_t \leftarrow \{d,t\}$ ,$C \leftarrow C \cup \{C_t\}$
12.                 $S \leftarrow S - \{d\}, break$
13.         end if
14.     end for
15. end for
16. if D has no change, End
17. else goto 4
End

Procedure Select_Template(S, sim, k)
1. choose d from D randomly
2. set T ← {}, T is the set of template t
3. if similarity of s and t is small than sim, $\forall t \in T$
4.     $T \leftarrow T \cup \{d\}$
5. if sizeof(T) < k, goto 1
6. retuen T
```

Fig. 1. The procedure of clustering algorithm.

In the first phase, we use Broder's min-wise independent hashing method[14] to compute a sketch of M fingerprints for each page. For the goal of identifying the pages that have similar templates, we take html tags and the strings between html tags as tokens rather than lines, words, or letters which is different from the tokens in Broder's algorithm[15]. The shingle in our algorithm is a contiguous sequence composed by tokens of html tags and tokens of strings between html tags. No matter how long a string between html tags is, it is seen as a token, which reduces the influence of the content blocks. So no matter the content blocks of the pages is the same or not, the same templates can be identified.

In the second phase, we cluster the pages based on the sketches. Our clustering algorithm is a random sampling algorithm whose time complexity is $O(NI)$ and space complexity is $O(N)$ where N is number of all pages and I is number of iterations. Considering that there may be a mass of pages which have no similar template with other pages, we randomly select k initial templates in each iteration. Time complexity of computing sketch for each document is $O(N)$ and time complexity of Select_Templates is $O(k^2)$ where k is number of templates selected in each iteration. And time complexity of clustering documents is $O(NMI)$ where M is the actual number of templates of all documents. So time complexity of our clustering algorithm is O(NI). Number of iteration I will be M/k approximately if there are M different templates, because the clustering algorithm can identify all the pages which have the same templates with the given k templates. When all M templates have been identified, the algorithm completes.

As illustrated in Figure 1, there are several parameters that may affect the result of our clustering algorithm. Threshold *sim* is an important parameter to measure the similarity between templates. The compression efficiency of the whole approach is related to the threshold *sim*. If *sim* is too high, there will be many pages that have no similar pages to compute delta data with, which lead to bad compression ratio. If *sim* is too low, the similarity of the pages within the cluster will be reduced, which leads to bad efficiency. Our experiments show that when similarity threshold is 50%, the whole compression efficiency is optimal.

B. Delta Computation

The second step of our framework is to perform a delta computation for the pages within each cluster. We use Myers' difference algorithm to compute the delta between target data set and reference data set where target data set is one randomly selected page in each cluster and reference data set is the others. Time of Myers' algorithm is $O(ND)$ where N is sum of the lengths of them and D is size of the minimum edit script for them. When D is not very small, Myers' algorithm consumes huge time to compute the delta. So improvement in efficiency of the difference algorithm is important to our approach. We propose a filtering optimization method to reduce the differences between two pages before computing delta which can further reduce the time of delta computation.

We firstly segment the web pages into sets of tokens before delta computation. Although there are many possible segmentation methods, such as letter-based, word-based, line-based and phase-based, we choose the segmentation method based on html tags. The reason for our choice it that our clustering algorithm clusters the pages that have similar template together; as a result, the pages within a cluster have many same html tags.

Although the difference algorithm is performed within each cluster, there are two reasons for which improvement of its efficiency is necessary: (1) there would be some false-positives in our clustering algorithm which have large difference from others within the cluster and consume huge time of delta computation, and (2) as mentioned in subsection 3.1, the similarity between the pages within each cluster can't be very high, so the delta computation consumes huge time which counts the most part of the whole time of our approach. For example if the similarity of two pages is 20% of sum of their lengths, the $O(ND)$ time for computing delta is then approaching a $O(N^2)$ time. So a filtering optimization method is needed to further reduce the difference between two pages before delta computation.

Our filtering optimization method can be divided into three steps: given two documents A and B to compare, (1) we use the filtering algorithm based on sliding window to produce the filtered document A' and B', and the corresponding edit script S_A and S_B. Document A can be later recovered from document A' and S_A, so does document B. (2) compute *SES'* (shortest edit script) between documents A' and B'. (3) use S_A , S_B and *SES'* to compute *SES* between Document A and B. It is important to note that document A' and B' are more similar, so the time of computing *SES'* between them is much shorter. The task of step (3) is the procedure of merging S_A, S_B and *SES'* together which can be completed in linear time.

Huang's work[16] on managing duplicate data gives us much inspiration in designing the filtering algorithm. Our filtering algorithm is as follows: given two documents A and B to compare, we compute a filtered document for A and B respectively consisting of sufficiently long fragments that exist in both of them in the following way: (1) a sliding window of size S is used to compute the fingerprint for each overlapping S characters of each document using Rabin's fingerprinting functions[17], and (2) each fingerprint of A is hashed into a hash table T. (3) for each fingerprint of B, if it finds a match in T with the fingerprints of A, the corresponding sequence in B is reserved. The intersection of such reservations forms the filtered document B'. We can get the filtered document A' through a similar process. The time and space cost of this algorithm are both $O(N)$.

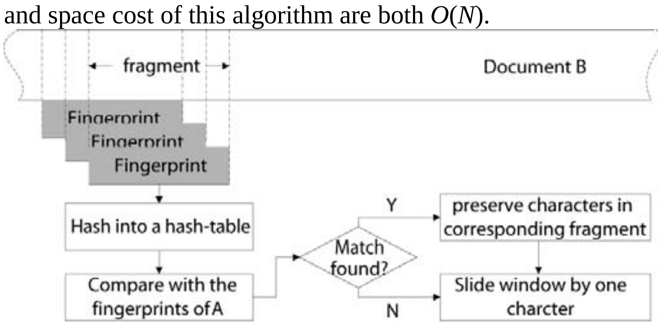
fragment
Document B
Fingerprint
Fingerprint
Fingerprint
Hash into a hash-table
Compare with the fingerprints of A
Match found?
Y
N
preserve characters in corresponding fragment
Slide window by one charcter

Fig. 2. filtering algorithm.

IV. EXPERIMENTS

In this section, we perform an experimental evaluation of our approach. The experiments are organized as follows: firstly, we evaluated the sensitivity of key parameters deciding the efficiency of our approach; secondly, we compare our delta compression approach with Gzip on the data set of 19 million web pages from a repository of 3 billion web pages collected by Web InfoMall[4], which is a Chinese web archive that has been built at Peking University since 2001. The web pages in our experiment are all the raw pages without the process of cleaning noisy because we think the 'noise' is template of web pages which is very important for our compression approach. All the experiments are run on Linux 64-bit system, with Intel(R) Xeon(R) 4core CPU at 2.5GHz, 6M of Cache and 2GB of RAM.

A. Parameter Sensitivity and Efficiency

There are two key parameters that can impact the efficiency of our approach, the threshold of similarity in the clustering stage and the length of sliding window in the delta computation stage. The data set in this subsection is 10000 web pages selected randomly from http://auto.sina.com.cn in Web Infomall. We will analyze the parameter sensitivity to our approach in detail.

Firstly, we evaluate the impact of similarity threshold on the distribution of web pages in the result of clustering step. Setting the similarity threshold to different values can produce subsets of pages that have different template similarity. We use *number of templates* and *coverage rate* to evaluate the impact on the distribution of web pages. *Number of templates* is the number of clusters. *Coverage rate* is defined as the ratio between the number of web pages that have templates and that of all web pages. And web pages that are not similar with any others are considered that they don't have templates and can't be compressed. For the other parameters, we set number of initial template number *k* to 30, size of sketch *m* to 100 and width of shingle *w* to 3.

The result is illustrated as Figure 3, which shows the change of *number of templates* and *coverage rate* when similarity threshold increases from 0 to 1. When the similarity threshold increases from 0 to 0.6, the coverage rate descends smoothly. But when the similarity threshold is higher than 0.6, the coverage rate descends rapidly, which indicates that there are a lot of web pages which have 0.6 similarity with other pages. The change of *number of templates* indicates that the number of clusters increases when the similarity threshold increases, but the size of cluster is decreasing. It's worth to say that when the similarity threshold is too high, the number of clusters descends rapidly. In the case, most of the similarity between web pages is not that high and mass of web pages are considered to have no similar template with any other web pages which reduce the number of templates. The higher similarity threshold is, the smaller the *number of templates* is.

Secondly, we evaluate the impact of similarity threshold on computation *time* and *compression ratio* in delta compression step. We use the result of clustering step as the data set of this experiment and use our delta compression algorithm to compress the web pages within each cluster. We set length of sliding window to 7 which is proved the most efficient value in next experiment.

The result is illustrated as Figure 4, which shows the change of *time* and *compression ratio* when similarity threshold increases from 0.1 to 0.9. The *compression ratio* reaches its peak which is 0.73 when similarity threshold is 0.5. And we adopt this value of similarity threshold in our system. The reason for the lower compression ratio in the range corresponding to similarity threshold between 0.1 and 0.5 is that the similarity between the pages within each cluster is too small and there are little common content between them.

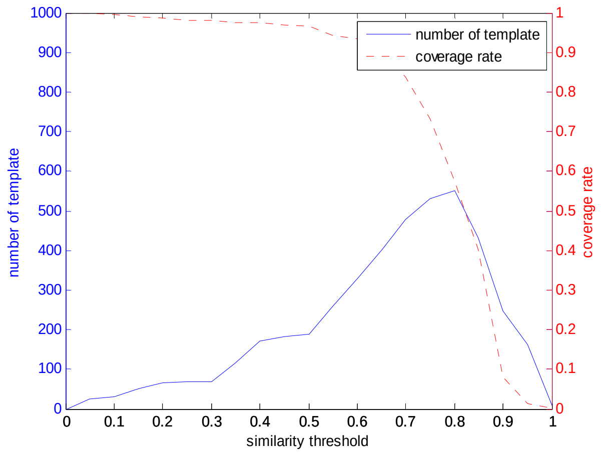

Fig. 3.impact on the distribution of web pages.

The reason for the lower compression ratio in the range corresponding to similarity threshold between 0.5 and 0.9 is that mass of pages are considered to have no template and have no pages to compute delta with. The time doesn't increase linearly with the descending of similarity threshold, which proves that our optimization can further improve the time of Myers' diff algorithm when the documents are not very similar.

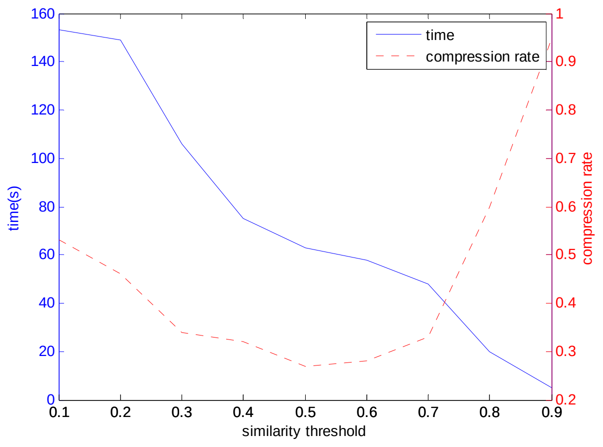

Fig. 4. Impact on the time and compression ratio.

Finally, we evaluate the impact of the length of sliding window on both *time* and *compression ratio* of optimized compression algorithm by changing the length of sliding window from 0 to 14. We first use our template-based clustering algorithm to cluster our data set into subsets of similar web pages and then use optimized Mayer's diff algorithm to compress the web pages in each cluster with different length of sliding window. For the other parameters, we set number of initial template number *k* to 30, the size of sketch m to 100 and the similarity threshold to 50%.

The result is illustrated as Figure 5, which shows the change of time and compression ratio when length of sliding window increases from 0 to 14. The change of *compression ratio* and *time* indicates that small length of sliding window will lead to poor time, but large length of sliding window will lead to poor compression ratio since it filters massive common tag sequences of small length. The time and compression ratio seems to reach a balance when length of sliding window is 4 to 8, which is a proper choice for our approach. When length of sliding window is small or even 0, the effectiveness of the filtering optimization is little and it's very time-consuming, which indicates that the filtering optimization can further improve the time of Mayer's diff algorithm with some decrease on compression ratio.

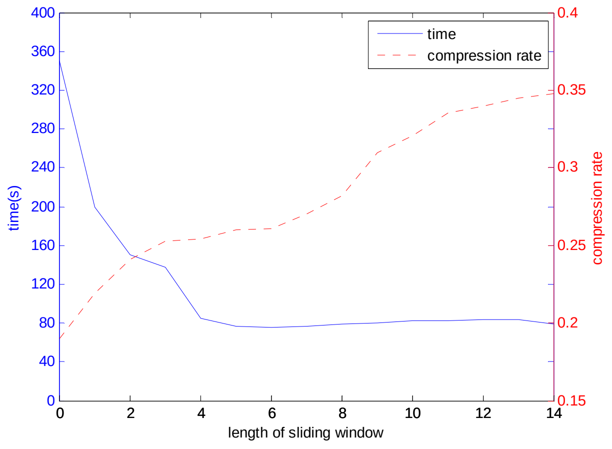

Fig. 5.Impact of the length of sliding window.

C. Comparison Experiment

We compare our delta compression approach named TBC (Template Based Compression) with Gzip in compression ratio on the data set of 19 million web pages breadth-first crawled by Web InfoMall. We also present the result of the combination of Gzip and TBC.

algorithm	Size of data set	time	Compression ratio
uncompressed	665G		100%
TBC	179G	65hours	27%
Gzip	137G	72hours	21%
TBC + Gzip	49G	84hours	8%

Table. 1. Comparison on large scale of web pages.

Table 1 lists the result of comparison of TBC and Gzip in compression ratio. While compression ratio of TBC is a little worse than that of Gzip, time of TBC is better than Gzip. It is important that the combination of Gizp and TBC achieve much better compression ratio than that of both of them, though time of the combination of Gizp and TBC is a little worse than them. The experiment proves that TBC can compress large scale of web pages well and it will achieve great compression effect when it is combined with Gzip.

V. CONCLUSIONS

The approach we proposed identifies and clusters similar web pages according to their similarity of templates and performs delta compression within each cluster. The clustering algorithm in our approach can efficiently divide large scale of web pages into subsets of web pages with similar templates.

The filtering optimization method we proposed further reduces the time-cost of delta computation algorithm used in the web pages compression within each cluster. The approach has been successfully used to compress a data set of 19 million web pages which is crawled from Chinese websites in a month by Web InfoMall using breadth-first crawling algorithm. Experiments show that our approach provides pretty good compression ratio when it is combined with Gzip.

ACKNOWLEDGMENTS

This research is financially supported by NSFC of China (Grant No: 61103027), 973 Projects (No：2011CB302300 and Guangdong Gov Projects (2011A090200063 and 2011B090400396) , and Shenzhen Gov Project (JC201104210107A and ZYA201106080025A）.

REFERENCES

[1] . J. Hunt, K. P. Vo, and W. Tichy. ACM Transactions on Software Engineering and Methodology, 7 (1998).

[2] . D. Gibson, K. Punera, and A. Tomkins. In Proc. 14th WWW (Special interest tracks and posters), pages 830–839 (2005).

[3] . E. W. Myers. Algorithmica, 1(2):251–266 (1986).

[4] . L. Huang, H. Yan, and X. Li. In Proceedings of The World Engineers' Convention, volume A, pages 217–222. China Science and Technology Press, Nov (2004).

[5] . L. P. Deutsch, "RFC 1952: GZIP file format specification version 4.3", May (1996).

[6] . W. Tichy. RCS: A system for version control. Software - Practice and Experience, 15, July (1985).

[7] . Eddy, W.F.; Mockus, A. & Oue, S. (1996). Journal of Computational Statistics and Data Analysis, Vol 23, pp. 29 – 43.

[8] . D. de Castro Reis, P. B. Golgher, A. S. da Silva, and A. H. F. Laender. In Proceedings of the International Conference on the World Wide Web, pages 502–511 (2004).

[9] . G. Valiente. In Proceedings of the International Symposium on String Processing and Information Retrieval, pages 212–219. IEEE Computer Science Press (2001).

[10]. W. Yang. Software – Practice And Experience, 21(7):739–755 (1991).

[12] . S. Flesca, G. Manco, E. Masciari, L. Pontieri, and A. Fifth International Workshop on the Web and Databases (2002).

[13] . Buttler, D. In: IC '04: Proceedings of the International Conference on Internet Computing, CSREA Press (2004) 3–9.

[14] . A. Broder. pages 21–29. IEEE Computer Society (1997).

[15] . A.Broder, A.Z., Glassman, S.C., Manasse, M.S., Zweig, G. Computer Networks 29(8-13) (1997) 1157–1166.

[16] . Lian'en Huang, Lei Wang and Xiaoming Li. Proceeding of the 17th ACM conference on Information and knowledge management, 63-72 (2008).

[17] . A. Broder. Methods in Communications, Security, and Computer Science, 143–152 (1993).