Research on Incremental Updating

Guiqiang Ni[1], Yingzi Yan[1,*], Jinsong Jiang[1], Jianmin Mei[1], Zhilong Chen[2] and Junxian Long[3]
[1]Institute of Command Information System, PLA University of Science and Technology, Nanjing 210007, China
[2]Nanjing Army Command College, Nanjing 210045, China
[3]Chinese People's Liberation Unit 69066, Wulumuqi 830092, China
[*]Corresponding author

***Abstract*—Compression and differential technology make the storage and transmission of files more efficient, and has an important role in the field of storage and synchronization. This paper introduces the development of incremental updating technology and the current major incremental updating algorithm. Dividing incremental updating algorithms into three classes by the searching method of sequence search, dictionary search and suffix tree search. Finally, test the typical incremental update algorithms and analysis their performance.**

Keywords-incremental updating; differencing; delta encoding

I. INTRODUCTION

Incremental update is to store and transfer data with the differences between files, instead of file itself. This difference is often referred to as a patch or incremental package. The target file can be recovered from the source file with this incremental package. The size of the difference is always not greater than the size of the target file, when the source file and the target file related, the difference between the information will be far less than the target file. The greater the similarity of the source file and the target file, the less the difference is, the smaller the incremental package is. The application of incremental updating technology has greatly reduced the cost of file storage and transmission. Much storage space, network bandwidth, time of update can be saved [2]. In addition, the network flow rate and energy consumption can also be reduced, especially suitable for low bandwidth and traffic sensitive network [10]. At present, this technology has been applied to the [1], software upgrade, data synchronization, cloud storage, protein sequence comparison and so on. And it is an essential part of most software configuration management system, which realizes the software version tracking.

The basic process of incremental updating is compare the source file and the target file into an incremental packet. And then incremental packet will be transferred to the target host; the target host can construct the target file with source file by decompression incremental package. For example, when a software needs to be updated, the software authors use both two versions of the software to release an incremental package, and provide download. Users can download and extract the incremental package file to upgrade the new software version, in this, the user's local software and server release software can be synchronized. The structure of incremental packet is the result of difference. So the process of constructing the incremental packet is divided into two steps, difference and encoding. In this paper, we study the difference algorithm. The procedure of difference is to find out the differences between documents, precisely the same information in files. This problem was first proposed as string correction by Levenshtein in 1965.And then, a series of differential algorithm are studied. The main difference of these algorithms is that the data structure, the compare algorithm and the encoding algorithm, so the complexity and compression rate of these algorithms are also different. Different information needs to be compressed in a light and efficient format, which ensures the difference information be transmitted through the network and recognized by different software and hardware platforms. Encoding format generally need to have the following four characteristics: independent of the differential algorithm, data canbe transferred, the output can be compressed and efficiently decoded. Many compression algorithms have their own encoding format to describe data, however, there are very few published ones except GZIP [4] encoding format. Delta encoding is incomplete data compression, and its PUBLISHED ENCODING FORMATS ARE GDIFF [14] format and VCDIFF [10] format.

II. RELATED WORK

Incremental updating problem originated in string correction raised by Levenshtein in 1965 [19]. Then the algorithm has been optimized, in 1976 UNIX system applied the first incremental updating tool diff[15]. This tool is only available to text file, it differences the file by file line, finds out the same units of the file and outputs a readable difference lists(encoding), which is a set of the least lines from the source file required for the synthesis of the target file. The different versions of the text file can be stored efficiently by using the tool. Obviously, the difference with the subsequent development of the algorithm comparison algorithm is not worth mentioning, because of the large and rough differences information it acquired, nor the differential results for compression. Later, the incremental updating algorithm is mostly refined to the byte level and uses compression algorithm similar to ZivLempel to compress the difference results [18], which greatly improves the compression ratio of the algorithm.

Three important indices to evaluate the performance of an incremental updating algorithm are compression ratio, time complexity and space complexity [5]. Compression ratio is the ratio of the size of the savings and the size of the target file. The smaller the incremental package, the higher the compression ratio, the better the compression effect. These three indices constraints each other, a good compression ratio often means greater operating time and space requirements. For the practical application of the algorithm, the goal is to perform synchronization as fast as possible, which means a tradeoff between speed and compression ratio. "Good enough" solutions are usually acceptable, as a better solution requires more time and space [3].

In addition, the special format of the file incremental update algorithm is researched gradually, such as the APK, HTML file incremental update. The general incremental updating algorithm ignores the special file structure, and cannot achieve the best results in special occasions. The research of the special incremental updating algorithm makes the special file structure better.

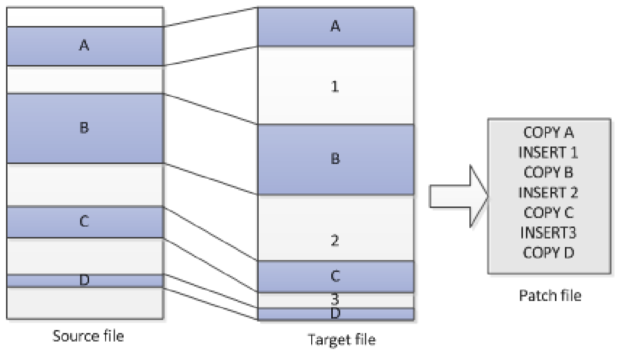

FIGURE I. PROCEDURE OF INCREMENTAL UPDATING ALGORITHM

III. TYPICAL INCREMENTAL UPDATING ALGORITHM

Differencing is the key of incremental updating algorithm. Its results directly determine the patch size and the efficiency of the algorithm. In essence, differencing is to find the repeating segments between the source file and the target file, then find all matches between the two files and write the location and the length of every repeating segment into the patch file with a special encoding method [7]. Because of a certain overhead for expressing a duplicate message and overlap maybe occurred between different repeating segments, algorithm also needs to choose which found repeating segments to use. Repeat segments are selected in case of the efficiency of the algorithm and the size of output incremental package,

The problem of finding repeating segments can be abstracted as the problem of finding common subsequence of two strings [8]. Among the known incremental updating algorithms, three kinds of searching methods are adopted. Which are sub sequence searching, dictionary searching and special data structures searching (such as suffix tree searching). The search efficiency of different search methods is different [9]. Efficient search methods often need more run time and space. According to the practical application, algorithm adopts the suitable search method.

A. Incremental Updating Algorithm Based on Sub Sequence Search

Early difference algorithms use the longest common subsequence (LCS) algorithm [17], Miller and Myers presented that the LCS algorithm’s shortcomings when used to find differences [11]. Edit distance is another measure method used to measure sequence differences. It is defined as the minimum number of edits required for a sequence to change to another sequence, and a shortest path found on the edit graph indicate a differencing result. DDIFF algorithm [18] proposed by Gui-qiang Ni, Zhi-long Chen [18] also uses the concept of edit graph. Compared to LCS, the edit distance can better describe the difference between the sequences, and output smaller difference information. Early search algorithms based on the comparison of characters. Which have high time expense and also unable to identify the byte transposition within a character. Tichy proposed block moving algorithm to solve this problem [6]. Instead of operate only one character at a time, the characters are operated as a block. Tichy defines two kinds of edit operation: block movement and add operation. Both are allowed to combine continuous operation of the same type acting on the same substring to one operation. Block movement is a copy of the substring from the old file, and add operation is to get the substring from the delta code. The idea of the algorithm is to read a new string and compare each position with the old one. For each location, if there is a match, as a copy instruction output the maximum repetition, otherwise output an insert instruction. Each repeat block can be represented by a block movement operation, and each difference block can be represented by an add operation.

The algorithm finds the optimal solution of maximum matching in the new and old string, the time overhead is O(nm) and O(nm2). Because the time to process a long string will be more than the time to directly copy the string, so the algorithm is not feasible in practical application.

B. Incremental Updating Algorithm Based on Dictionary Search

In order to avoid iterating the old version to search substring, dictionary structure is raised. Dictionary is a hash table. It calculates fingerprints of all short constant length prefix. And consider the fingerprints a key value to store the prefix’s position in the old file. When searching for a substring in the new version string, the algorithm calculates the substring’s hash value, and then query the corresponding location in the dictionary of old version string. So that old version string will not be iterated.

Dictionary structure greatly improves the execution speed of the algorithm. The time bound of find a match is O (n). However the bound of space expense still high. In order to solve the transposed string matching and reduce space expense, Trendafilov proposes to use a sliding window method. The method is based on the assumption of spatial location: Even assuming that shift, repetition is relatively close to the position. Therefore, the choice of any length sufficiently large enough windows, these two strings included in the respective repeating are within the sliding window. Fingerprints and subsequent differential is completed only in the window. Old string is a fingerprint window, if the window is the entire string.

A famous incremental updating algorithm based on dictionary structure is VCDIFF ,which is also a common delta encoding format. VCDIFF combines the target file with the source file and the target file is part of the dictionary constructs a delta based on dictionary searching, which eliminate the redundancy within the target file itself efficiently. It constructs the delta with copies from the dictionary, and use three instruction to encode patch file. There are ADD, COPY and RUN instruction. The RUN instruction indicates that one byte is repeated several times, which increase the compression ratio a lot. Xdelta and open-vcdiff are both its implementations. They divide the old version into chunks, and use the dictionary for the chunk fingerprints. Xdelta optimizes the generated instructionsand prioritizes speed over compression ratio as a more popular tool.

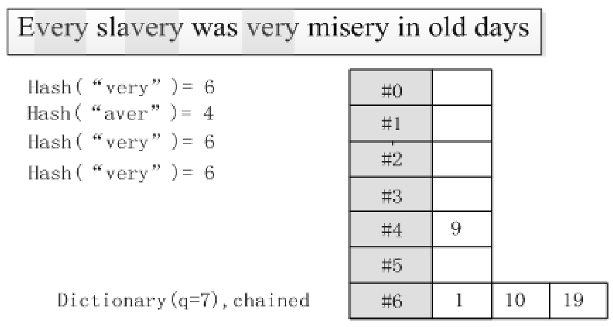

FIGURE II. A DICTIONARY DATA STRUCTURE MAPS HASHES) INTO POSITION

C. *Incremental Updating Algorithm Based on Suffix Tree Search*

Suffix tree is a data structure for storing a string of all suffixes [12]. Suffix tree is the only optimal algorithm that can find substring of arbitrary length p in a string, including small repeat that cannot be identified in the dictionary algorithms. Finding a substring time complexity is O (p). A suffix tree contains all the substrings. Construct a suffix tree time complexity is O (n).

Suffix tree diagram Note that the equation is centered using a center tab stop. BSDIFF [14] is based on suffix searching. When a completely identical block is found in two versions, BSDIFF tries to expand exact matches in both directions so that a similar but not completely identical block is found [1]. Larsson and Sadakane's suffix sorting algorithm is used to construct suffix array for the old version [15]. The patch constructed by BSDIFF is much slightly than the new version [13]. However, the delta can be compressed into a patch with small size, because the delta consists of large series of zeros. And the more similarities between the old and new versions, the more zeros exist in delta. Bsdiff is a famous tool of this method's implementations.

IV. EXPERIMENT AND ANALYSIS

In order to measure the performance of typical incremental updating algorithm, three classical algorithm commonly used are selected to be compared. They are Rsync, bsdiff and Xdelta. Experimental samples for testing are six different files. They are three software's two different version suitable for Linux, Windows and Android system.

The test samples are JDK, http-netware-bin2, Xdelta, Xpdf, UCbrowser and Xchat. Their size and platform are showed in Table I.

TABLE I. TEST SAMPLES

Platform	**Source File**	**Size(KB)**	**Target File**	**Size(KB)**
Linux	Jdk1.5.0.16	6187	Jdk1.5.0.22	6182
	http-netware-bin2.2.24	16659	http-netware-bin2.4.4	22082
Windows	Xdelta3.0.2	107	Xdelta3.0.8	0.308
	Xpdf3.0.3	8903	Xpdf3.0.4	9696
Android	Ucbrowser9.3	12284	Ucbrowser 9.9	14072
	Xchat2.6.6	6530	Xchat2.6.8	7264

In the experiment, the average data compression ratio of software of Rsync, xdelta and bsdiff were 13.2%, 60.3%, 63.6%. On average, the data compression ratio of bsdiff algorithm is the highest. When the 3 and 6 samples of the 2 software update service, all incremental updating algorithm's compression rate is higher, indicating high degree of similarity between old and new versions of the 2 sample; when the update of the issue processing 5 sample, all update algorithm compression ratio of whole body is not high, indicating the similarity between old and new version of the file is relatively low.

When dealing with No. 1, No. 2 and No. 4 sample software update service, all incremental updating algorithm for the overall compression ratio in the middle level, indicating the degree of resemblance between the 3 sample in the new and old file is middle, the scene is the test of incremental updating algorithm performance.

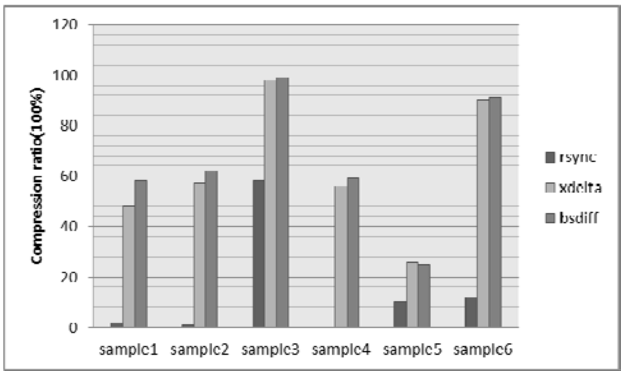

FIGURE III. COMPRESSION RATIO

V. CONCLUSION

Incremental updating algorithm is widely used, and has important significance in the era of massive data. Efficient data structure is introduced to make the incremental update algorithm more practical and optimize. Dictionary structure based algorithm need less memory, the suffix tree based algorithm has small execute cost. Choosing the way to search substring by the actual implementation will fix the problem better. In addition, with the further research, more and more researchers design the corresponding incremental updating process according to the specific application scenarios, so that the incremental update algorithm can better adapt to the application environment.

REFERENCES

[1] G. Banga, F. Douglis, and M. Rabinovich. Optimistic deltas for WWW latency reduction. In1997 USENIX Annual Technical Conference, Anaheim, CA, pages 289–303, January 1997.

[2] M. Chan and T. Woo. Cache-based compaction: A new technique for optimizing web transfer.In Proc. of INFOCOM’99, March 1999.

[3] M. Delco and M. Ionescu. xProxy: A transparent caching and delta transfer system for webobjects. May 2000. unpublished manuscript.

[4] J. Gailly. zlib compression library. Available at http://www.gzip.org/zlib/.

[5] J. Hunt, K. P. Vo, and W. Tichy. Delta algorithms: An empirical analysis. ACM Transactionson Software Engineering and Methodology, 7, 1998.

[6] J. MacDonald. File system support for delta compression. MS Thesis, UC Berkeley, May2000.

[7] T. Suel and N. Memon. Algorithms for delta compression and remote file synchronization. InKhalid Sayood, editor, Lossless Compression Handbook. Academic Press, 2002. to appear.

[8] W. Tichy. RCS: A system for version control. Software - Practice and Experience, 15, July1985.

[9] J. Ziv and A. Lempel. A universal algorithm for data compression. IEEE Transactions onInformation Theory, 23(3):337–343, 1977.

[10] D. Korn, J. MacDonald, J. Mogul, and K. Vo, "The VCDIFF Generic Differencing and Compression Data Format," RFC 3284 (Proposed Standard), June. 2002.

[11] J. Bentley and D. Mcilroy, “Data compression using long common strings,” *in Proc. Data Compression Conference*, Snowbird, USA, pp. 287-295, Mar, 1999.

[12] N. J. Larsson and K. Sadakane, "Faster suffix sorting," *Theoretical Computer Science.*, vol. 387, no. 3, pp. 258-272, July. 2007.

[13] C. Percival, “Naive Differences of Executable Code,” draft paper dated 2003 and available from: http://www.daemonology.net/bsdiff.

[14] A. van Hoff and J. Payne, “General Diff Format Specification,” August 21, 1997. URL: http://www.w3.org/TR/NOTE-gdiff-19970901

[15] J. W. Hunt and M. D. McIlroy, “An Algorithm for Differential File Comparison,” Bell Laboratories, 1976.

[16] N. Samteladze and K. Christensen, "DELTA: Delta encoding for less traffic for apps," *in Proc. IEEE Conference on Local Computer Networks*, Clearwater, USA, pp. 212-215, Oct. 2012.

[17] D. W. Jones, “Application of Splay Trees to Data Compression, “Communications of the ACM, 31, no. 8, pp. 996-1007, 1988.

[18] Ni G, Chen Z, Jiang J, et al. Incremental updates based on graph theory for consumer electronic devices[J]. IEEE Transactions on Consumer Electronics, 2015, 61(1):128-136.

[19] D. Korn, J. MacDonald, J. Mogul, and K. Vo, "The VCDIFF Generic Differencing and Compression Data Format," RFC 3284 (Proposed Standard), June. 2002.