High Efficient Consistency Maintenance Strategy of Real-time String Text Editing Systems

Liping Gao, Wenfeng Tang
School of Optical-Electrical and Computer Engineering, University of Shanghai for Science and Technology, Shanghai, 200093, China. Email: lipinggao@ usst.edu.cn;wenfengtang@st.usst.edu.cn

Abstract

Replicated architecture is widely used in the field of real-time collaborative text editor. The idea of address space transformation provides a new way for concurrency control. During concurrent processing, it retraces the document status back to the state when the operations are generated to maintain consistency. However the previous concurrency processes strategy is based on single characters, the transmission cost during processing is too high since every character is packaged and broadcasted to remote sites after they are created by local site. When the network load is high, especially when the network is unstable, this shortage will be even more obvious. Due to this problem, this paper presents a consistency maintenance strategy based on string editing operations, and proposes the string splitting mechanism combined with the idea of the address space transformation in order to maintain consistency. This strategy increases the transmission efficiency while guaranteeing the consistency of the shared document.

Keywords: *consistency maintenance, address space transformation, string operation, splitting mechanism*

1. Introduction

The rapid development of computer technology and the wide application of network communication have a profound impact on people's production and daily life. Human society becomes more and more collaborative and distributed. Remote collaboration becomes more and more prevalent. The distributed collaborative systems provide a possible solution to achieve this goal [1] [2] [14] [15].Users geographically separated can interact via the network, and the response time is an important evaluation parameter of the interactive network. Since quick response time will improve the user's experience. However the delay of network communication is high and uncertainly. Thus how to ensure a quick respond time in coordination system becomes a factor to be considered in the case of high network latency. Given the current development of the communication network, it is impossible to eliminate the network latency, and we can only improve the response speed on the system level. The emergency of the data replication technology provides a good solution to the problem. By replicating the remote data to the local, users can perform the operations locally without considering the impact of network delay, which ensures a good user's experience [16][17].

An important field of the collaborative environment is the text editor field; multiple users edit the same document freely through a network, and how to maintain all the intention of all the multiple users while maintaining the consistency of the document is the focus of the study. In the collaborative text editing field, there are causally dependent and concurrent relationships among the operations. The causal dependence refers to a causal relationship between the operations and the concurrency means independent relationship between the operations. How to handle the concurrent relationship between the operations, and how to maintain the operations' intention, are

the focus of the research in the text editor field.

Most of the previous text editing systems are to process a single character, and provide corresponding control strategies to maintain consistency and achieve some results, such as the COT algorithm, GOTO algorithm, AST algorithm and so on[2][3][4][15], but few involve in the string editing system.

The string text editing has practical examples, such as the copy and paste operations. In the single character text editing system, a single character is transported between the sites; each site receives a character from another site, and will call the control algorithm to handle it to achieve the desired consistency. In the editing environment with little number of characters, the algorithm can maintain consistency at all the sites successfully. However, when processing environment with massive characters, the algorithm has an obvious shortage, because the transmission efficiency is not so high, especially in the case that network is unstable. If the strings are transported between the various sites, the transmission efficiency will have an effective improvement, especially in environment with huge amount of characters.

This paper is divided into the following sections: The second chapter describes some related work in the field of text editors, the third chapter arises the consistency maintenance issues, the fourth chapter describes the storage structure of the nodes in the string text editor, the fifth chapter gives consistency maintenance strategies and algorithms, the sixth chapter gives specific examples to verify the correctness of the algorithm, the seventh chapter analysis the efficiency of algorithm, the eighth chapter gives the acknowledgement from the practical aspects, and the final chapter give a summary and outlook of article.

2. Related works

2.1. Consistency maintenance strategy

Data replication technology is widely used in group editing systems [1]; a group of users share the document and edit the text freely. Due to the operations from multiple users may appear concurrently in the same region or the same characters and different execution order may lead to different execution results without the concurrency control algorithm. Consistency means that the document of all users is all the same finally after the editing of multiple users. In an ideal group system, whether the operations are concurrent or not, what the execution order is, the document gets the same result, which meets the CCI consistency model [4].

In the single character text editing system, previous works propose special strategies to solve the inconsistency caused by the concurrent operations, including operation transformation (OT) and address space transformation (AST). The former starts from the operation itself, transforms the operation to the execution form properly through a series of conversions, and achieves consistency maintenance; while the latter starts from the document status, and by the mark and retrace strategy, and execute the operation. Both of them will solve the inconsistency caused by concurrency.

Operation Transformation strategy has been developed rapidly during the past twenty years, and now supports a variety of applications, including group undo[5][6][10][11], group awareness [8], operation notification and compression [12], spreadsheet and table centric applications[14], HTML/XML and tree structure document editing system[3][7], word processing and slide creation [7][11][14], trans-parent and heterogeneous application-sharing [1][9][14], and mobile replicated computing and database systems etc.

Operation transformation starts from the operation’s prospect, and the operation intention will be retained through transformation correctly. From the initial dOPT algorithm, the operation

transformation technology have a great development, like ET and IT transformation algorithm, COT algorithm, GOT algorithm [2][3][4][5].The series of the proposed algorithms can not only solve the group do problem, but also provide better solutions to the group undo problem [5] [6] [14].

The document status of the operation execution and that of the operations generation may be not identical; while the prerequisite for the proper execution of the operation is that two states are the same. AST is proposed to solve the problem from the view of document status; by retracing the document back to the state when the operation is generated [4] [13] [16]. Compared with the operation transformation, this strategy looks more intuitive. This article uses the address space transformation technology to solve the string text editing problems.

2.2. Address space transformation

In this paper, AST is used to solve the problem. Thus, here we give a brief overview of the strategy. AST algorithm is based on the mark and retrace algorithm and provides a new idea to solve the complicated conflict. Different from operation transformation strategy, AST algorithm starts from the status of the document to solve the consistency maintenance problem. State vector timestamp is proposed to judge the causal relationship between the operations.

When an operation reaches a site, the document status may have been changed different from the operation`s generating state, thus executing these operations directly may lead to inconsistency. During the processing of the AST strategy, it hides the possible impact and retraces the document status back to the status when the operation is generated by mark and retrace process, and then the operation can be performed in the new document, After the execution of the operation, the document status is then retraced to the concurrent status. In the implementation process, the function “retracing” will hide the execution effect that may affect the operation, and the function “rang-scan” is used to find a specific location for execution, and function “retrace” is called to include the operation results into the document [4].

A document has a linear structure containing a number of characters, with each character be associated with multiple operations, but each operation has only one character node. The timestamp as well as the mark of the effective or ineffective are stored in corresponding character node, which indicates whether the current node is visible or not at the current moment. Fig.1 is an example of the address space strategy diagram, in this figure, there exists three nodes corresponding to the operations a, b, c and the timestamp and effective or ineffective identifiers are also depicted. In such a case, the document presented to the user is "ac ".

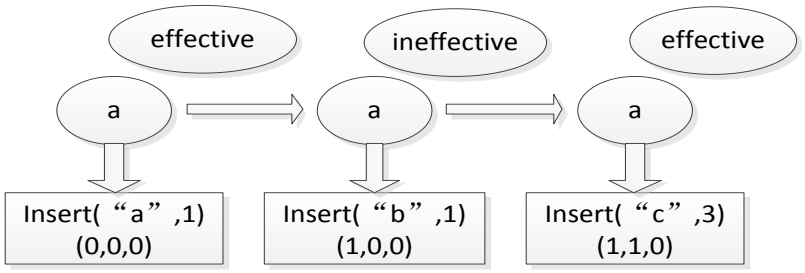

Fig.1 the address space strategy diagram

3. Consistency maintenance

3.1. Conflict problems

In the string text editor, the basic operations are Insert and Delete operations of strings or characters. Compared to the single character editor, the Insert and Delete operations are relative complex, and there exists large differences in the type of the operations. In the single character text editor, the Insert position is nothing more than before or after a character, and the Delete operation is only to delete a specific character, but in the presence of the string editor, the Insert position is not only before or after a string, but also has the case that insert a string into an existing string; the Delete operation may not only delete a particular string, but also deletes a specific character from a string. For the situations that similar to the single character, the string can be seen as a whole and transplant the single character processing strategies directly for the concurrence control; as for the latter, the single character processing algorithms cannot solve the problem, and need to find a new strategy.

Supposing that the operations O_1 and O_2 are from different sites, O_1=insert ("hello", 1), O_2=insert ("world", 1), O_1 and O_2 want to insert different strings in the same position. Without control strategy, the different execution sequences of the two operations will lead to different results: if O_1 completes first, the final document is "hello world", while if O_2 completes first, the result will be "world hello". Obviously violates the CCI model since different execution result is achieved. Another example, assuming that there is a document "th a nice day", this is a bad statement, There exists an operation intending to inserted the string "is" behind the "th" to fill the statement to a whole statement, while there also exists another operation hoping to insert the character "at" in the same location. It is clear that the two operations are also concurrent, and if they are executed directly there will still be consistency puzzle; in addition, if an operation has inserted a character "hello" before this statement, the document has changed, execute the operation directly will lead to inconsistency, the solutions to these problems have not been totally dependent on the strategy processing the signal character.

This paper proposes corresponding strategy for the cases above mentioned above that signal character strategies cannot solve effectively and it can solve the problems in the string group editing system basically.

3.2 Modified operation definition

Given the complexity of the string operations, the definitions of the operations need to be modified to adapt to the new situations:

Insert ("string", pos, left):"string" represents the string to be inserted;

"pos" represents the Insert position;

"left" is a symbol, there are two meanings: if left is 0, it means that there will be a string to be inserted in the position "pos"; if left is not 0, it means that the string will be inserted into the existing string, and the "left "represents that the string will be inserted in the position counting from left.

Delete (pos, n, left):

"pos" indicates the position of the strings to be deleted;

"left" is the symbol, if it is 0, it means to delete the string at the position "pos";if left is not 0, it means to delete n characters from the string at the position "pos" from the position "left" .

For example, there is a document "hello world", O1=insert ("hello", 2,0) represents the string "hello" will be inserted after the string "hello", and the document becomes "hello hello world"; O2=insert ("China ", 1,2) represents to insert the string "China" in the string "hello", the

document will be "hChinaello world"; O3=delete(1,2,0) means to delete the first position string "hello", the document becomes "world"; O4=delete(1,2,2) means to delete two characters from the first string "hello" from the second character, the document becomes "hlo world".

3.3 Analysis of operation execution cases

According to the previous definition of the operations, the Insert operation has two parameters, one representing the location to be inserted, and another representing the string to be inserted. Thus, there will be the following situations to be considered: if “left” is 0, the treatment is relatively simple; while when "left" is not 0, two situations will be considered, the parameter "pos" is equal or not equal. If equal, that means to operate in the same string, if not that means to operate in different strings. The following examples corresponding to the discussion above are given for better understandings.

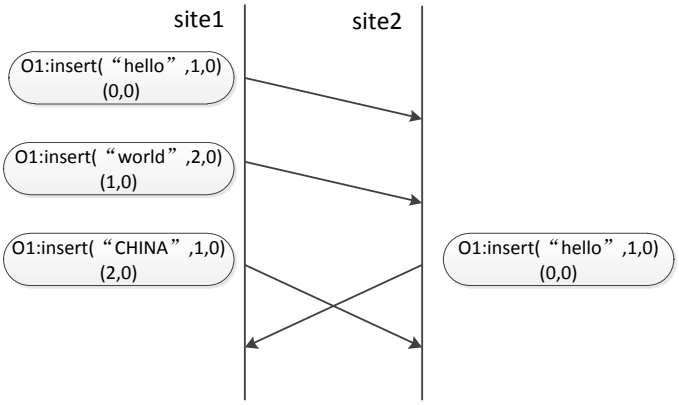

Fig.2 the overall string insertion

In Fig.2, the string is inserted between strings other than the internal string. Based on the definition of the previous operation, all the operations are the whole string operation, all of the insertion is executed between strings, and the integrity of the string is not destroyed. Operations O_1 and O_2 are to insert string in the document, O_3 inserts the string "CHINA" in the first position of the document, and O_4 inserts the string "USA" in the first position, too.

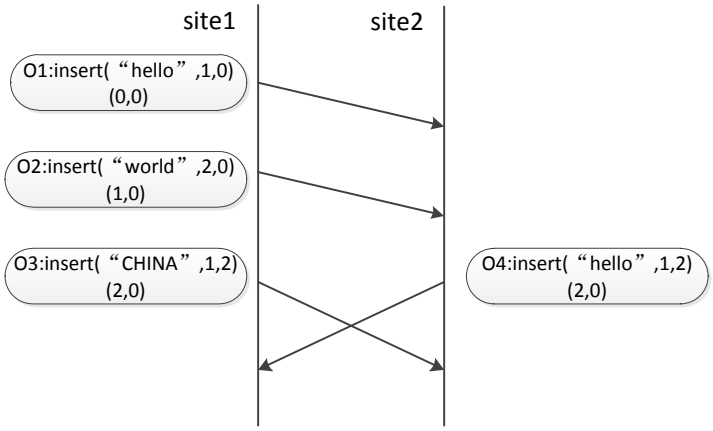

Fig.3 Insert string into the same string

In Fig.3, this situation corresponds to the case that the parameter "pos" is equal. O_1 and O_2 are the same to the situation described above, and intend to insert the strings between the strings. O_3 and O_4 are to insert the string internal the specific string. O_3 means to insert the string "CHINA"

in the second position of the first string, O_4 means to insert the string "USA" in the second position of the first string.

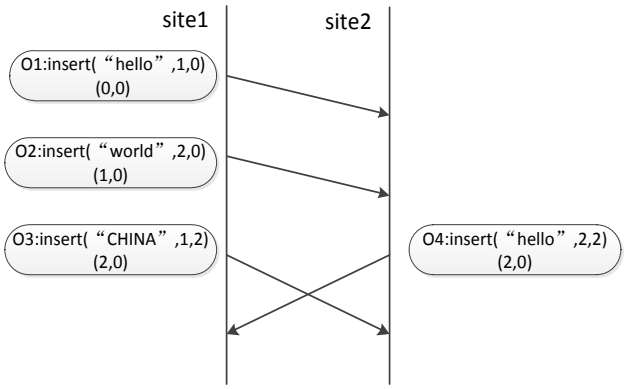

Fig.4 Insert string into different strings

In Fig.4, the situation is the case the parameter "pos" is unequal, and is in different strings. O_3 inserts the string "CHINA" in the second position of the first string; O_4 inserts the string "USA" in the second position of the second string.

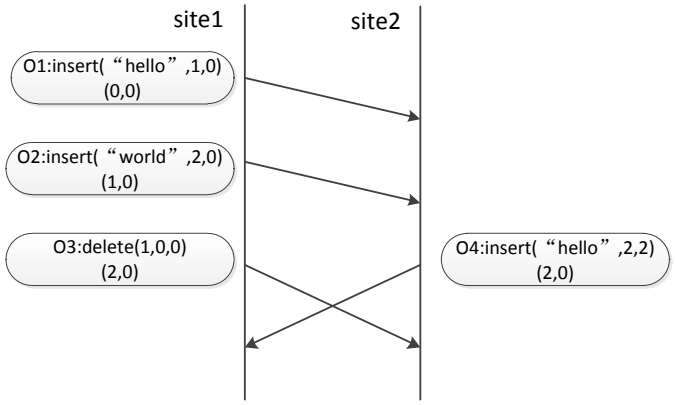

Fig.5. Delete the entire string

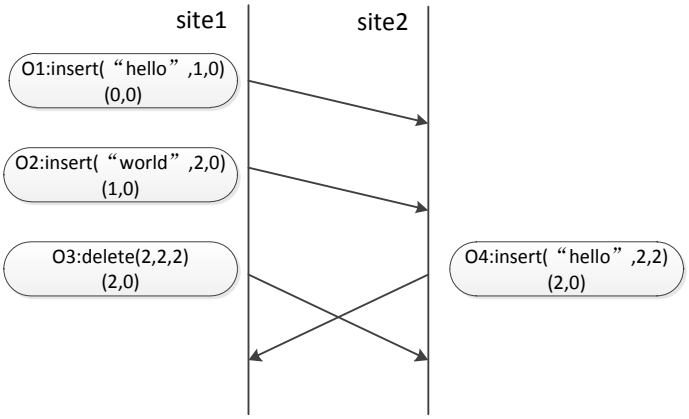

Fig.6 Delete a specific character from a string (a)

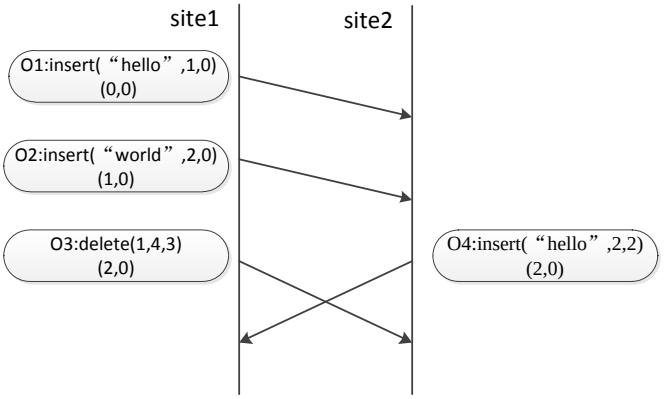

Fig.7 Delete particular character from a string (b)

Now consider the more complex situations that the insertion and deletion are mixed, such as the two cases above. In Fig.5, O_3 deletes string at the first position. In Fig.6, O_3 aims to remove two characters from the second position from the first string. At site 1, if O_3 completes first, the address space will be changed when O4 executes. In Fig.7, O_3 means to delete characters spanning in two strings.

4. Storage structure

After the execution, the operations need to be stored for latter processing and comparisons, such storage structure is called history buffer (HB) [1]. AST is proposed to give a new solution to the concurrent operations to maintain consistency, and also propose a new storage structure to store the operations. The linear storage structure is a good solution to the store the single characters [4]; this paper uses a similar storage structure to store the strings on string level.

While handling the string, this paper adopts such an approach: when the users of each site edit the document, when a word is completed, each character of the word is packaged and then broadcasted to all the other sites; this is different from the traditional way that produces a signal character and then broadcasts it soon. In this way it is possible to improve the transmission efficiency and reduce the error. The space or punctuation between words is the separators.

This paper extends the storage structure and the operation is stored based on the following agreement: when the Insert operation is carried out between the strings ,the string is directly stored as a node, and if the Insert operation is carried out in a string, the operation will be attached to the node representing the string; as for the Delete operation, this paper adopts a similar strategy, and the Delete operation is attached to a specific string; as for the delimiters between strings, this paper treats them as independent node.

According to the agreement above, assuming two sites, Site1 and Site2, and a set of operations O_1, O_2, O_3, O_4, O_5, and the execution procedure is shown in Fig.8, a diagram will be got and shown in Fig.9, the final status of the document shall be "hello world! “.

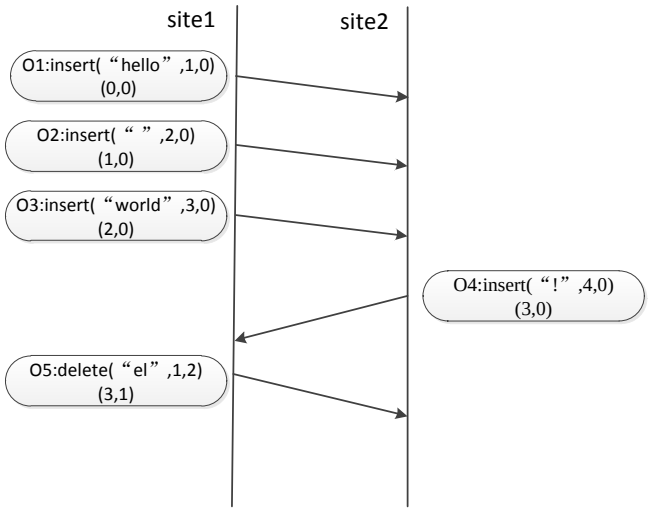

Fig.8 Operation example

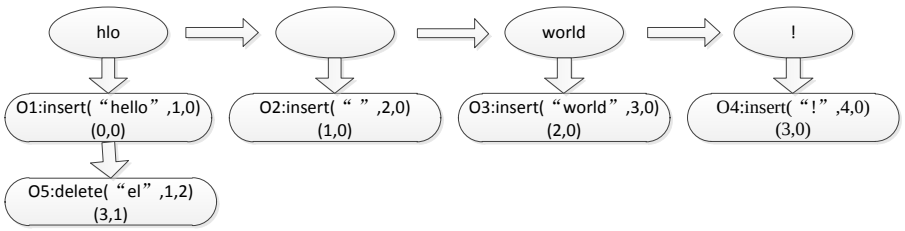

Fig.9 diagram of operation storage

For each node, the structure is shown below:

Fig.10 node structure

"string" indicates the content of the node ;

"flag" is the node’s identifier and indicates whether the node is effective or not and its value is "effective" or "ineffective";

"ptonext" indicates a pointer to the next node , if the next node do not exist, it will be null;

"ptodown" indicates a pointer pointing to a node associating with it, if the associating node do not exist, it will be null too;

For example, , the first node in Fig8 is "hllo". It’s flag is "effective", "ptonext" points to the node "e", "ptodown" points to a node " " (This is a space).

5. Control strategies

5.1 analysis of operation relationship

This paper has extended the operation previously and the traditional definition of concurrency is not suitable for the string text editing system, and it needs to be redefined.

Assuming that there are two operations, O_1=insert ("string1", pos1, left1), O_2=insert ("string2", pos2, left2), and then there should be three conditions as follows:

1. pos1 = pos2 ,left1 = left2
2. pos1 = pos2 ,left1 != left2

3. pos1!= pos2

Now the relationship is defined as follows:

Assuming that operation O_1 = insert ("string1", pos1, left1), O_2 = insert ("string2", pos2, left2), then $O_1 \parallel O_2$, if and only if pos1 = pos2, left1 = left2.

In the cases above, there exists concurrency in case 1, and there not exist concurrency in case 2 and case 3. As for the causal dependency, this paper continues the previous definitions and symbols.

Then, according to the definition, it should be $O_1 \rightarrow O_2$, $O_3 \parallel O_4$ in Fig.2.

5.2 Analysis of operation execution

In the single character transmission processing system, the document refers to the view of the current state, and that is a series of characters presented to the user, and the operations are based on the concurrent character. But in the string processing, the user's operation is based on strings, and the AST transplantation strategy is an algorithm that replaces the single character with the whole string.

This paper proposes the splitting strategy to resolve the inconsistencies that may appear. When there exists operation involving in internal insertion, split the string basing on the operation parameters and then use the AST algorithm to solve the problem. Because AST algorithm and the timestamp is inseparable ,and the timestamp represents the causal or concurrency relationship between operations, in order to keep the causal relationship between the characters after splitting ,the timestamp of the character after the split remains the same to that divided before.

5.3 Design of consistency maintenance algorithm

For the case shown in Fig.2, this paper modifies AST algorithm to adapt to the new environment. When handling these operations, the string is viewed as an indivisible unit; each operation maintains the original time-stamp. It should be noted that, because the operations have been modified to adapt to the string algorithm, there needs an appropriate conversion here to adapt the AST algorithm by discarding the parameter "left". For example, the operation "insert ("hello", 1, 0)" is converted into "insert ("hello", 1)", and the time stamp remains the same.

For the case shown in Fig.3, this paper views the string in the position "pos" as a new text document and operate the operations at the new document. Since it is an internal Insert operation, the string is split based on the Insert position, and then two substrings are got and the substrings get the same stamp of the string split before. For example, O_3 and O_4 intend to insert "hello" in the document, and both mean to insert in the second position. In such a case the second string "hello" will be split to substrings "h" and "ello", and then the operations are executed, and then the case is similar to insert a new string between two strings, then adopt the strategy in case 1.

Of course, there exists Insert operations at different positions, as shown in Fig.4, the strategy is similar to case 2, and the difference is that the inserte is not in the same document and there exists no concurrency.

The case shown in Fig5 and Fig6 can be handled by using similar strategies, too. In Fig.5, the string to be deleted is viewed as a whole and then the AST strategy is transplanted directly. In Fig6, the string "world" will be split at the position "left" and "left+n" respectively.

The similarity of applying the splitting strategy in the Delete and the Insert operation is the maintenance of the time stamp, and the time stamps of all the substrings are same to the previous string. The difference is that the split in delete may obtain more than two substrings. The length of the string to be removed will be considered, just like the cases shown in Fig.6 and Fig.7. If the

delete operation is done in one string, split the string into three substrings according to the position parameters "left" and "left + n", and the substring in the middle is the string to be deleted and will be marked as ineffective. If two or more strings are involved, split the string according to the parameter "left" first and get two substrings, and find the last string to be deleted according the length of the to-be-removed string and split the string, and the strings between the second substring of the first string and the first substring of the last string all set to be ineffective.

Assuming that there is a document "welcome to shangghai", and an operation means to delete the character "g" from "shangghai", and then the string “shangghai” is split into three substrings "shang", "g", and "hai", the substring "g" is marked as ineffective. Another example is that an existing document "thank youu vvery much", and an operation wants to correct the mistakes by deleting the string "u v", and then the string "youu" will be split into two substrings "you" and "u", the string "vvery" will be split into two substrings "v" and "very" respectively, and then the substrings "u" and "v" is marked as ineffective. The timestamp is taking the strategy above.

As for the copy-paste operation, the user means to paste the specific content at a specific location, and to follow the Insert operation strategy to handle with it. The similarity between the Insert and copy-paste is that both need to find the insert position, and the difference is the complexity of the paste content. If the paste string targets only one string, it can be solved according to the strategy discussed previous, but if the paste targets more than one string, there needs to split the operation into more insert by the non-word character. For example, an operation means to paste the "hello world", there needs to spilt the string into three insertions, and each insertion have the same timestamp, which is the same with the entire string split before.

The operation will be executed according to the following algorithm: Firstly, judge the type of the operation, if it is Insert and the Insert position is between two strings, then call the function “Insert_Execute” to insert the string into the document; if the Insert position is in one string, then call the function “Divide_Insert” to insert the string into a specific string; for the Delete, the operation’s type also needs to be considered, when deleting the entire string, call the function “Delete_Execute” to delete the entire string; when deleting some characters from one string, the function ”Divide_Delete” will be called to delete the specific characters; if it is a paste operation, the function “Divide_Paste” will be called to paste the strings.

According to the analysis above, the basic algorithm is presented as follows:

```
Procedure1：String_Execute(S, O) // Handling operations, S represents the current document status, and O is the operation to be executed.
Begin
If (O is insertion)
    If（O.left=0） //insertion is among the strings
        Insert_Execute(S, O)
    Else //insertion is internal the string
        S=HB.pos //get the string at position “pos” as the document S
        Divide_Insert(S, O)
Else if (O is deletion) //O is deletion
    If（O.left=0） //delete the entire string
        Delete_Execute(S, O)
    Else //delete characters internal the string
        Divide_Delete(S, O)
Else if (O is paste) //O is copy-paste
    Divide_Paste(S, O);
End
```

The implementation of the Insert is similar to the single character insert process, but the operations in the AST algorithm only have two parameters: the insert character and insert position. This paper redefines the operation, it is necessary to convert the operation to a form suitable for AST algorithms, the function "Convert" achieves this goal, after that the AST control algorithms can be called. The algorithm is described as follows:

```
Producer2：Insert_Execute(S, O) // Function for insertion between the strings, the document status is S, insertion O
Begin
O' <- Convert (O)  //convert the operation into a form suitable for AST
Control-Algorithm (S, O')   //call the AST control algorithms
End;
```

The case that inserts a string into an existing string is more complex, and there needs to split the string based on the insert position, the timestamps are the same to that split before. After the division, the substrings are viewed as a new document, and then the situation is equal to the situation that inserts strings between strings, and then the strategy that inserts the string as a whole can be used. The algorithm is described as follows:

```
Producer3：Divide_Insert(S,O)   //Function is inserting the string internal the string S and O.string is to be inserted
Begin
s1, s2 <- Divide(S) //split the string basing on the insertion position and get two substrings
S'= {s1, s2}
Insert_Execute(S', O)
End
```

The strategy that deletes the entire string adopts the single character strategy, that is the mark and retrace strategy, and the characters are marked ineffective. The algorithm is described as follows:

```
Producer4：Delete_Execute(S, O) // Function for deleting a string, the document status is S, O is deletion
Begin
Set HB.pos ineffective // mark the characters to be deleted ineffective
End
```

When there needs to delete some characters from a string, first judge whether the Delete involves multiple strings: if not, the string at the delete position is split into three substrings, and the middle substring is just the string that needs to be deleted and is marked ineffective; if so, judge the number of the involved string firstly, and then find the first and the last string involved, then split them, and all the characters involved should be set ineffective. The algorithm is described as follows:

```
Producer5：Divide_Delete(S, O) // function deletes some characters from the strings; the string need to be deleted is O.string
Begin
If (O.left+O.n <= S.len) //delete some strings internal the string
    s1, s2, s3 <- divide(s) //split the string basing on the insertion position and get three substrings
    Set s2 ineffective
Else            //deletion involves more than one string
    s1, s2 <- divide(s)
    If (n-(S.len-left+1) < HB. (pos+1).len) // deletion involves two non-null strings
```

```
        s' = HB.(pos+1)
        s3, s4 <- divide(s')
        Set s3 ineffective
    Else      // deletion involves two non-null strings
        Count = judge (n-(S.len-left+1)) //counting the involved number
        S' = HB. (pos +count+1)  //find the last string involved
        s3, s4 <- divide(S')
        Set HB.pos, HB.pos+1 ...HB. (pos + count+1) ineffective
        Set S3 ineffective
End
```

While handling copy-paste, the strings will be split into multiple Inserts and then be inserted into the document orderly. When inserting, the function "Insert_Execute" will be called and the algorithm is as follows:

```
Producer6：Divide_Paste(S, O) // function to achieve pasting a string in a document
Begin
O1, O2...Oi <- divide (O) // operation is split into three of insertions, and executed orderly
Repeat
Insert_Execute(S,Oj)
Until j=i
End
```

6. Case analysis

Suppose that there is a blank document, three users and a set of operations and the operation sequences is shown in Fig11, now this section will adopt the algorithm above to solve a real case.

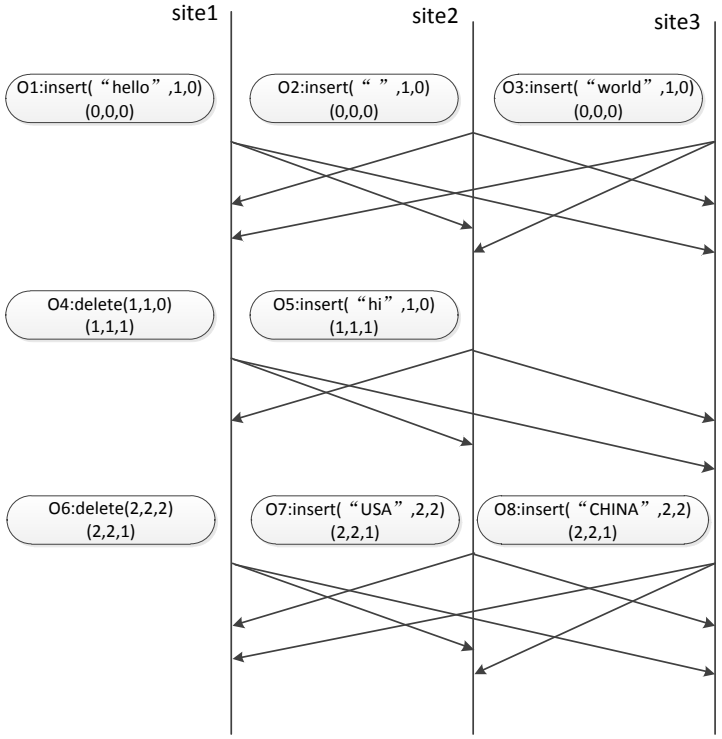

Fig.11 operation execution example

Assume that the initial state of the document is of site1 is S_0, by observation, the operations will be executed in three stages: at the first stage, the operations O_1, O_2, O_3 are executed respectively at each site to obtain the document S_1; at the second stage, O_4, O_5 are executed to get the document S_2; at the third stage, O_6, O_7, O_8 are executed to get the last final document. The following is the execution process of each stage.

Firstly, let's analyze the execution of the first stage. According to the definition of concurrency, all the inserts are to insert strings between strings and $O_1 \parallel O_2$, $O_2 \parallel O_3$, $O_1 \parallel O_3$. At site1, O_1 is executed immediately. When O_2 arrives, the document at this time has changed compared to that when O_2 generates, and the function "insert_Execute" is then called to find the correct location of O_2, and thenO_3 is executed in the similar way. After executing O_3, the document state at site 1 is "hello world". Similarly, after all the operations are executed at site2 and site3, the document is same to that at site1. So at the first stage, after the completion of O_1, O_2 and O_3, the document of each site reaches "hello world".

Secondly, at the second stage, both O_4 and O_5 are conducted internal a string; at site1, O_4 deletes the first string "hello" firstly, and the document becomes "world ". When O_5 arrives, since the document has changed, the function "insert_Execute" is called to retrace the state back to the state S_1 when O_5 generates. After the retrace process, the string "hi" is inserted into the document, and then retrace process is called to retrace to current document status to contain all the results. As last, the document becomes "hi world". The handling of the operations at site2 and site3 is similar with that at Site1 in a similar way, thus we don't give detailed description here. The final results are the same with that at site1.

Finally, at third stage, all the operations are concurrent with each other and all the operations are executed in the second string of the document. The second string is viewed as a new document S. At site1, O_6 is executed firstly, and the function "Divide_Delete" is called to split the string into three substrings "w", "or" and "ld" firstly basing on the parameters "n" and "left" ,and then the execution of the operation at this moment is similar to that in second stage. At last the string at the second position becomes "wUSACHINAld", and the final document at site1 is "hi wUSACHINAld". The operations at site2 and site3 execute similarly, and after execution, the documents are the same with that at the site1.

Here exist more complex cases, as is shown in Fig.6. At site1, since $O_4 \parallel O_5$ and the characters O_5 targets to delete are involved in two strings, when O_5 is executed, the function "Divide_Delete" will find the position of the string firstly. Since the string to be deleted has surpassed the two strings, the string "hello" is split to "he" and "llo" firstly, and "llo" is set to be ineffective. There are still characters that are not deleted, and then the next non-empty string "world" is split into "w" and "orld", and the "w" is set to be ineffective. When O_4 is executed, the function "insert_execute" will find the position of the string "USA" and the final document of the site is "heUSA orld". Similarly, the results of site2 and site3 are the same to that at site1 after their execution.

7. Efficiency analysis

The analysis of the algorithm shows that the cost of the algorithm is mainly in two aspects, one is the retracing process, and the other is the execution of the operation. As for the attaching process of the operation to the corresponding node and the change of node identifier they can be completed in O (1). Thus the following part focuses on the two main aspects.

The first section is the analysis of the execution efficiency. In order to execute the operation, the algorithm needs to find the position of the operation first. For Insert operation the function "range-scan" selects the Inserte position, and for the Delete operation, there only needs to modify

the effective or ineffective flag of the nodes, and it is completed in time O(1), the complexity of the function "range-scan" is O (m), where m is the number of ineffective nodes [4]. For the document has a linear structural organization, the process seeking the position is the process of traversing the list and the complexity is O(n), where n is the number of the nodes in the list.

The following is the analysis of the efficiency of retracing process. Retracing process is to retrace the document back to the status corresponding to the time stamp SV when the operation is generated by changing the node's identifier, only when all the operations attaching to nodes are causally before the operation at SV, the node's identifier will not be changed, and if not, it will be changed. The operations that have not been executed are stored in the request queue Q, and will be removed from the queue after their execution. Assuming that the length of Q is h, that's to say the retracing process will modify at most h nodes' identifier. Assuming that the number of operations attaching to each node is d, then the complexity of first retracing process is O (d*h). The first retracing changes the identifier of the node, and will recover them in the second retracing; the complexity of recovery process is O (h +d).

Based on the above analysis, the actually complexity of the execution of the Insert operation is O (m +d* h+ h +d), and the Delete is O (d*h+ h +d).

Assuming that there are k strings, with each probabilistic length of the string is n, h and d are consistent with the definition previous. In the single character editing system, the complexity of Insert operation is O (m*n +d*h*n +h*n + d*n), and the Delete operation is O (d*h*n + h*n + d*n). And in a string text editing system, the Insert operation complexity is O (m +d*h + h + d), and the Delete operation is O (d*h +h +d). Obviously the latter will be more efficient.

8. Experiments

To verify the algorithm, an experiment is made. The experiment simulates a document editing process; three users edit the same document simultaneously. First simulates the single character mode. Each user generates a character, and each insert is randomly inserted into a specific position in the document and the delete is also the same. Then the users simulate the string mode, the same three users, reading strings from the previous document, and write to a new document, and finally the two documents are consistent. The time of the completion of the same document is recorded in both editing modes. Five groups are experimented; each group has a different number of characters.

The experimental environment is described in the following:

Platform: ubuntu11.10

Language: Linux C

CPU clock speed: 2.26GHz

Memory: 2G

The following chart presents the experiments results.

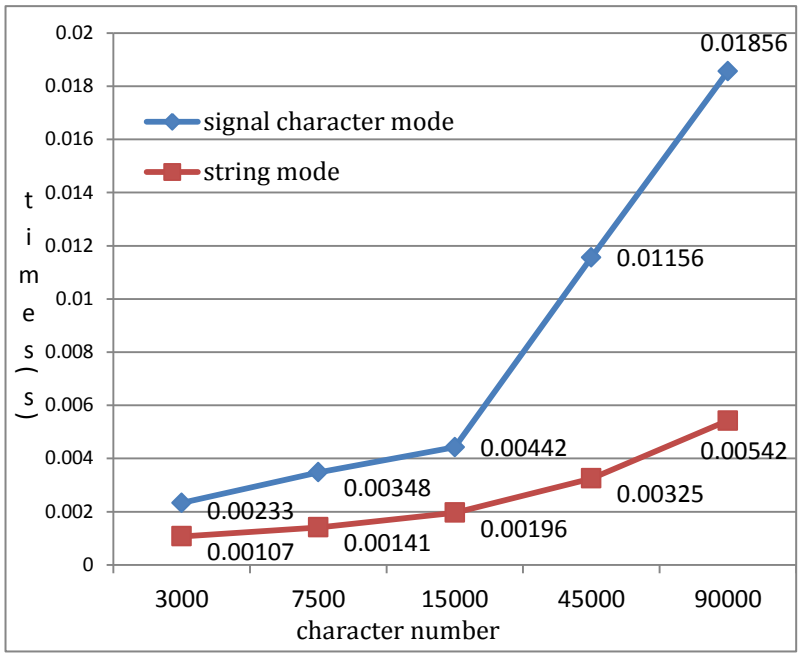

Fig12 consuming comparison between character mode and string mode

From Fig.12, we can see that the string mode is a less time consuming mode in the completion of the same document compared to the signal character mode, and it is more efficient, especially when there are a large number of characters, the advantage is more obvious. This experiment verifies the efficiency improvement of the new algorithm.

9. Conclusions and future work

The AST algorithm used in the single character text editing system has been proved to be a good solution to the single character editing problems [4], but it is powerless in the string text editing system. This paper proposes a splitting strategy combined with AST to solve the problem, and compared to the previous strategy, it improves the processing efficiency greatly and at the same time meets the CCI consistency model [5].

This article firstly analyses various issues that need to be considered in the string text editing system, and lists various situations that may appear in string handling, including the overall strings Insert and Delete, Insert and Delete internal strings, as well as the more complex cases where delete involves more than one string, as well as copy-paste operations. Then it discusses the solutions to these problems and finds that when handling the whole string operations, in some circumstances, the single character processing strategy still works, such as the whole strings Insert and Delete, because the single character is the string whose length is one. But when it comes to the operation involving the string internal strings, the previous strategy becomes powerless. The next part of the paper details a lot of solutions to such problems and proposes an algorithm based on splitting strategy, and also analyses the efficiency of the algorithm to prove the efficiency improvement of the algorithm.

This paper extends the definitions of operations; both the Insert and Delete operations have a mark to distinguish the different types of the operation, that is the operation targeted at the overall string or the operation targeted at the internal string, and considers them separately in algorithm design section. At the same time the paper continues the logic of the dependent and concurrent operations, but to the concurrent relationship, the paper redefines it to make the

definition be more complete after the introduction of mark attaching to it. In the algorithm design phase, this paper analyses the questions raised at the beginning of the article respectively, and verifies it through specific examples and practical experiment.

A complete text editing system is not only able to execute the operations, but also should have error modification. All the contents in this article only settle the process to execute the operation, but do not involve the undo operations, this is the work needs to be done later.

Acknowledgement

The work is supported by the National Natural Science Foundation of China(NSFC) under Grant No. 61202376, Shanghai Natural Science Foundation under Grant No. 15ZR1429100, Innovation Program of Shanghai Municipal Education Commission under Grant No. 13YZ075, Shanghai Key Science and Technology Project in Information Technology Field under Grant No. 14511107902, Shanghai Leading Academic Discipline Project under Grant No. XTKX2012, and Shanghai Engineering Research Center Project under Grant No. GCZX14014 and C14001.

References

[1] C.A. Ellis,S.J. Gibbs .Concurrency Control in Groupware Systems. 1989 ACMO-89791~317-S/89/0005/0399.

[2] Chengzheng Sun, Clarence (Skip) Ellis. Operation Transformation in real-Time Group Editors: Issues, Algorithms, and Achievements. In Proc of 1998 ACM Conference on Computer-Supported Cooperative Work, Seattle, USA, Nov.14-18, 1998.

[3] David Sun, Chengzheng Sun. Operation Context and Context-based Operational transformation. CSCW'06, November 4–8, 2006, Banff, Alberta, Canada.

[4] Ning Gu, Jiangming Yang and Qiwei Zhang. Consistency Maintenance Based on the Mark & Retrace Technique in Groupware Systems. GROUP'05, November 6-9, 2005, Sanibel Island, Florida, USA

[5] Chengzheng Sun. Undo as Concurrent Inverse in Group. Editors ACM Transactions on Computer-Human Interaction, Vol. 9, No. 4, December 2002.

[6] ABOWD, G. AND DIX, A. 1992. Giving undo attention. Interact. Comput. 4, 3, 317–342.

[7] S. Xia, D. Sun, C. Sun, D. Chen, and H.F. Shen. Leveraging single-user applications for multi-user collaboration: the CoWord approach. In Proc. of the ACM Conf. on Computer-Supported Cooperative Work, Nov. 2004.

[9] N. Vidot, M. Cart, J. Ferrie, and M. Suleiman. Copies convergence in a distributed real-time collaborative environment. In Proc. of the ACM Conf. on Computer-Supported Cooperative Work, pages 171-180, Dec. 2000.

[10] A. Prakash and M. Knister. A framework for undoing actions in collaborative systems. ACM Trans. on Computer-Human Interaction, 4(1):295{330, Dec.1994.

[11] N. Preguica, M. Shapiro, and J. Legatheaux Martins. Automating semantics-based reconciliation for mobile databases. In Proceedings of the 3th Conference Francaise sur les Systems d'Exploitation, Octo 2003.

[12] H.F. Shen and C. Sun. A flexible notification framework for collaborative systems. In Proc. of the ACM Conf. on Computer-Supported Cooperative Work, pages 77-86, Nov. 2002.

[13] B. Lushman and G. Cormack. Proof of correctness of Ressels adOPTed algorithm. Information Processing Letters, (86):303-310, 2003.

[14] D. Li and R. Li. Transparent sharing and interoperation of heterogeneous single-user applications. In Proc. of the ACM Conf. on Computer-Supported Cooperative Work, pages 246-255, Nov. 2002

[15] Hansu Gu, Xing Xie, Qin Lv, Yaoping Ruan, and Li Shang. Etree: Effective and efficient event

modeling for real-time online social media networks. In Web Intelligence and Intelligent Agent Technology (WI-IAT), 2011 IEEE/WIC/ACM International Conference on, vol. 1, pp.
300-307. IEEE, 2011.
[16] Hansu Gu, Mike Gartrell, Liang Zhang, Qin Lv, and Dirk Grunwald. AnchorMF: towards effective event context identification. In Proceedings of the 22nd ACM international conference on Conference on information & knowledge management, pp. 629-638. ACM, 2013.
[17] Hansu Gu, Haojie Hang, Qin Lv, and Dirk Grunwald. Fusing Text and Frienships for Location Inference in Online Social Networks. In Web Intelligence and Intelligent Agent Technology (WI-IAT), 2012 IEEE/WIC/ACM International Conferences on, vol. 1, pp. 158-165. IEEE, 2012.

Authors

Liping Gao

She graduated from Fudan University, China with a PhD in 2009 in Computer Science. She received her BSc and master degree in Computer Science from Shandong Normal University, China in 2002 and 2005 respectively. She is doing her research work in University of Shanghai for Science and Technology as an assistant professor. Her current research interests include CSCW, heterogeneous collaboration, consistency maintenance and collaborative engineering.

Wenfeng Tang

He is a postgraduate student in University of Shanghai for Science and Technology. He obtained his BSc degree in Electronic Information Engineering from Henan University of Science and Technology, China. His current research interests include CSCW, collaborative design and collaborative computer.