Log compression of move operations

David Barrett-Lennard
10 April 2007

Introduction
In this paper we investigate algorithms for compressing move operations in the log (or history buffer).

Log compression is extremely important. For example, during the normal editing of text most operations only insert or remove individual characters. For the purposes of interactive collaboration, such operations are assigned unique (s,t) values. Without log compression these single character operations would be very expensive to store in the repository.

It is important to understand that log compression must be lossy because we ignore the (s,t) values of the individual operations in order to achieve compression. Without that it wouldn't be possible to convert single character operations into stringwise operations.

It follows that we don't expect merging to be compatible with IT,ET. Instead we are only interested in the overall effect on the document state.

Statement of the problem
Let the log consist of contextually serialised operations [O1, O2, ..., On]. We are interested in find an equivalent operation O. ie

$$O \sim [O1, O2, ..., On]$$

It suffices to deal with the case of merging a pair of contextually serialised operations. Apply pairwise merging of operations n-1 times should achieve the desired result.

Effects document versus the working set document
To support IT/ET move operations are defined with respect to the effects document – ie the hypothetical document in which extractions never actually take place, and instead characters in the effects document are marked according to whether they are "present" (ie appear in the working set document).

Furthermore extractions and insertions are specified in post insertion coordinates. This allows a single move operation to extract the same characters that it inserted. Therefore a single operation can represent the combined effect of many operations on the effects document. We see therefore that this representation is necessary to implement lossless merging that remains compatible with IT/ET.

We can distinguish two types of log compression according to whether it respects equivalence on the effects document or only on the working set document. The latter leads to far better compression because characters that are inserted then deleted again have no effect on the working set document and therefore can be ignored. One can

imagine examples where many days of editing is performed on a given document and overall only a small number of large chunks of text are added or removed from the working set document. However the effects document will tend to record all the work that was subsequently deleted. It is not uncommon for this to far exceed the size of the end result.

Compression with respect to the working set document would seem preferable. However, it raises the question of how to deal with the PtoQ maps stored on a client that will compress the log before checking in changes to a repository.

One option is for the client to undo all the operations in the log, compress the log, then reapply the operations. This will give the required changes to the PtoQ maps.

Alternatively, it is conceivable that during the compression of the log adjustments to the local PtoQ maps are made as required.

Holding areas

Consider that we only support move operations, ie there is no explicit support for insert and delete operations. Then it is necessary to introduce *holding areas* to allow for the effect of insertion and deletion.

A holding area is a document with an OID and persists that is used to allow for creation of destruction of matter as far as the "real" documents are concerned.

This simplifies IT/ET because there is only one type of operation (ie "move"). However the use of holding areas raises the question of how log compression will be achieved. Somehow the identity of the holding areas must be regarded as not significant, providing opportunities to allow for proper log compression.

The use of holding areas makes log compression rather difficult. For example, consider that three sites concurrently perform the following operations.

O1 : Delete character : move from S1 to holding area H1
O2 : Delete character : move from S1 to holding area H2
O3 : Move character : move from S1 to document S2

Suppose O1 < O2 < O3. Let site 1 store [O1 O2' O3']. Then O2' will move the character from H1 to H2, and O3' will move the character from H2 to S2. Note therefore that O3' looks like a pure insert operation even though it really represents a move.

This leads to a rather nasty observation: Given a log [O1, ... , On] we will have operations that refer to holding areas by their identity and we can't assume any of the following

1. a move into a holding area doesn't necessarily represents a real delete, because it may be dominated by a move that brings the deleted characters back to life.

2. A move from a holding area into a document doesn't necessarily represent a pure insert.

In fact, we can't really assume much at all unless we process the entire log. This makes it difficult to see how we could implement some form of incremental log compression algorithm.

Proposal

Consider that we don't employ holding areas identified with OIDs. Instead operations must directly represent pure deletes and inserts (in addition to moves). One immediate benefit is that we avoid the overhead of lots of tiny, one character holding areas generated while editing text. This could be significant.

When a delete operation is IT'd with a move operation we assume that the delete always dominates the move. ie all sites agree that the array elements have been permanently deleted.

In the analysis below we assume single character insert/delete and move operations.

Effects document

The effects document is a hypothetical document in which insert and move operations always insert a single character at the designated q position in the destination document. Each character in the effects document carries a flag to mark whether it is currently present.

IT and ET

Insert operations can never be disabled under IT. There is no tracking rule. The inserted character will always be marked as present. The insertion q-position must be shifted as required but will never track into a different document.

Delete operations should track where the character to be deleted is currently present, if it is present. If not present then it should track to the last location before its deletion.

Insert and Insert : Easy

Insert and Move :