1U−6

リアルタイム型分散エディタにおけるバッファ間の一貫性制御方式

大野隆一　相田仁　齊藤忠夫
東京大学　工学部

1　はじめに

複数の人が同時に一つの文書の編集を行なうことができるリアルタイム型分散エディタにおいては，それぞれのノードでバッファを維持しているために，それらのバッファの内容の一貫性を維持しつつ，バッファへの操作の応答性を良くする必要がある．

本稿では，このようなバッファ間のConcurrency controlをとる手法を提案し，その得失を論じる．

2　分散エディタにおけるConcurrency control[1]

リアルタイム型分散エディタでは通常それぞれのノード内で編集対象の文書をバッファにキャッシュすることにより応答性を良くしている．この場合，バッファに対する変更はグループ内の全てのノードのバッファに対して行われる必要があり，幾つかのノードで同時に書き込みが起こった場合などにも全てのバッファの内容が同一に保たれている必要がある．バッファへの変更操作は通常自バッファに対しては即座に行なわれるが，他ノード上のバッファに対しては伝送遅延を伴う．従って，他のノードからの変更要求は順番通りくるとは限らず，全てのノード上のバッファの一貫性を取る何らかの機構が必要となる．

分散エディタにおけるバッファのconcurrency controlに用いられる手法としては，(a)データへのアクセス権を得てから操作を行なうことでアクセス競合を防ぐもの，(b)操作は即座に行なえるが，競合がある場合のみ何らかの方法でアクセス競合を解決する必要のあるものの二つがある．(b)の手法におけるアクセス解決手段としては，(1)ユーザの介入による方法(Dependency-detection)，(2)undoによる方法(Reversible Execution)，(3)他のノードからの処理数を比較することによる方法(Operation transformation)などがある．

Floor Controlやlockingのような(a)の手法はそのデータへのアクセス競合がおこらないことは保証するが，応答性は良くない．それに対して(b)の手法では変更操作を並行して行なえるため応答性が向上する．ここでの問題は，(3)の方法ではglobal orderingの順番通り変更操作が行なわれるため変更操作が待つ（queueに入る）必要があり，(2)の方法では変更操作が待つ必要がない代わりにundoされる可能性があることである．

A Concurrency Control Scheme in Real-time Distributed Editors

Ryuichi OHNO, Hitoshi AIDA, Tadao SAITO

University of Tokyo

3　no-wait execution

3．1　概要

ここでは基本的にundoの必要がなく，かつglobal orderingに関係なく変更操作を行なえるような方式について考える．つまり，他のノードからの変更要求は原則的に待たされることなく（但し，まだ到着していない挿入部に対する消去は待つ必要があるが），かつ，正しい位置に対して変更がなされる．これを実現するために，本方式では一度に挿入された部分を一つの挿入バッファとし，その挿入バッファ及び挿入バッファ内オフセットを指定することで位置の指定を行なう．さらに，削除の際にも情報を残すことで他の操作に影響されずにバッファ内での位置を指定することが可能となる．

3．2　バッファ構造

バッファは最初にファイルから読み込まれる初期バッファと，挿入操作により付け加えられる挿入バッファからなる．それぞれのバッファにはそのバッファを識別するための識別子，及び，そのバッファ中での消去されている位置の情報からなる．（消去の際にも，バッファが消されることはなく，消去の印が付けられるだけである．）

3．3　操作

・挿入操作
挿入操作により新たな挿入バッファが作られる．

・消去操作
消去操作により初期バッファまたは
挿入バッファ中の一部分に消去の印が
付けられる．

3．4　Packet Format

変更要求の際に他のノードに送られるパケットフォーマットを図１に示す．

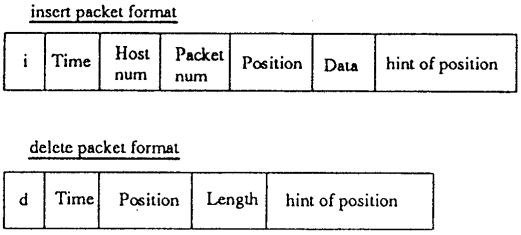

図１　　Packet Format

図１でTimeはこのパケットが送り出された時間を，Hostnumは送り出したホスト識別子を，Packetnumは同じホストから同時に送り出されたpacketがあった場合にそれぞれを識別するための番号を示す．以上３つの情報が先に述べた挿入バッファを識別するための識別情報になる．

Positionは挿入バッファを識別するための識別情報と挿入バッファ内でのoffsetからなる．hint of positionは検索を容易にするために挿入（削除）位置のだいたいの位置を知らせる．例えば，定期的にバッファ全体に行番号を振り，その（だいたい正しいであろう）行番号をヒントとして目的位置を探すのに用いる．

3．5　操作例

図２に挿入及び削除の例を示す．

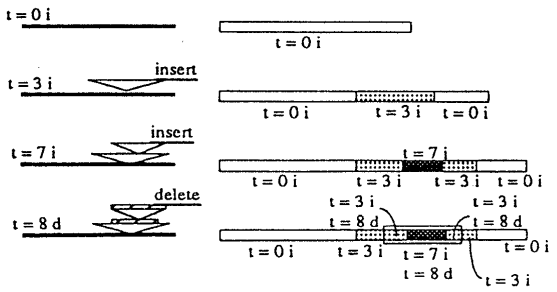

図２　バッファへの挿入及び削除

まず，t=3において挿入が行なわれる．この場合，t=0の挿入バッファを識別するための識別情報とその挿入バッファ内での位置が決まれば挿入位置が決まる．t=7においても同様にt=3の挿入バッファの識別情報とオフセットにより挿入位置が決る．t=8では削除を行なってる．この場合t=3の挿入バッファの識別情報とオフセットと消去する長さ，及びt=7の挿入バッファの識別情報により消去される位置が決まる．

3．6　得失

このような方法の利点はglobal orderingの順序通りに変更操作を行なう必要がないく他のノードからの変更要求は原則的に待たされないために並列度が上がること，及び，正しい位置に対して常に変更が行なわれるためにundoの必要がないことである．

この性質は特に遠隔地間で低速回線を介して分散エディタを用いているためにあるノードからの変更要求バケットがなかなか他のノードに伝わらない場合などに有効であろう．

この方法の欠点はバッファの構造が複雑になるために応答性が悪くなることと，単体のマシン上でのエディタの単なる改造では済まないことである．最初の点はCPUのスピードがある程度高速であることを仮定すれば大きな問題にはならないと思われる．

また，このような方法の用途としては複数の人がある程度他の人の作業も意識しながら一つの文書を短い時間でざっとまとめてしまう場合などが考えられる．きちんとまとめる場合にはさらにfloor controlなどの別モードを用いる必要があるであろう．

4．おわりに

本稿では，分散エディタにおけるConcurrency controlの一手法を提案した．今後はさらにシステム全体の機能についても検討を加え，実装を進めていきたいと考えている．

参考文献

[1] Ellis,C.A., and Gibbs, S.J. Concurrency control in groupware systems. In Proceedings of the ACM SIGMOD '89 Conference on the Management of Data