リンク構造ファイルシステムにおけるファイルバッファおよび履歴読出しのオーバーヘッドの評価

12G470　田中 孝典 （最所研究室）

本論文では，挿入・削除操作におけるディスク I/O およびファイルバッファの効果および過去のバージョンを読み出すオーバーヘッドについて述べる．

1　はじめに

当研究室では，ファイルシを可変長ブロックの集合として扱うリンク構造ファイルシステムの開発を行ってきた [1]．可変長ブロック同士のリンクをつなぎかえることによって指定位置に可変長ブロックを直接挿入したり，指定位置の可変長ブロックを削除することができる．さらに，ファイルに変更が行われたときにファイル変更前のリンクの状態を記録することでファイルシステムレベルで履歴をもつことができる．

本研究では，挿入・削除操作で発生するディスクの読み書き回数の評価に加え，ファイルバッファの実装およびその効果の確認を行った．さらに，履歴から過去の状態を読み出す機能を実装し，ファイルから過去の状態を読み出す場合に発生するオーバーヘッドの評価を行う．

2　履歴読み出し

リンク構造ファイルシステムにおけるファイル構造を図 1 に示す．ファイルに対して挿入・削除操作を行ったとき，各ブロックからつながれているリンクがつなぎ変えられる．同時に過去の状態を記録すために理例管理用ブロック LHist を作成することで，操作前の状態を復元することが可能となっている．シーケンシャルナンバ（以降連番）を持っている．連番を指定することによりファイルの過去の状態を読み出す．ブロックから複数の LHist が存在する場合，指定した連番より大きい LHist のうちもっとも小さい LHist のリンクをたどることで指定した状態を読み出す．例えば図 1 で LHist4 を指定した場合，”Dummy Block → Block1 → LHist5 → Block3 → Block2 → LHist4 → Dummy Block ”の順番でリンクをたどり履歴を読み出す．

3　ファイルバッファ

アロケーションブロック (AB)[2] のバッファリングを行う．バッファを管理するために図 2 に示す双方向循環リストを構成する．バッファは AB の読み込みのたびに LRU で入れ替えを行う．リストの先頭から最終アクセスからの期間が短い順に配置する．図 3 に示すようにディスクから AB を読み出す必要があり，バッファに空きがない場合はリストの末尾 BH の AB を上書きし BH を先頭に移動することで読み込んだ AB を格納する．読み込む AB がバッファ内に存在する場合は，ディスクから読み込みを行わずに対象の AB を指している BH をリストの先頭に移動する．

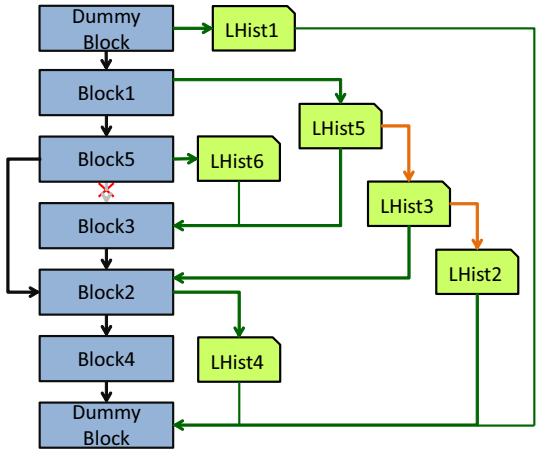

図 1: リンク構造ファイルシステムのファイル構造

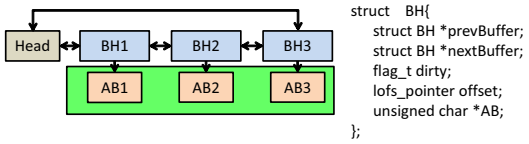

図 2: バッファ管理用循環双方向リスト

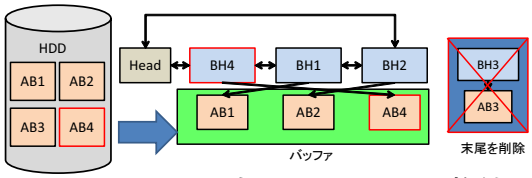

図 3: ディスク中 AB のバッファ格納

4　評価

4.1　挿入削除による読み書き回数

連続で 5,000 個のブロックを挿入したのち，ランダムに 500 個のブロックを削除を行ったときの各操作に

要する読み書き回数の平均を調べた．はじめに行う挿入位置として先頭，末尾，ランダムの 3 つの場合について調べた．表 1 に結果を示す．別の AB 上のブロックからのリンクの確率が高いランダム挿入の読み込み回数が他より約 1 回多くなった．一方，先頭挿入の書き込みは，ブロック間リンクと LHist とのリンクを同時に書き込めるため他より約 1 回少なくなった．

表 1: 読み書き回数

挿入位置	ディスク (R)	ディスク (W)
ランダム	5.03	3.18
先頭	3.96	2.19
末尾	3.98	3.08

4.2 挿入削除操作のバッファの効果

バッファリングを行ったときの読み込みに対する影響を調べた．4.1 節と同じ条件のファイルをバッファ数を変更して作成した．なおファイルの AB 数は最終的に約 110 個であった．図 4 に実験結果を示す．先頭および末尾に連続で挿入する場合は，参照する AB が集中するため少ないバッファでもディスクからの読み込みが急速に減少している．一方ランダム位置に挿入する場合は，別の AB 上のブロックからのリンクの確率が高くかつ分散しているためバッファ数が増えるに従い徐々に減少した．

4.3 履歴読み出しのオーバーヘッド

4.1 節で作成したファイルについて最新状態と履歴を読み出すときに次のブロックを読むためにかかったリンクの読み込み回数しらべた．図 5 に実験結果を示す．横軸は読み出す履歴の連番を示す．先頭に連続して挿入したファイルでは連番が小さいときがもっとも読み込み回数が多い結果になった．図からはみ出ては

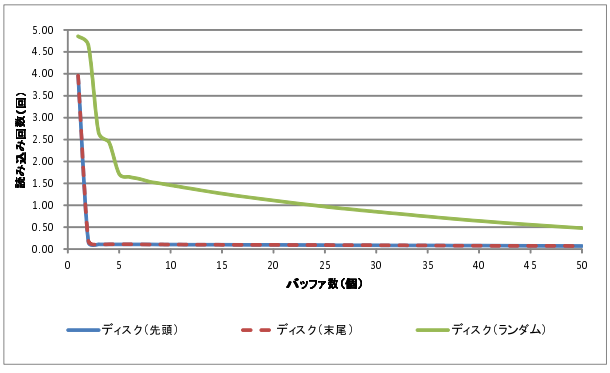

図 4: バッファの数による読み込み回数の変化

いるが，連番が 55 のとき平均 87.32 回になっていた．先頭挿入するとファイルの先頭を示す Dummy Block から挿入回数分の LHist が作られるため，連番の小さい履歴を読み出す場合挿入回数に比例した LHist をたどることになったためである．末尾挿入ファイルは挿入操作によって同じブロックから複数の LHist を作られることがないため，次ブロックを読み出しに 1 回の LHist へのリンク読み出しと条件に一致するブロックのリンク読み出しの合計 2 回で次ブロックを読み出すことができるため，古い履歴であっても新し履歴と変わらない回数で読み出すことができる．一方ランダム位置の場合は，LHist のリンクの連鎖が分散されるため先頭挿入ほど集中はしないが，連番が小さいものほど読み込み回数が増えた．

5 まとめ

本研究では，履歴読み出し機能およびファイルバッファの実装を行った．さらに，ディスクの読み書き回数でリンク構造ファイルシステムの挿入・削除操作および履歴読み出しのオーバーヘッドの評価を行った．その結果，バッファを用いることでディスク読み込み回数を減らすことができることを確認した．履歴読み出しに関しては先頭挿入については履歴の管理方法を改善する必要があることが分かった．　今後の課題として，ファイルの種類によって必要なバッファの数が異なるため，適切な数のバッファ割り当て機構の検討を行う必要がある．また，実時間による測定を行いディスク中のブロックの配置によって受ける影響を調べる必要がある．

参考文献

[1] 津紀孝, 最所圭三, “行指向ファイルシステムについて”, 平成 18 年度電気関係学会四国支部連合大会論文集, 15-37, p.242, 2006

[2] 小林憲弘, “リンク構造ファイルシステムの効率化および従来のファイルシステムとの互換操作の設計・実装”, 香川大学大学院工学研究科,　修士論文,2012

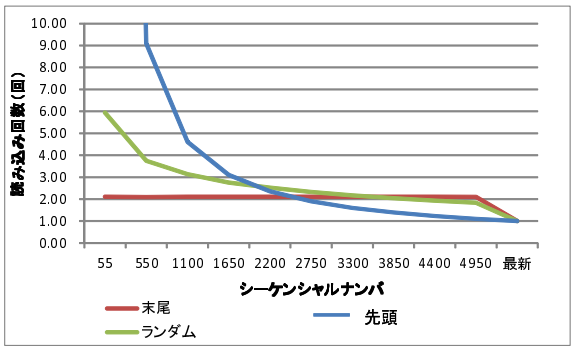

図 5: 履歴読み出しのオーバーヘッド