リンク構造ファイルシステムにおける空き領域管理機能の設計と評価 － 不要履歴の回収および断片化の解消－

10G217　大橋 洸一　（最所研究室）

ファイル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」に、不要領域の回収、およびに回収した空き領域の断片化を解消するためのデフラグ機能の設計について述べる。

1　はじめに

当研究室では、リンク構造ファイルシステムの開発を行っている [1]。このシステムはファイルを従来の構造のないバイト集合として扱うファイルシステムとは異なり、可変長ブロックの集合として扱う。可変長ブロック同士をリンクで繋ぐことにより、従来のファイルシステムでは実現できなかった指定位置へのデータ挿入や削除を行うことができる。さらに、ファイル操作で変更されるリンクの状態を LinkHistory(以下 LHist) と呼ぶ構造体を用いて記憶する事により、ファイルシステムレベルでの履歴機能を実現している。またこれらのブロックや履歴のうち、ユーザーやシステムが指定した不要な領域を回収する機能を実装している [2]。

しかし、可変長ブロックを用いているため断片化が発生するが、従来の空き領域回収ではそれに対応していなかった。また、リンク構造ファイルシステムにブロックや LHist を一箇所に纏めることにより、より効率的な領域管理を行うための構造体である AllocationBlock の実装が行われたため、不要領域の回収機能をこれに対応する必要が生じた。

そこで本研究では、不要履歴の回収機能を AllocationBlock に対応できるように改修し、さらに断片化を解消するためのデフラグ機能の設計を行った。また、設計したアルゴリズムを用いた時のディスクアクセス回数を基にした性能評価を行った。

2　不要履歴の削除機能

本研究では、ユーザーやシステムが指定した履歴の削除を行う機能の再設計を行った。この機能では、指定した履歴に繋がれているリンクを削除、または変更することにより、対象の履歴が他のブロックのヘッダ (以下 DBH) から参照出来ないようにすることで履歴の削除を行う。この際空き領域としての回収は行わず、後述するデフラグ機能により削除領域の回収を行う。この点が従来のリンク構造ファイルシステムの空き領域回収機能と異なる点である。

また、履歴の回収は前後の履歴の相互関係が損なわれないように行う必要がある。そのためには、以下のいずれかの条件を満たさなければならない。

- 前の LHist が存在しない (先頭の LHist である)
- DBH から繋がれているリンクが前の LHist と同じである
- DBH へと繋いでいるリンクが前の LHist と同じである

これらの条件を満たす例を図 1 に示す。図の状態では LHist-A、LHist-B、LHist-C の順に LHist が作られている。LHist-A は最も古く作られた先頭の LHist であり、前の LHist が存在しないためそのまま削除が可能である。また LHist-B は、前の履歴である LHist-A とリンク元の DBH である DBH-X が同じであるため同様に削除が可能である。しかし LHist-C は、前の履歴である LHist-B と同じ DBH へのリンクがないため、そのまま削除することはできない。

削除が不可能であると判断された LHist-C はリンクの繋ぎ変えを行わず、隠蔽フラグを付加する。この隠蔽フラグが付加された LHist はユーザーからは不可視になり、直前の LHist-B の削除が行われた場合に実際に削除することができる。これにより、履歴の相互関係を損なうことなく履歴の削除が可能になる。

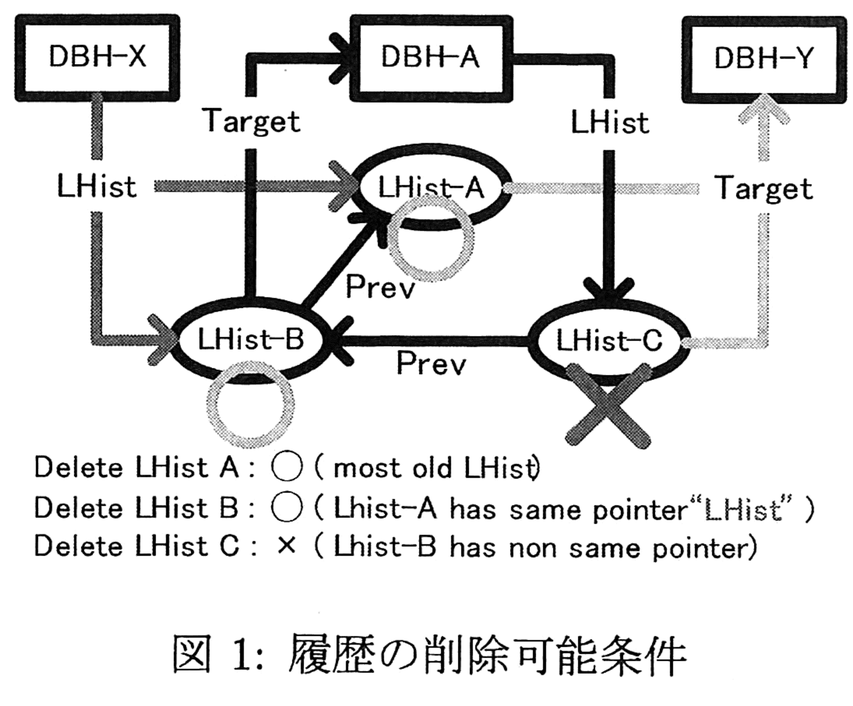

図 1: 履歴の削除可能条件

3　デフラグ機能

本研究ではメモリを対象としたガベージコレクションであるコピー GC をベースにして、デフラグ機能の設計を行った [3]。コピー GC とは領域のうち到達可

能な削除されていない領域を新しく用意した領域にコピーし、その後コピー元の領域を回収する。

この手法を用いたデフラグ機能の概要を図 2 に示す。図の (1) の状態からデフラグを行う場合、まず (2) のようにDBH とブロックの実データ部分 (以下 DBLK) のうち、ファイルから辿ることのできるブロックである DBH1 と DBLK1、DBH3 と DBLK3 を新しく用意した領域にコピーする。その後 (3) のように、辿ることのできる履歴である LHist1 と LHist4 を新しい領域にコピーする。最後に (4) のように、コピー先の領域をコピー元に置き換え、コピー元の領域を空き領域を管理するための構造体 (FSM) に繋ぐ。この際不要な領域である DBH2 と DBLK2、LHist2 と LHist3 は新しい領域へとコピーされず、その分の空き領域が新しい領域で纏められる。これにより空き領域の断片化が解消される。

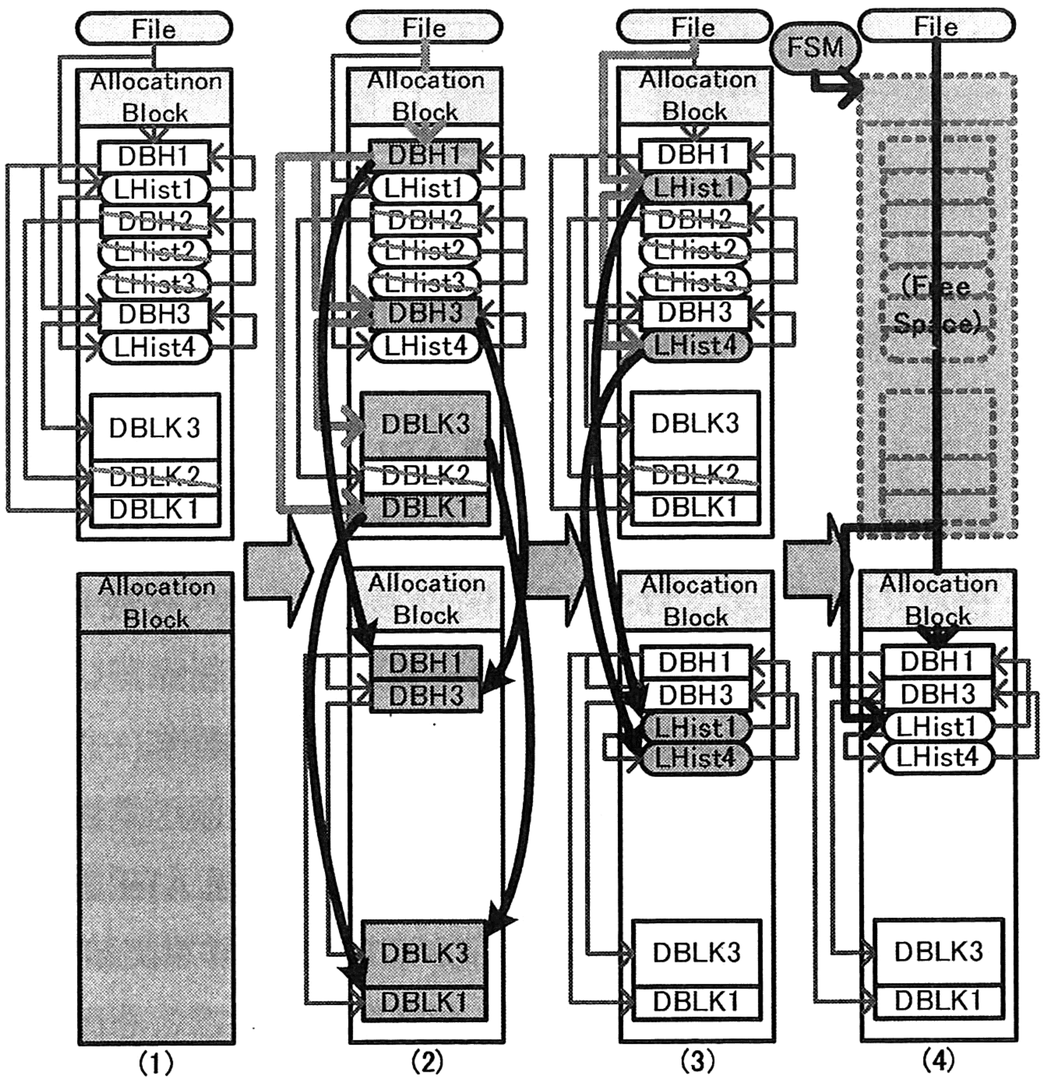

図 2: デフラグの概要

4 評価

これらの設計した機能に対して、ソースコードに基づく性能評価を行った。設計したアルゴリズムを用いて、それぞれの処理の工程数からデフラグに必要な処理時間やメモリの容量のシミュレーションを行った。

一度に読み込む DBH の数 (readB) と一度に読み込む LHist の数 (readL)、全体の DBH の数 (B)、削除操作による LHist の数 (delL) を変化させた場合の、デフラグの処理時間のシミュレーション結果を図 3 に示す。この結果から readB、並びに readL は一定以上の個数があれば処理時間は短くなり、また DBH の数の増加量より、削除したまま履歴だけを残した DBH の増加量のほうが、デフラグの処理時間に与える影響が大きいことが読み取る事ができる。

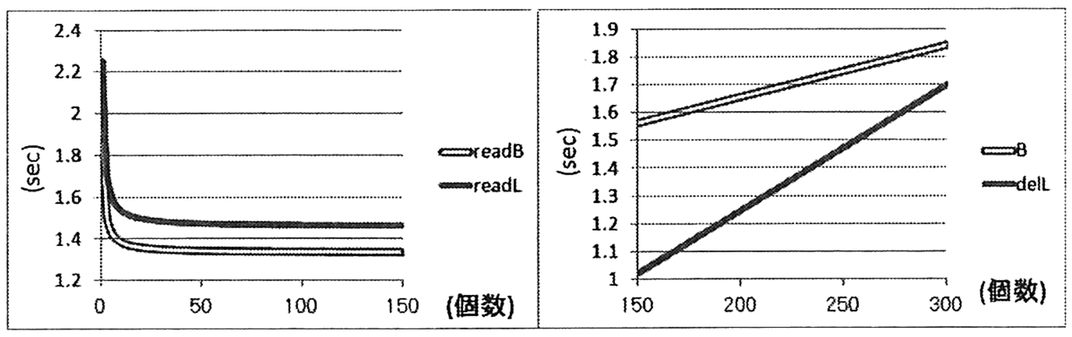

図 3: 処理時間の評価

同様に、使用するメモリ容量のシミュレーション結果を図 4 に示す。readB と readL の増減によるメモリ使用量の変化を比較すると、readB の方が readL に比べて初期値こそ大きいものの、readL の方がメモリ使用量の増加が激しい事が図から読み取る事ができる。

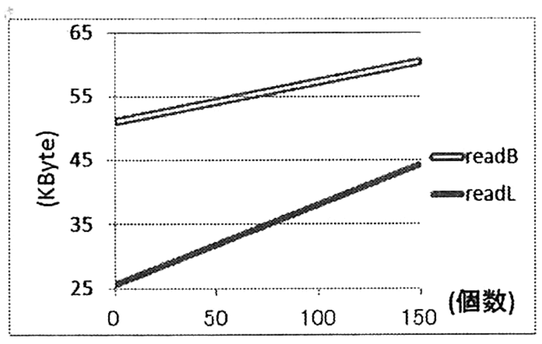

図 4: メモリ使用量の評価

5 まとめ

以上、リンク構造ファイルシステムの空き領域回収機能とデフラグ機能の設計と評価について述べた。しかし、これらは現在設計段階の機能であり、実装には至らなかった。そのため、今後はこれらの機能の実装を行うとともに、処理時間を短縮するためより効率的な処理方法と、デフラグを自動で行う機能の実現が必要がある。

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” 電気関係学会 四国支部連合大会, 15-37, p242, 2006

[2] 大橋洸一, 最所圭三, “リンク構造ファイルシステムにおけるガベージコレクションについて” 平成 22 年度 電気関係学会 四国支部連合大会,17-26, p298, 2010

[3] Fenichel, R.R, ”A LISP garbage-collector for virtual-memory computer systems”, (1969), Communications of the ACM