# 情報処理学会第40回全国大会論文集

December, 1989



ICOT

Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan

(03) 456-3191～5
Telex ICOT J32964

**Institute for New Generation Computer Technology**

| | | |
|---|---|---|
| TM-0829 | SIMPOSボリューム・コンパクション | 加藤龍夫,<br>佐藤　忠 |
| TM-0832 | SIMPOSライブラリアンのカスタマイズ機能 | 児玉泰子,<br>藤本誠司 |
| TM-0834 | 論理型プログラミング環境における文法開発支援ツール | 木下　聰,<br>佐野　洋 |
| TM-0836 | グラフ理論による制約解析および手順生成 | 永井保夫 |
| TM-0837 | 依存構造解析のためのSAXのデバック環境 | 福本文代 |
| TM-0838 | 知識検証支援システム | 戸辺啓子 |
| TM-0840 | 自然言語実験支援環境LINGULST | 小野寺浩 |
| TM-0842 | 並列プログラムの動作評価に関する一検討 | 安藤津芳 |
| TM-0843 | ペトリネットによる並列プログラムの動作解析に関する一考察 | 前田賢一 |
| TM-0844 | 極小限定の確率的意味について | 佐藤　健 |
| TM-0845 | 並列制約充足アルゴリズムとそのKLIによる実現 | 杉本　勉 |
| TM-0846 | 並列推論マシンPIM/iの概要 | 大原輝彦 |
| TM-0847 | 並列推論マシンPIM/iの要素プロセッサのアーキテクチャ | 武田浩一 |
| TM-0848 | 並列階層キャッシュメモリの性能評価 | 村谷博文 |
| TM-0849 | マイクロコード・チェッカ「マイクロリント」の開発 | 宮内信仁 |
| TM-0850 | PIM/m要素プロセッサのアーキテクチャ | 中島　浩 |
| TM-0851 | 知識ベース指向並列処理システム | 横田治夫 |
| TM-0852 | 新聞社説記事における照応現象 | 柴山昌宏 |

# ＳＩＭＰＯＳボリューム・コンパクション

加藤龍夫，佐藤　忠　三菱電機東部コンピュータシステム株式会社

## 概　要

SIMPOS( Sequential Inference Machine Programing and Operating System　逐次型推論マシン・プログラミング＆オペレーティングシステム )のボリューム・コンパクション機能について、その設計方針と効果について発表する。

SIMPOSファイル・システムは、木構造を持つUnix流のデータ分散型ファイル・システムである。このようなファイル・システムの速度性能は、データ分散によるディスク・ヘッド・シークに大きく左右される。我々はこの点を重視し、まずSIMPOS 4版においてデータ分散を部分的に抑制する方式を採用した。次にデータ分散抑制の方針をさらに進め、各ディレクトリがディスク上に占める記憶領域を、ファイル本体も含めて再配置するのが当ボリューム・コンパクションの機能である。従って本機能の使用によりファイル・アクセスが高速化される。

## １．　はじめに

SIMPOSファイル・システムは、データ分散型システムとして開発された。即ちユーザ・ファイル、ディレクトリ・ファイルの全てが、時間の経過とともにディスク上に拡散してゆく。この拡散の状況はヘッド・シークに関係するため、ディレクトリ・ツリーをたどり、目的のファイルにアクセスするまでの時間に大きく影響を与える。

以下の議論のために、ここでファイル・システムのあらましを説明する。

ファイル・システムにとってディスクの最小管理区分はデータ・ブロック( 4KB ) である。以下ではデータ・ブロックを単にブロックと呼ぶ。ボリューム初期化時には、ファイル・システムの最小構成であるシステム核が生成される。ここにはボリューム・ラベルをはじめ、すべてのファイルを管理するファイル・ファイルの最小形、解放されたブロックを管理するフリー・ブロック・マップ・ファイル、解放されたファイル・ファイル・レコードを管理するフリーi _noマップ・ファイル、ルート・ディレクトリ・ファイルの最小形等が配置される。

ファイル・ファイルはレコード形式のファイルで、ファイルの生成に伴い必ず１レコードが確保される。最初のレコード前半はファイル属性の記憶に用いられ、後半はそのファイルが所有するブロック番号の記憶に用いられる。最初のレコードにブロックがエントリしきれない場合は、同じファイル・ファイル内にブロック記憶専用のマップ・レコードを用意する。

ファイル・ファイル自身もこの形式に従うファイルの一種であり、ブロックはディスクの至るところに分散して存在する。しかしそのマップは、システム核内に固定してとられ、この点他のファイルとは異なる。

ディレクトリも、上に述べた形式に従って管理されるファイルの一種である。そのデータはエントリ名等の属性を記憶するディレクトリ・レコードの集合であり、各レコードはファイル・ファイル・レコードをレコード番号でポイントしている。

この様子を図１に示した。ユーザが >sys>user>me>fileというパス名のファイルに対してオープン命令を出すと、システムは左上のルート・ディレクトリから探索を開始し、矢印に従って最後にファイルのデータに達する。

すなわち、ディスク・ヘッドはファイル・アクセス時に次のものを求めて移動する。

a) ディレクトリのブロック（複数の場合あり）

b) ファイル・ファイル・レコード（複数の場合あり）

a),b) の繰り返しの後、

c) ファイルのブロック（複数の場合あり）

これらのうちa)とc)はブロック間隔、すなわちブロック番号の差を縮小する工夫をすれば同様に高速化できる。

a),b) 間のシークを減少させるには、ディレクトリからポイントされたファイル・ファイル・レコードを、ディレクトリ・データに近接して置けば良い。

b),c) 間のシークを減少させるには、ファイルのブロックを今述べたファイル・ファイル・レコードに近接して置けば良い。

またb)でファイルが大きい場合には、１ファイルの所有するファイル・ファイル・レコード間でのシークも問題となる。これはフリー・レコード番号の取り扱いで、a),c) のブロック番号の場合と同様に解決できる。

以上の問題解決には、番号の連続するブロックがディスク上で同一、又は近隣のシリンダに置かれていることが必要である。

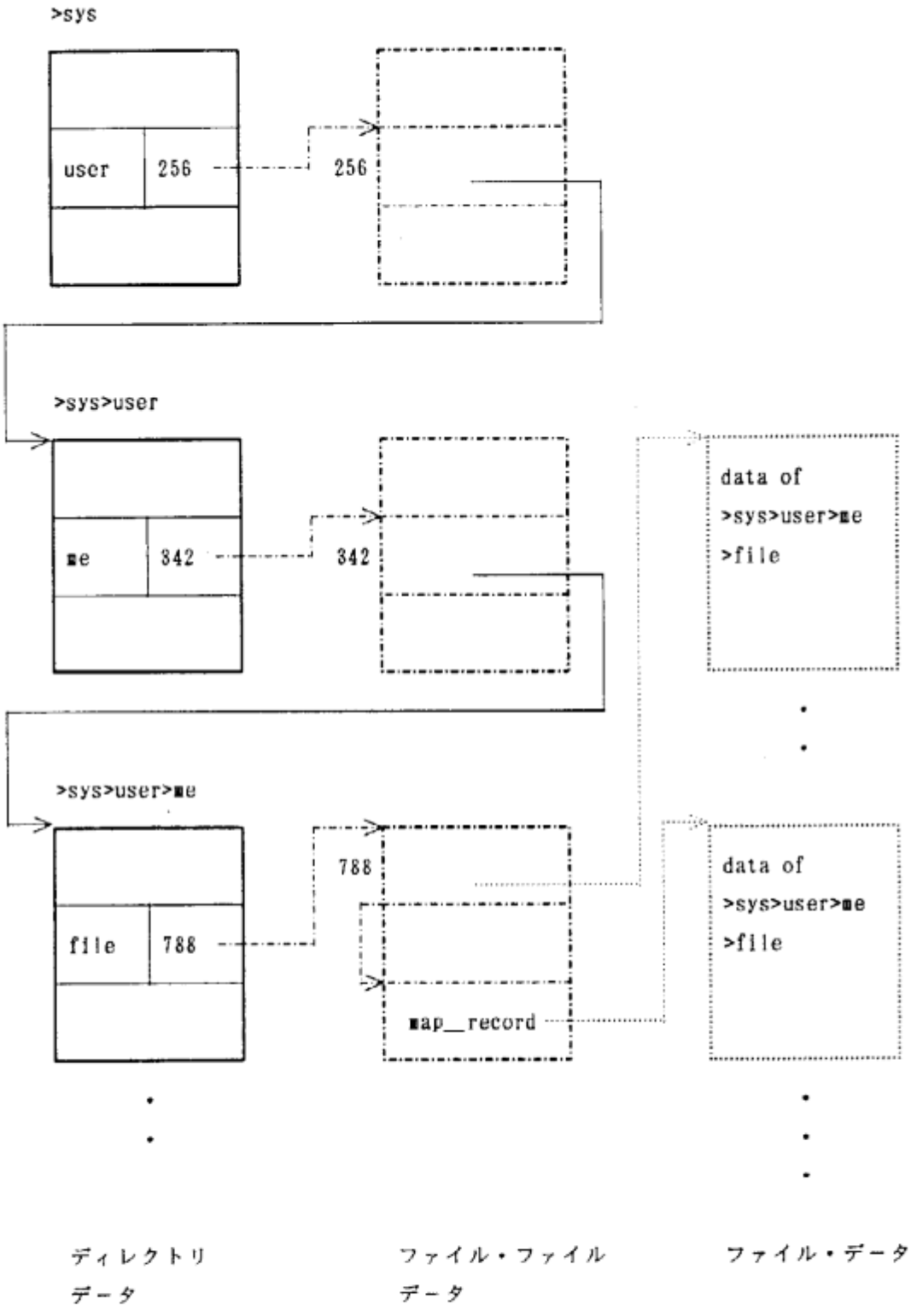



図1　ファイル・システムの構造

## 2.　ファイル・システム稼動時の改善（4版）

1．のa),b),c)の各々、すなわちブロック間隔、ファイル・ファイル・レコード間隔については、ファイル・システムの稼動時に改善することができた。

一度使用されて解放されたブロック、ファイル・ファイル・レコードは、我々がマップ・ファイルと呼んでいるビット・マップ・ファイルに登録され、新ファイルの生成、ファイルの拡大時に探索、獲得される。

ファイルのブロック間隔、ファイル・ファイル・レコード間隔増加抑制の解決策はいずれも同様で、

解放資源の探索は、そのファイルが既に持つ記憶資源（番号）の近傍を優先的に行う

というものである。

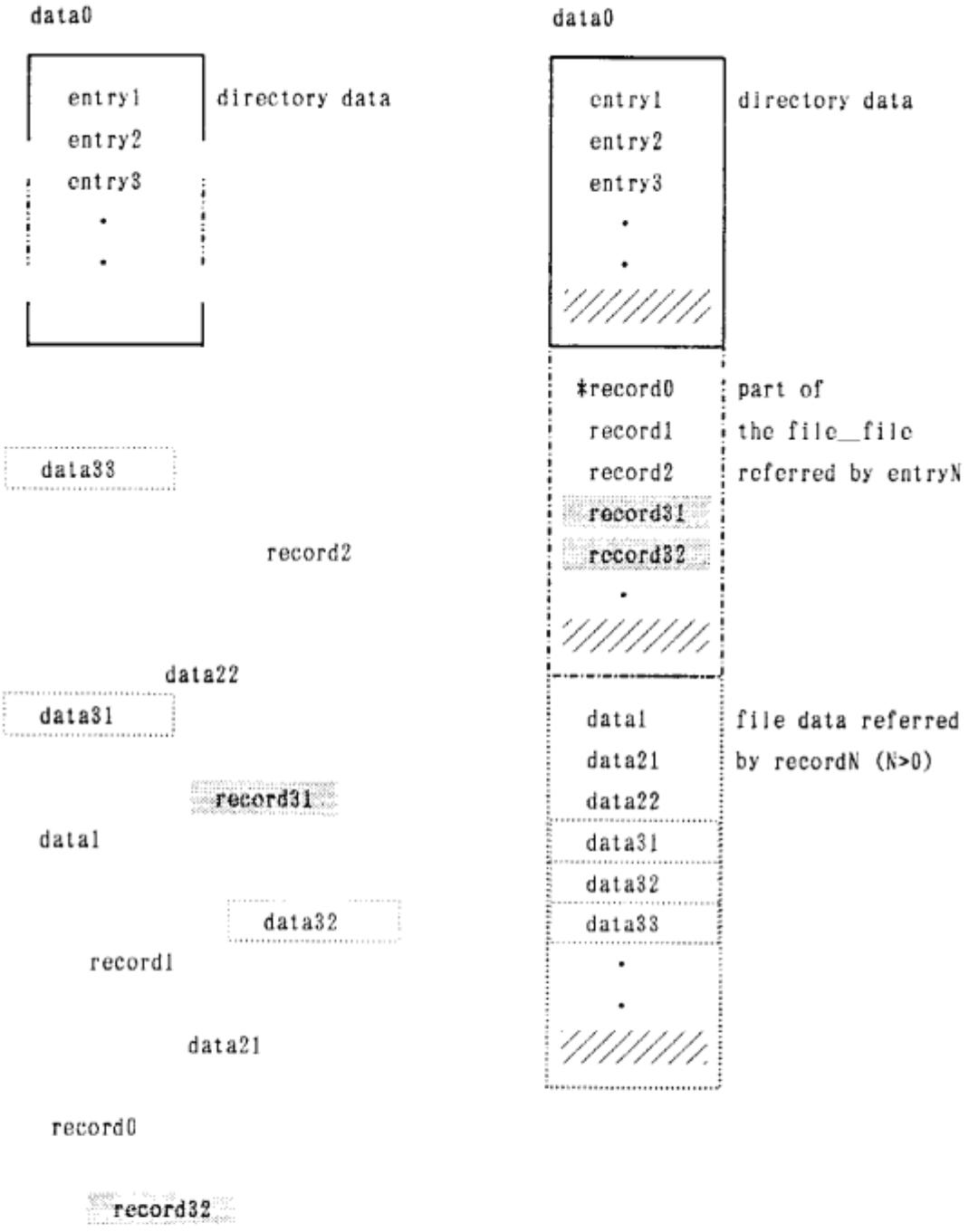
## 3.　ボリューム・コンパクションによる改善（5版）

2．の改善の他に、1．a),b) の間隔、b),c) の間隔をつめることが解決すべき問題として残っている。ファイルの生成時に、親ディレクトリ・ブロックの近くにあるファイル・ファイル・ブロックを探し、その内部を中心としてフリー・レコードを探索したり、ブロック・マップのあるファイル・ファイル・ブロックを検索し、そこからフリー・ブロックの探索を行うのは、ファイル・システム稼動時にはオーバー・ヘッドとなる。

そこでこの問題を機能呼び出しとして解決したものがボリューム・コンパクションである。

まず以下の説明のために、ディレクトリ・セグメントを定義する。ディレクトリ・セグメントは、ディレクトリのすべてのエントリに関する1．のa),b),c)、すなわちディレクトリのブロック、そこからポイントされるファイル・ファイル・レコード、エントリ・ファイルのブロックの集合である。

コンパクションは、分散したディレクトリ・セグメントの要素を上に述べた順にブロック番号の連続した領域に移動する。コンパクトになったディレクトリ・セグメントをコンパクション・セグメントと呼ぶことにすると、コンパクション・セグメントはディレクトリ・ツリーに従って順次配置される。ディレクトリ・セグメントの形成と、コンパクション・セグメントの配置を図2、3に示した。




注）***N(N- 整数）で、ディレクトリ・エントリとそれに対応するディレクトリ・セグメント類を識別する。N = 0 は親ディレクトリの属性を示す。

図２　コンパクション・セグメントの形成

図２の斜線部は空き領域を示している。ファイル・ファイル部、ブロック部に余裕を持たせる理由は次の２つである。

① コンパクション後に生成したエントリのファイル・ファイル・レコード、ブロックが親ディレクトリの遠方にとられて、サービスに偏りが生じるのをある期間防ぐ。
② コンパクション後に生成したエントリにより拡張される親ディレクトリのマップ・レコード、ブロックが、遠方に割り当てられるのをある期間防ぐ。

このコンパクション・セグメント内に空き領域を持たせるという方式に意味を持たせるため、コンパクション機能の追加に伴いファイル本体部で次の改修を行った。

ファイルを生成する時は、その親ディレクトリを起点として空きファイル・ファイル・レコード、フリー・ブロックを探索する。

これは①を保証するものである。②については、２．の４版での改修が保証している。

また図３は、コンパクション・セグメントをディレクトリ・ツリーを左からなぞるように配置することを示している。新しいエントリが必要とする記憶資源は、親ディレクトリのコンパクション・セグメント内に確保できなくなった場合でも、近辺の至る所にあるコンパクション・セグメントのいずれかに取られることになる。この効果は、コンパクションによって発生した空き領域がなくなるまでの間持続する。

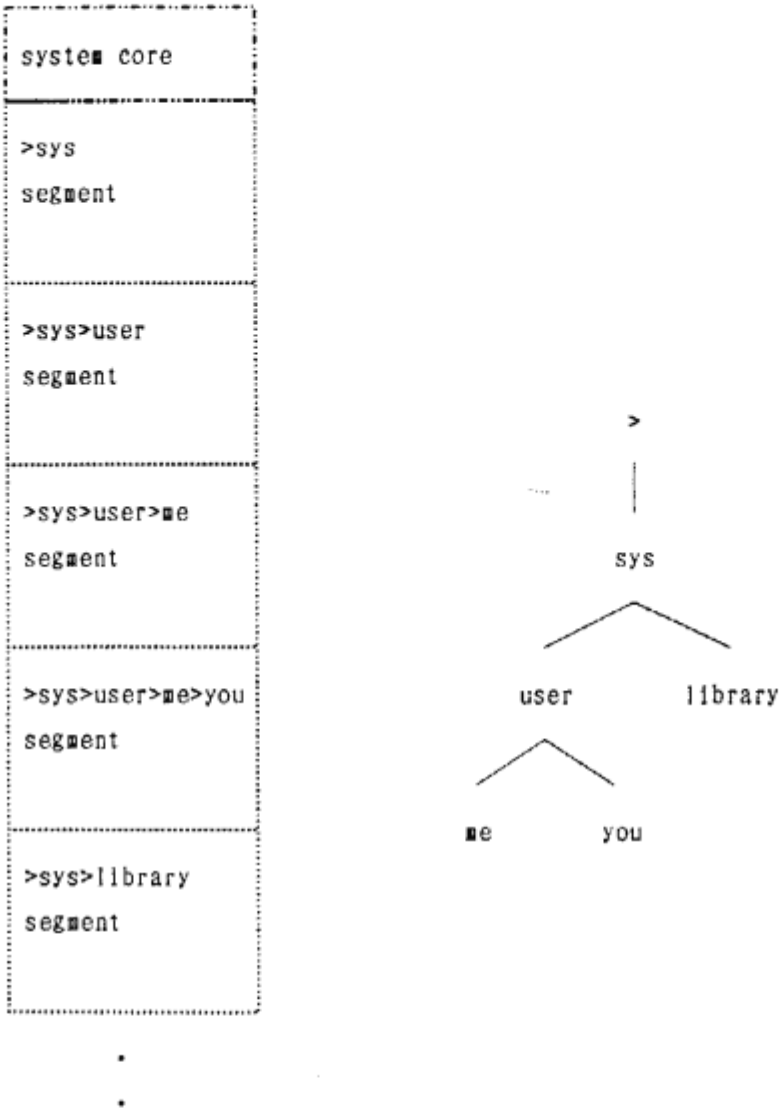



図3　コンパクション・セグメントの配置

## 4．終わりに

このように、SIMPOSのボリューム・コンパクションは、ヘッド・シークの減少によるアクセス速度向上を目指したものである。そのため我々はブロック番号の連続した領域にディレクトリ・セグメントを移動した。これによって実際にアクセス速度が向上することは、1．の終りで述べた"ブロック番号の連続領域は、ディスクの同一または隣接するシリンダに割り付けられていなければならない"という必要条件が、SIMPOSのディスク・ハンドラにより満たされていることにより保証される。

これを裏ずけるため、現在コンパクション前後でのファイル・アクセス性能比較を行っている。

このボリューム・コンパクション機能例は、デバイス構造をボリューム管理以上のレベルで直接意識しない木構造ファイル・システムの最適化として意味があるものと考える。