デバイスミックスストレージシステムにおける アクセス情報を利用したファイルの再配置

相坂　勇気† 　飯澤　健†† 　小沢　年弘†† 　横田　治夫†

† 東京工業大学　〒 152–8550 東京都目黒区大岡山 2 丁目 12–1
†† 株式会社 富士通研究所　〒 211–8588 神奈川県川崎市中原区上小田中 4–1–1
E-mail: †aisaka@de.cs.titech.ac.jp, ††{iizawa.ken,t.ozawa}@jp.fujitsu.com, †††yokota@cs.titech.ac.jp

あらまし　低コストでストレージシステムの性能を向上させる為の手法として，異なる性能のデバイスを組み合わせるデバイスミックスストレージシステムが提案されている．大容量で安価な HDD と高速だが高価な SSD によるミックス環境が例としてあげられるが，システム全体の性能を向上させるために，デバイス間の効率の良いデータ移行が求められる．本研究では，Tiering 方式のデバイスミックスストレージシステムにおいて，ファイル単位でデータの移行を行い，移行の際にファイル間のアクセスの関連性を用いることで性能の向上を目指す．

キーワード　デバイスミックスストレージ，ファイルアクセストレース，データ再配置

1. はじめに

近年の情報技術の進化によって世界で扱われるデータ量は増大し続けており，2011 年は約 1.8 ゼタバイトだったデータ総量は，2020 年には約 40 ゼタバイトに達するとも言われる [1]．データを保存するストレージの 1 つである HDD は，大容量化によってデータ量増加に対応しているが，一方でその性能の伸びは頭打ちとなっている．そのため，HDD のような磁気ディスクではなく，半導体素子を用いた記録媒体である SSD(ソリッドステートドライブ) に注目が集まっている．

SSD は，HDD よりもランダムアクセスに優れ，また省電力である等の優れた特徴がある．しかし，容量あたりの価格が未だ HDD より高価であり，性能向上のために大容量の HDD で構成された既存のストレージシステムを全て SSD で置き換える，といったことは現実的でない．そのためコストを抑えつつ性能を向上させる為の選択肢として，SSD と HDD を組み合わせるデバイスミックスストレージシステムが提案されている．

デバイスを組み合わせる手法としては，Cache 方式と Tiering 方式がある [2]．図 1 に，SSD と HDD による Cache 方式と Tiering 方式のイメージを示す．

Cache 方式では，HDD に全てのデータを置き，SSD を HDD アクセスの際のキャッシュとして用いる．SSD には HDD 内データの一部コピーが置かれ，LRU などのキャッシュアルゴリズムを用いて制御される．SSD の容量が今後増大していくことを考慮すると，SSD と HDD でデータの重複が起こる Cache 方式はストレージ領域の使用効率という観点から見ると好ましくない．一方 Tiering 方式では，データを SSD と HDD で分けて配置し，ユーザからの I/O はデータ配置によって対応するディスクに振り分けられる．Tiering 方式においては，性能向上のため一定時間ごとに HDD と SSD の間でデータの再配置を行うことが一般的である．再配置の際は，アクセスの統計情報を用い，例えば一定時間のアクセスの頻度が高いデータを

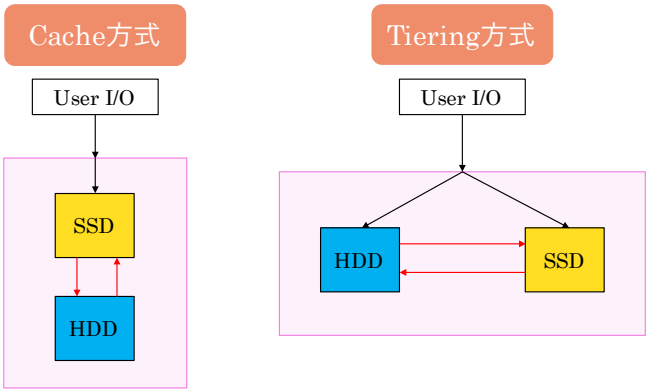

図 1　SSD と HDD による
デバイスミックスストレージシステム

SSD に移行する等の手法が広く知られているが，アクセスパターンによってはその変化を一定期間内の頻度情報から分析・捉えることが難しい [3]．

デバイスミックスストレージの研究は，ブロックベースのシステムに関するものが多い．しかしユーザがデータにアクセスするのはブロックの粒度ではなく，ファイルの粒度であるため，ユーザのデータアクセスの傾向は，ブロックベースではなくファイルベースのアクセスの情報により直接的に現れると考える．そこで本研究では Tiering 方式デバイスミックスストレージにおいて，ユーザのアクセスの傾向を捉えるため，ユーザがデータを利用する粒度であるファイルの単位でデータの移行を行い，その際アクセス履歴の動的解析から求めたファイル間のアクセス関連性に基づいた，アクセス傾向予測を用いる手法 DACFIR(Dynamic Access Correlation base FIle Relocation) を提案する．

2. 関連研究

Tiering における効率的なデータ移行の研究としては，例え

ば文献 [3] がある．この研究は Tier 間の移行を subLUN という 1GB の領域ごとに行うブロックベースの Tiering デバイスミックスストレージシステムを対象としたものである．ここでは分単位のアクセス情報から IO が集中している領域を特定し，その都度移行することで性能向上を図っている．また，文献 [4] は Tier 間の移行を extent という 256MB の領域ごとに行うブロックベースの Tiering デバイスミックスストレージシステムを対象としたものであるが，ここでは一定時間内のアクセスの統計ではなく，アクセスごとにリスト形式で記録したアクセス時刻の間隔の情報を用い，そこから予想した extent に対する次のアクセス時刻を再配置に用い，性能向上を図っている．

Cache 方式，Tiering 方式について，両者を併用するシステムの研究もある．文献 [2] は，そのような両方式を併用したストレージシステムの有用性に関する研究であり，SSD を Cache 方式におけるキャッシュとして使うか，また Tiering 方式における高速階層として使うかの割合を示す SSD キャッシュ容量率を変化させ，複数のワークロードに対してシミュレーションを行い Cache 方式と Tiering 方式の併用がどのような条件で効果があるかを調査している．

本研究で用いたファイルアクセス関連性の抽出法は，文献 [5] で提案されたものに基づく．複数台のディスクで構成されるストレージシステムの省電力化のために，あまりアクセスされないファイルを集めた低頻度ディスクの回転を停止させることで電力消費を抑えるシステムについて，動的にファイル間のアクセスの関連性を抽出し，関連してアクセスされるファイルを 1 つの低頻度ディスクにまとめて配置することにより，不要なディスクのスピンアップを抑え消費電力を削減する DACE という制御手法を提案している．DACE におけるアクセス関連性の抽出について，概要を以下に示す．

DACE では，あるファイルのアクセスに対して一定時間 T の前後に別のファイルへのアクセスが起こる事を，アクセスの共起と呼ぶ．図 2 に，アクセスの共起の例を示す．

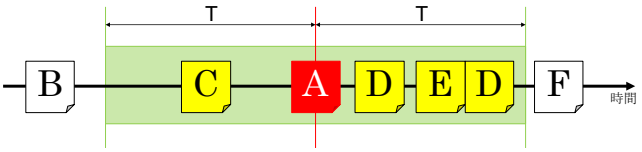

図 2　ファイル A に対するアクセスの共起

あるファイルに対する共起の回数は，そのファイルとの同一タイミングでのアクセスの起こりやすさを表す．

DACE による共起のカウントでは，各ファイルに対してアクセスが共起したファイル，及び共起した回数がアクセス関連性テーブルという形で記録される．表 1 に，アクセス関連性テーブルの例を示す．

DACE では，あるファイルが複数台ある低頻度ディスクのいずれかに再配置される際，同じく低頻度ディスクに再配置される対象となっているようなファイル群とのアクセスの共起回数から関連性を計算し，高い関連性をもつようなファイルと同一のディスクに再配置されるように制御することで，消費電力を削減している．

表 1　ファイル A のアクセス関連性テーブル

File ID	Correlation Count
D	65
C	11
B	7
G	5
⋮	⋮

本研究の提案手法 DACFIR は，DACE に用いられていたアクセスの共起から動的にファイルの関連性を求める手法を Tiering 方式のデータ移行に応用し，HDD から SSD へデータを移行する際に，より関連性が高いファイル群の単位でファイルを移行することで性能向上を目指した制御手法である．

3.　ストレージシステムモデル

ここでは，本研究において仮定したストレージシステムについて述べる．

本研究では，性能要件を見極めるため多重化を行わず，SSD1 台と HDD1 台による Tiering 方式のデバイスミックスストレージシステムを仮定する．

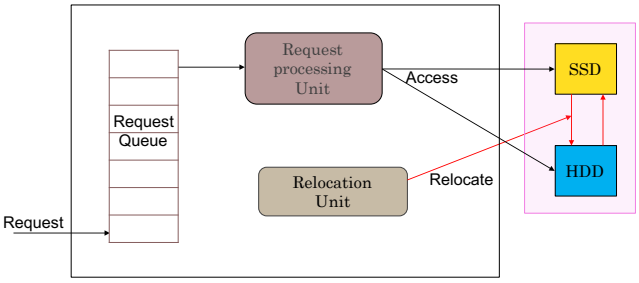

図 3　SSD と HDD による
Tiering システムモデル

仮定したシステムは，要求の貯まる queue, 要求を処理するユニット (Request processing Unit)，データの再配置を行うユニット (Relocation Unit) から構成される．この構成は，他の Tiering 方式の研究における構成を参考にしている．

ファイルの再配置は，FileHandler 内の再配置ユニットが，一定時間 I[s] ごとにファイル N 個を SSD - HDD 間で swap するという方法で行う．ここで，再配置のパラメータ I・N は，システム起動の際に設定し，起動している間は固定であるとする．再配置中のファイルに対するアクセスについては，Read ならば移行元の tier から読み，Write についてはブロックすることとする．

4.　再配置手法

ここでは，本研究で比較する再配置の手法について説明する．

4.1　アクセス頻度による再配置 LFU

提案手法と比較するためベースラインとして LFU(Least Frequently Used) を規定する．LFU では，次の再配置のタイミン

グまでの一定期間におけるアクセスの情報を用い，HDD 内のアクセス頻度が高かったファイル N 個を SSD へ移行，SSD 内のアクセス頻度が低かったファイル N 個を HDD へ移行する．

再配置では，アクセスの情報を用いてアクセス頻度テーブルを作成する．アクセス頻度テーブルは，SSD と HDD 内のファイルについて，File ID 及び期間内にアクセスされた回数を記録する．

表 2 にアクセス頻度テーブルの例を，図 4 に LFU におけるファイル再配置のイメージを示す．

表 2　アクセス頻度テーブル

Files in SSD		Files in HDD	
File	Access Count	File	Access count
D	2455	A	3474
Z	1201	D	1142
R	1161	H	732
G	856	B	717
⋮	⋮	⋮	⋮

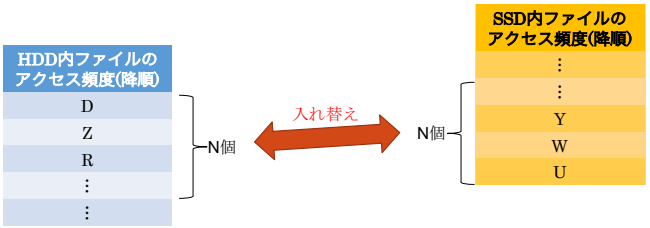

図 4　LFU におけるファイル再配置のイメージ

4.2　提案手法 DACFIR(Dynamic Access Correlation base FIle Relocation)

提案手法では，HDD から SSD へのファイルの移行において，SSD への移行対象の決定の際に，動的にカウントしたアクセスの共起回数を用いる．SSD から HDD への移行では，LFU と同様アクセス頻度が低かったファイル N 個を移行するものとする．

頻度による再配置 LFU では，HDD からの移行対象は一定期間のアクセス数上位のファイル N 個となるが，DACFIR では SSD 移行対象の N 個の決定の際に，より関連性が高いようなファイルのまとまりでファイルが移行されるようにアクセスの共起回数を用いる．

以下に，DACFIR における SSD への移行対象決定の流れを示す．自然数 K, M を DACFIR のパラメータとして用いる．まず，HDD 内の一定期間内のアクセス数上位 K*N 個のリストをアクセス頻度テーブルより取得，そのうち N/M 個 (ファイル群 α とする) についてを移行対象に決定する．

次に，残りの (K*N - N/M) 個について，ファイル群 α 内のファイルにより共起された回数の総和を各ファイルについて求め，その共起回数の上位 (N - N/M) 個を SSD への移行対象に決定する．図 5 に DACFIR におけるファイル再配置のイメージを示す．

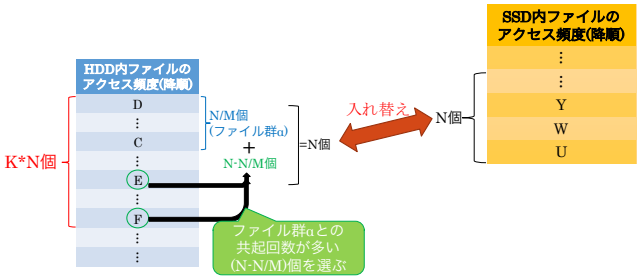

図 5　DACFIR におけるファイル再配置のイメージ

パラメータ K を大きくすると，アクセスの共起回数の計算量が増えてしまうが，よりアクセス頻度が少ないようなファイルについても，アクセスの共起によって移行対象に決定される可能性が上がる．パラメータ M は，ファイル群 α のファイル数を移行ファイル数の何分の 1 にするかのパラメータであり，この値を増やすとファイル群 α のファイル数は小さくなり，その分共起回数から決定されたファイルが多く移行対象になる．

このような決定により，次のタイミングでアクセスが集中するようなファイルについて，前のタイミングのアクセスの頻度では捉えられないものも，アクセス頻度上位のファイルとのアクセスの共起という形で捉えることが出来るようになると期待される．

移行対象決定の方法としては，他にも移行先ディスク内の全ファイルとの共起回数を計算して移行対象を決定する，等様々な方法が考えられたが，ここでは仕組みが簡単で，かつシステムへの実装が容易になるように設計した．

5.　実　　験

5.1　実験目的

提案手法の性能，及びパラメータによる性能の変化や性能向上の条件を調べるため，以下に示す複数の実験を行った．

5.2　実験環境

図 6 に再配置手法の評価のための実験システムの構成を示す．また，表 3 に実験環境を示す．

表 3　実験環境

CPU	Xeon X5460 3.16GHz
Memory	48GB
OS	Linux 2.6.32(x86 64)
SSD	SAS 3.0 MLC 400GB
HDD	SATA 3.0 7200rpm 1TB
Language	C++

アクセスの流れを示す．まず，図の左上にある Workload File に基づき，図上部の Request sending Parent Thread が Child Thread を生成する．Child thread は，Tiering 制御を担う FileHandler へアクセス要求を投げ，FileHandler から結果が戻ってくるまでの時間を測定する．本研究においての応答時間とは，Child Thread がこの部分で測定する時間にあたる．

FileHandler は，内部に要求をためる Request Queue を持つ

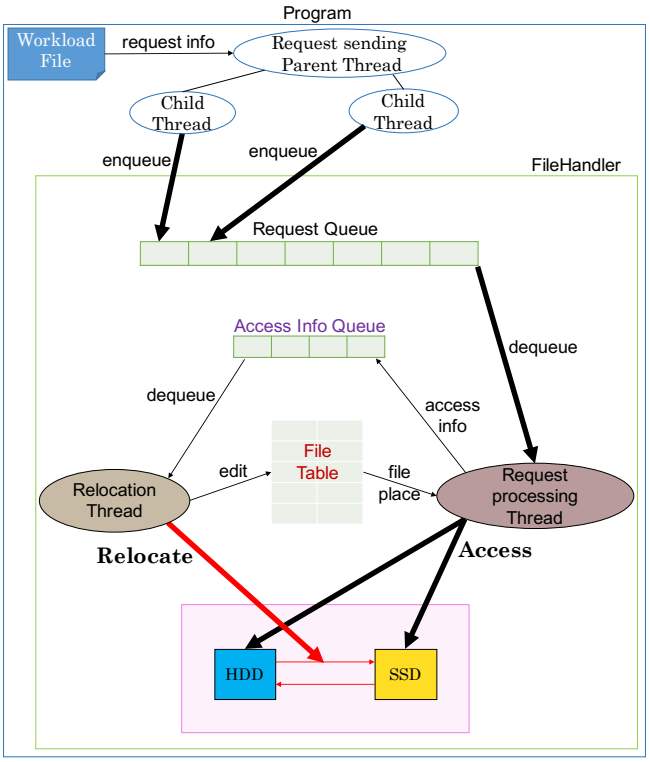

図 6　実験システムの構成

ており，外部から投げられた要求は最初この queue に enqueue される．要求の処理は，FileHandler 内の Request processing Thread が Request Queue から要求を dequeue し，ファイルの配置が書かれている File Table から取得したファイルの配置に基いてディスクへと Access，という流れになる．

ファイルの再配置は，FileHandler 内の Relocation Thread が担当する．Relocation Thread は，起動時に設定されるパラメータ I[s], N 及び再配置種別に基づいて，一定時間 I 毎に SSD - HDD 間で N 個のファイルを swap する．再配置に必要なアクセスの履歴等は，Access Info Queue を用いて，Request processing Thread/Relocation Thread 間でやり取りされる．

5.3　ワークロード

性能評価実験の際のワークロードは，ハーバード大学の EECS で利用されていた NFS サーバのログ [6] を元に，ある一定期間の READ 及び WRITE の要求を抽出し，アクセス対象のファイルごとにまとめたものを利用した．ファイルの初期配置は，File ID が偶数のファイルが SSD，File ID が奇数のファイルが HDD に置かれるようにした．簡単のために，SSD と HDD 内のファイル数は固定とした．表 4 に，ワークロードの概要を示す．

表 4　性能評価実験に使用したワークロードの概要

IOPS	2.22
Read ratio	68.3%
Average access size	241 KiB
Record period	3 hour 9 seconds

5.4　実験 A: 提案手法のパラメータの調整

5.4.1　実験設定

提案手法 DACFIR について，前後何秒のアクセスを共起とみなすかの値 T(Correlation interval) を変化させた際の性能を測定する実験 (実験 A とする) を行った．

以下では，(I, N) = (60, 5), T=t_1 と設定した DACFIR を DACFIR1-t_1，(I, N) = (3600, 25), T=t_2 と設定した DACFIR を DACFIR2-t_2 と呼称する．I(Relocation interval) の 60 秒 (=1 分)，及び 3600 秒 (=60 分) という設定は，人の感覚でわかりやすい値を意識し設定した．実験 A では，T のみを 5, 10, 30, 50, 70, 90, 100, 150 と変化させた DACFIR1-5〜DACFIR1-150, DACFIR2-5〜DACFIR2-150 について平均応答時間を測定した．DACFIR における他のパラメータ K, M は今回は (K, M) = (2, 2) と固定した．これは，次の実験 B においても同様である．K, M を変化させた場合の性能の測定については今後の課題とする．

各設定におけるパラメータ等を表 5 に示す．

表 5　実験 A の各パラメータ

Relocation Type	I (Relocation interval)	N (Relocation files)	T (Correlation interval)	K, M (Parameter for DACFIR)
DACFIR1-5	60	5	5	2, 2
DACFIR1-10	60	5	10	2, 2
DACFIR1-30	60	5	30	2, 2
DACFIR1-50	60	5	50	2, 2
DACFIR1-70	60	5	70	2, 2
DACFIR1-90	60	5	90	2, 2
DACFIR1-100	60	5	100	2, 2
DACFIR1-150	60	5	150	2, 2
DACFIR2-5	3600	25	5	2, 2
DACFIR2-10	3600	25	10	2, 2
DACFIR2-30	3600	25	30	2, 2
DACFIR2-50	3600	25	50	2, 2
DACFIR2-70	3600	25	70	2, 2
DACFIR2-90	3600	25	90	2, 2
DACFIR2-100	3600	25	100	2, 2
DACFIR2-150	3600	25	150	2, 2

5.4.2　実験 A の結果

DACFIR1-5〜DACFIR1-150 で測定した平均応答時間のグラフを図 7 に，SSD ヒット率を図 8 に示す．DACFIR2-5〜DACFIR2-150 で測定した平均応答時間のグラフを図 9 に，SSD ヒット率を図 10 に示す．

平均応答時間の結果を見ると，DACFIR1-30 が平均応答時間=2038(usec) で最も性能が良かった．60 秒ごとに 5 個再配置した DACFIR1-5〜DACFIR1-150 の中では T=30 に設定した DACFIR1-30 が最も性能が高く，3600 秒ごとに 25 個再配置した DACFIR2-5〜DACFIR2-150 の中では T=150 に設定した DACFIR2-150 が最も性能が高い．SSD ヒット率の結果を見ても，DACFIR1-30, DACFIR2-150 の場合に SSD ヒット率が僅かに高くなっていることが分かる．

このような結果から当該ワークロードでは，60 秒ごとに 5 個再配置する場合は T=30, 3600 秒ごとに 25 個再配置する場合は T=150 とすることで SSD ヒット率が向上し，平均応答時間

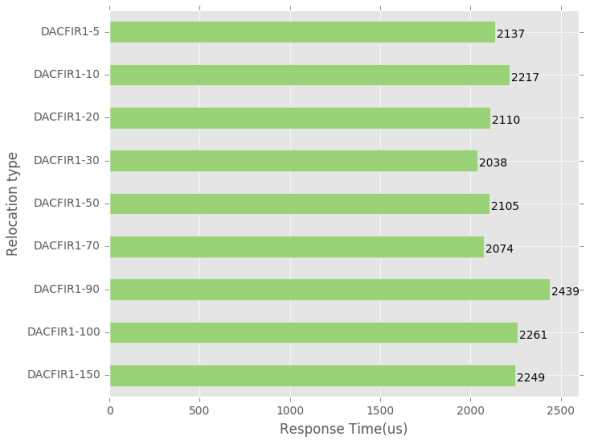

図 7　実験 A の平均応答時間グラフ-1 (I, N = 60, 5)

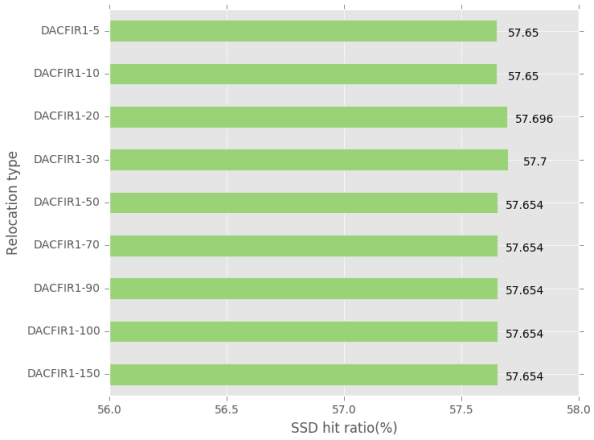

図 8　実験 A の SSD ヒット率グラフ-1 (I, N = 60, 5)

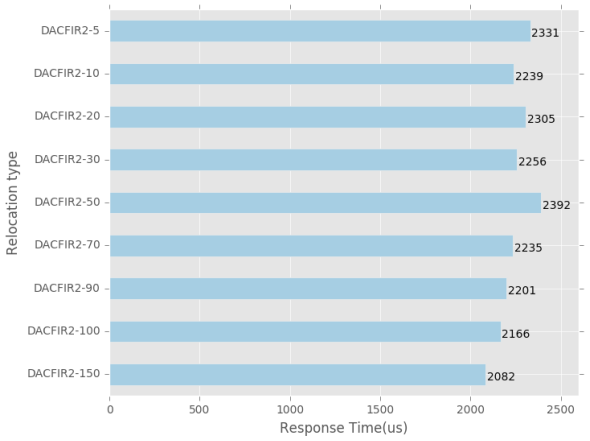

図 9　実験 A の平均応答時間グラフ-2 (I, N = 3600, 25)

の短縮に繋がったのではないかと考えられる．

5.5　実験 B: 他の再配置手法との比較

5.5.1　実 験 設 定

次に，初期配置からファイル配置を変化させない (NoR, No Relocation), ベースラインとしての LFU，及び実験 A で調整

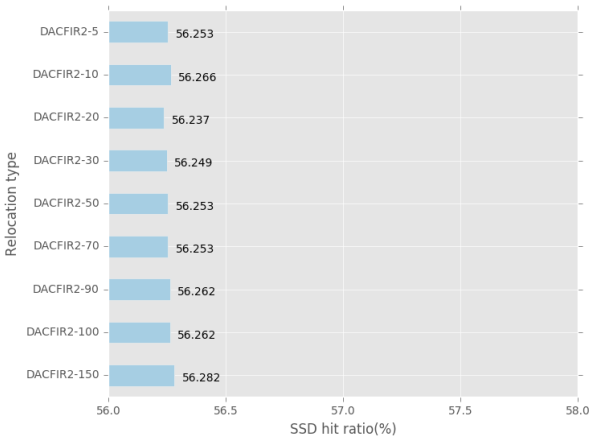

図 10　実験 A の SSD ヒット率グラフ-2 (I, N = 3600, 25)

したパラメータを使った提案手法について平均応答時間を測定し比較する実験を行った．

各設定におけるパラメータ等を表 6 に示す．

表 6　実験 B の各パラメータ

Relocation Type	I (Relocation interval)	N (Relocation files)	T (Correlation interval)	K, M (Parameter for DACFIR)
NoR(No Relocation)	-	-	-	-
LFU1	60	5	-	-
LFU2	3600	25	-	-
DACFIR1-30	60	5	30	2, 2
DACFIR2-150	3600	25	150	2, 2

5.5.2　実験 B の結果

ファイルを初期配置から移動しなかった NoR，アクセス頻度のみによる再配置を行った LFU1 及び LFU2, 実験 A で調整したパラメータを用いた DACFIR1-30 及び DACFIR2-150 における平均応答時間を図 11 に示す．

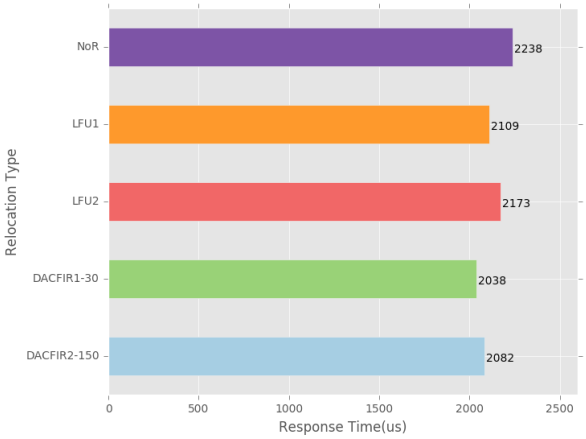

図 11　実験 B の平均応答時間グラフ

実験 B の結果を見ると，NoR，再配置なしの場合よりその他の再配置を行った場合のほうが性能が良くなっている．特に DACFIR1-30，つまり再配置間隔 I=60，再配置ファイル数

N=5，共起とみる間隔 T=30 と設定した場合の提案手法が最も性能が良くなっていることが分かる．

LFU1 は LFU2 より，また DACFIR1-30 は DACFIR2-150 よりも良い結果が出ている．LFU1/DACFIR1-30 は 60 秒ごとに 5 個のファイルを再配置する設定であり，一方 LFU2/DACFIR2-150 は 3600 秒ごとに 25 個のファイルを再配置する設定である．このことから，ファイルベースの Tiering においては，少数のファイルをこまめに再配置する方が性能の向上が見込めると考える．

6. まとめと今後の課題

6.1 ま と め

本研究では，デバイスミックス環境における効率的なデータ移行のため，ユーザがデータにアクセスする粒度であり，ユーザのデータアクセスの傾向がより直接的に現れると考えられる，ファイルをベースにしたシステムについて研究を行った．またデータ移行の際に，ファイル間のアクセスの関連性を用いる手法 DACFIR を提案した．本手法ではアクセスの共起を数えることでファイル間の関連性を動的に抽出する手法を応用し，より関連性が高いファイル群の単位でファイルを移行することで性能向上を目指した．再配置なしの場合，アクセス頻度による再配置を行った場合，提案手法を用いた場合を比較するための実験システムを構成し，実験を行った．適切にパラメータを設定した提案手法で，良い性能が出る場合があることを実験により確認した．

6.2 今後の課題

Tiering 方式におけるデータの再配置は再配置するデータのサイズと再配置間隔，及びワークロードによって大きく結果が変化すると考えられ，考察のため設定を変化させた複数の場合について更に測定を行う必要がある．また，提案手法で使用するパラメータも複数の値に設定し，実験を行う必要がある．特に，今回の実験では提案した手法においての関連性の利用の際のパラメータについて，アクセスの共起とみなす間隔 T についてのみしか測定が出来なかった．他 2 つのパラメータ (K, M) についても変化させた設定で測定を行う予定である．複数あるパラメータの値を実験的に決定していくのはコストが高いため，ワークロードに応じてパラメータを動的に決定できるような仕組みについても検討したい．

更に，今回の実験においてはファイルの初期配置は File ID の偶奇によって SSD と HDD に分けたため，SSD と HDD にほぼ同数のファイルが置かれる環境となっているが，使用できる SSD の領域がより少ない場合なども考慮した実験を行う事も必要であると考える．加えて，今回の研究ではファイルごとのサイズの違いは考慮せず，ファイルの数のみを考えた設定で研究を行ったが，大きいサイズのファイルによる移行コストの増大や，SSD の容量制限などの問題を考えるため，ファイルサイズの情報も利用するようシステムの拡張が必要と考える．

ミックスするデバイスとして今回は SSD 及び HDD を用いたが，その他に現在研究が進められている PCM や MRAM などのストレージクラスメモリを利用した構成も考えられる．ストレージクラスメモリの性質なども考慮した制御手法について，検討していく．

文　　献

[1] 総務省, ”平成 26 年版情報通信白書”, pp.100f., 2014 http://www.soumu.go.jp/johotsusintokei/whitepaper/ja/h26/pdf/

[2] 林真一, 薦田憲久 , ”階層ストレージにおけるボリューム階層化方式と SSD キャッシュ方式の評価,” 電学論 C(電子・情報・システム部門), vol.134, No.3, pp.459–465, 2014.

[3] 大江和一, 岩田聡, 本田岳夫, 河場基行, ”On-The-Fly - Automated Storage Tiering (OTF-AST) の提案,” 情報処理学会システムソフトウェアとオペレーティング・システム研究会, pp.1-10, 2014.

[4] 新屋敷裕太, 飯澤健, 小沢年弘, 荒堀喜貴, 横田治夫, ”アクセス時間に基づいた次アクセス予想によるデバイスミックスストレージシステムの制御手法”, 第 8 回データ工学と情報マネジメントに関するフォーラム (DEIM 2016), 2016.

[5] 入谷優, 横田治夫, ”動的に記録されたアクセス関連性に基づくデータ配置による省電力ストレージシステムの消費電力と性能に対する影響”, 第 4 回データ工学と情報マネジメントに関するフォーラム (DEIM 2012), 2012.

[6] Daniel Ellard, Jonathan Ledlie, Pia Malkani, and Margo Seltzer, ”Passive NFS Tracing of Email and Research Workloads,” In Proceedings of the 2nd USENIX Conference on File and Storage Technologies, pp. 203–216, Berkeley, CA, USA, 2003. https://www.eecs.harvard.edu/margo/papers/fast03/paper.pdf