Vol.2014-DPS-159 No.34
Vol.2014-MBL-71 No.34
2014/5/16

仮想化環境における データマイグレーションによるディスク読み書き性能の向上

杉本 洋輝[†1]　山口 実靖[†1]

ストレージの I/O 性能を向上させる手法としてストレージ内のデータの配置を変更する手法があり，これまで研究されてきている．近年急速に普及が進んでいる仮想マシン（VM）環境では，ストレージ上に巨大な仮想 HDD イメージファイルが複数配置され，巨大なイメージファイル間の長距離のシークを繰り返し I/O 性能が大きく低下することが多い．よって，VM 環境ではデータの再配置による I/O 高速化が特に重要になると考えられる．しかし，これまでに提案されてきた再配置手法の多くは VM 環境を想定しておらず，これらの手法を工夫なく VM 環境に適用しても得られる性能向上は限定的であると予想される．二重ファイルシステムで構成される VM 環境を考慮した手法として山田らによる再配置手法[1]があるが，当該手法は読み込み処理のみを対象としており，書き込み処理を考慮していない．本稿では 1 台の物理ストレージ上に複数の仮想 HDD イメージファイルが存在する環境を想定し，ホスト OS とゲスト OS の両方にファイルシステムが存在する VM 環境に適したデータ再配置手法および書き込み処理を考慮した手法を提案し，その性能評価を行う．評価の結果，提案手法により最も高い性能向上が得られることが確認できた．

1. はじめに

計算機システムで扱うデータ量の増加に伴い，増加したデータを高速に処理するためにストレージシステムには高い I/O 性能が要求されるようになっている．しかし，主要なストレージデバイスであるハードディスクドライブ（HDD）はデータへアクセスする際，磁気ヘッドのシーク，磁気ディスクの回転といった動作を伴うためデータへのランダムアクセスの性能が高くない．ランダムアクセス性能の向上を実現する手法の一つに，ストレージ上のデータの配置を変更する手法があり，これまで研究されてきている．これらの手法では，高い頻度でアクセスされるデータをストレージの中央に配置して平均シーク距離を低減させたり[2]，連続アクセスされるファイル群を近隣に配置することによりシーク距離を削減されるなどをしている[3]．

また，近年は VM 環境が普及してきているが，VM 環境では巨大な仮想 HDD イメージファイルがストレージ上に複数存在することが多く，その場合にストレージデバイスは巨大な仮想 HDD イメージファイルの間のシークを頻繁に行うことになり I/O 性能が低くなる．よって，VM 環境ではストレージ上のデータの再配置による高速化が特に重要であると言える．しかし，これまでに提案されてきた多くの再配置手法は物理ストレージ上にデータが直接格納されていることを想定しており，図 1 のように二重にファイルシステムが構築される VM 環境を想定していない．二重ファイルシステム構造の VM 環境を想定した再配置手法として山田らよる再配置手法[1]があるが，当該手法は読み込み処理のみを対象としており書き込み処理を考慮していない．よって，VM 環境および書き込み処理に適したディスク再配置に関する考察が重要であると考えられる．

本稿では，二重ファイルシステム環境および書き込み処理に適した再配置手法の提案と評価を行う．

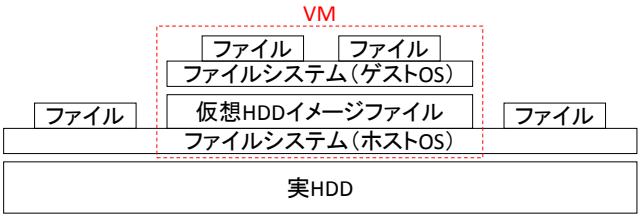

図 1　二重ファイルシステム構造

2. Ext2，Ext3，Ext4 ファイルシステム

Ext2，Ext3，Ext4 は BSD の Fast File System(BSD FFS)をもとに開発されたファイルシステムであり，複数のブロックをブロックグループにまとめて管理する．通常，ブロックサイズは 4096[byte]である[4][5]．ブロックグループの構造を図 2 に示す．図の左側はファイルシステムでフォーマットした HDD 上のレイアウトを示しているおり，ブートセクタは PC を起動する際に OS を読み込むためのものである．図の右側はブロックグループ内のレイアウトを示したものである．inode については 2.1 節，データブロックについては 2.2 節で解説をする．Ext4 にはエクステント機能があるが，本研究では使用していない．

2.1 inode

本節で inode について説明をする．inode は個々のファイルやディレクトリに関する情報を管理している[4][5]．ファイルやディレクトリの情報としては表 1 に示すようなデータが保持されている．表 1 の様に inode 内にはデータブロックのブロックアドレス（データが配置されているディスクのアドレス．次節にて述べる）が含まれている．各ブロックアドレスは 4[Byte]で表現される．

†1 工学院大学大学院工学研究科電気・電子専攻
Electrical Engineering and Electronics, Kogakuin University Graduate School.

Vol.2014-DPS-159 No.34
Vol.2014-MBL-71 No.34
2014/5/16

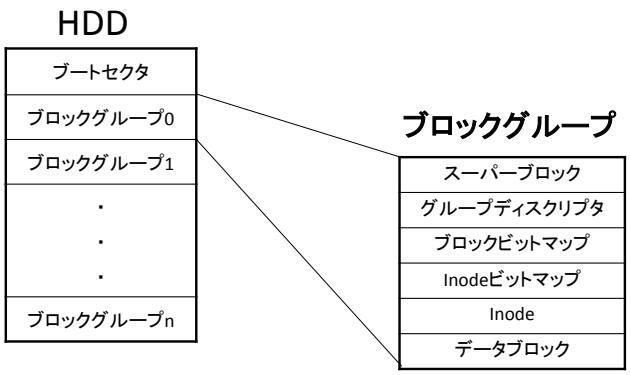

図 2　ブロックグループ構造

表 1　inode の詳細

格納アドレス[Byte]	説明
0 ～ 1	パーミッション
2 ～ 3	ユーザーID の下位 32bit
4 ～ 7	ファイルサイズの下位 32bit
8 ～ 11	最終アクセス日時
16 ～ 19	最終更新日時
40 ～ 87	直接参照ブロックアドレス
88 ～ 91	1 段間接参照ブロックアドレス
92 ～ 95	2 段間接参照ブロックアドレス
96 ～ 99	3 段間接参照ブロックアドレス
108 ～ 111	ファイルサイズの上位 32bit
120 ～ 121	ユーザーID の上位 32bit

2.2 データブロック

本節にて，図 2 のデータブロックと inode 内のブロックアドレスについて説明をする．ファイルのデータは，ディスク内のデータブロックに保存される．各ファイルのデータがディスクのデータブロック内のどのブロックに保存されているのかは, inode 中のブロックアドレスに記録されている．ブロックアドレスにはファイル内の各ブロックが保存されるディスク内のブロックの番号が格納されており，このブロックアドレスを参照することによりファイルデータにアクセスすることができる．

inode 内には 12 個の直接参照のブロックアドレスが格納されている．図 3 の様に，Block address[0]から Block address[11]が直接参照ブロックアドレスであり，ファイル内の 0 個目から 11 個目までのブロックがディスク内のどのブロックに保存されているかを示している．

Block address[12]から Block address[14]には，1 段，2 段，3 段間接参照ブロックアドレスが格納されている．1 段参照のブロックアドレスの場合，図 4 の様に Block address[12]で示されるブロックには,「ブロックアドレスのリスト」が格納されている．通常，1 ブロックが 4096 バイトであり，ブロックアドレスが 4 バイトであるため,「ブロックアドレスのリスト」には 1024 個のブロックアドレスが格納されている．すなわち，Block address[12]のアドレスで示されるブロックを読み込むことにより 1024 個のブロックアドレスが入手され, これらが 1024 個分のファイルブロックの格納場所を示している．2 段参照の場合，Block address[13]で示されるアドレスには,「ブロックアドレスのリスト」が格納されており，それぞれのブロックアドレスが示す先にはさらに「ブロックアドレスのリスト」が格納されている．そして，そのアドレスが指す先にファイルのデータが格納される(図 5 参照)．3 段参照の場合は図 6 の様に，Block address[14]が示す先に「ブロックアドレスのリスト」があり，それぞれからポインタを 3 回たどることによりファイルのデータにアクセスすることができる．

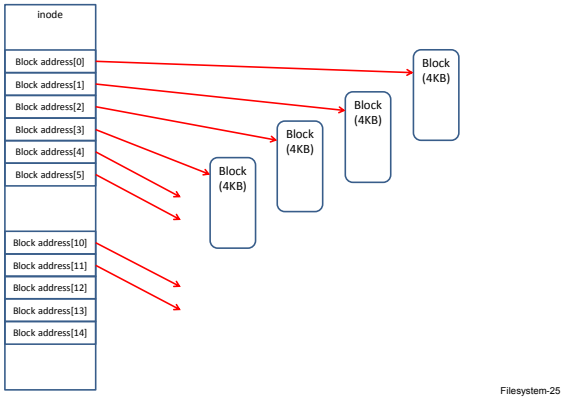

図 3　直接参照の構造

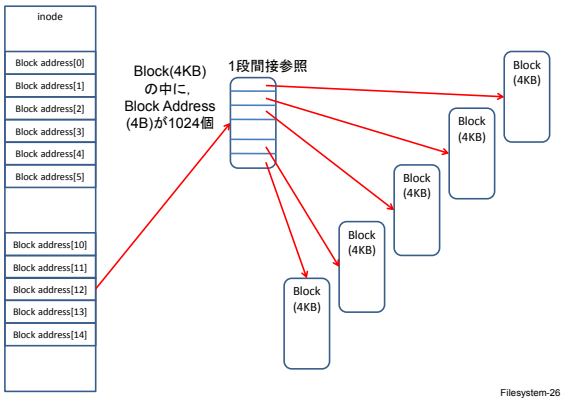

図 4　1 段間接参照の構造

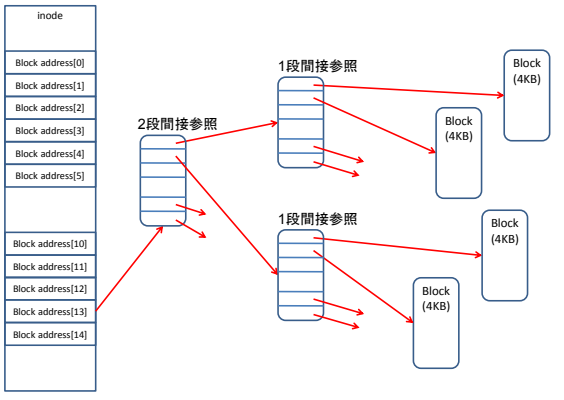

図 5　2 段間接参照の構造

Vol.2014-DPS-159 No.34
Vol.2014-MBL-71 No.34
2014/5/16

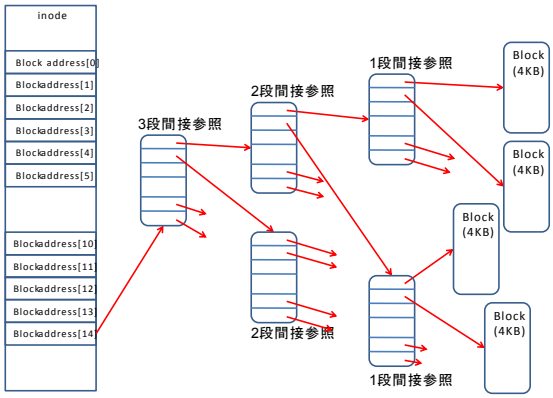

図 6　3 段間接参照の構造

3. 仮想マシン

仮装マシン(VM)は，仮想化された CPU，メモリ，HDD などの計算機資源を用いて動作する仮想的な計算機である．実マシン上で動作する OS をホスト OS，VM 上で動作する OS をゲスト OS と呼ぶ．

VM 環境では実マシンの実 HDD 内の特定の領域が仮想 HDD として確保され，それがゲスト OS に HDD として認識される．仮想 HDD を作成するには 2 つの方法がある[1]．1 つは図 1 の様にホスト OS のファイルシステム上にイメージファイルを作成して，それを仮想 HDD とする方法である．この方法ではイメージファイル内にゲスト OS のファイルシステムが構築されるため，ホスト OS のファイルシステムとゲスト OS のファイルシステムの二重ファイルシステム構造となる．ゲスト OS 内でディスクアクセスが生じると，ホスト OS 上でイメージファイルへアクセスが発生する．もう 1 つの方法は実マシンのパーティションを仮想 HDD として割り当てる方法である．この方法ではストレージ上の空きパーティションに直接ゲスト OS のファイルシステムが構築される．ゲスト OS 内でディスクアクセスが生じると，割り当てたパーティションへのアクセスが発生する．

イメージファイル方式では，仮想 HDD イメージファイルがホスト OS 上の通常ファイルとして扱われるため，VM 環境を他の実マシンに移動することや，ホスト OS ファイルシステム層においてディスクレイアウトを変更することが可能となる．ただし，ゲスト OS におけるファイルアクセスがゲスト OS とホスト OS の両 OS のファイルシステムを経由して処理されるため，I/O 処理に伴う負荷は多くなると考えられる．

これに対してパーティション方式を用いた場合は，ゲスト OS におけるファイルアクセスはホスト OS の実パーティションへのアクセスとなるため，経由するファイルシステムはゲスト OS のものだけであり，I/O 処理に伴う負荷はイメージファイル方式より少ないと考えられる．ただし，VM 環境の移動やディスクレイアウトの変更は困難であり，再配置の柔軟性はイメージファイル方式より低いと考えられる．

4. ブロックアドレス書き換えによるデータマイグレーション

本節では，本研究で用いる VM イメージファイルの再配置手法の説明を行う．2 章で述べたように，ファイルデータの格納場所は inode 内のブロックアドレス(参照ブロックを含む)により決定されている．よって，ブロックアドレスを修正することによりデータの格納場所の変更(データマイグレーション)を行うことができる．

VM のイメージファイルのブロック再配置は以下の手順で行う．

(a) 移動対象ブロックのブロックアドレス(移動元ブロックアドレス)を取得する．
(b) ブロックビットマップより空きブロック情報を取得し，移動先ブロックアドレスを決定する．
(c) 移動先ブロックアドレスへデータのコピーを行う．
(d) inode 内のブロックアドレスを移動元から移動先に書き換える．

5. 関連研究

本章にて，本研究と関連する既存の研究を紹介する．初期のディスクレイアウトの理論に関する研究として文献[2]があり，シミュレーションによる研究として文献[6]-[9]がある．文献[2]では，最高頻度アクセスデータをストレージの中央に配置する organ pipe 手法がランダムアクセスに適していることが示されている．また，organ pipe 手法を現実のワークロードに適用した研究として cylinder shuffling[9]がある．文献[9]ではシリンダー単位で並び替えを行うが，並び替えの単位をブロックとすることによりさらなる高速化を実現した研究として文献[8]がある．また，実システムにおける block shuffling について最初に述べた研究として文献[10]がある．

また，これら再配置の実現には HDD の幾何学的構造が既知である必要があることが多く，物理ディスクから幾何学的構造を調査する研究として文献[11]-[14]がある

次にファイルシステムレベルの研究を紹介する．FFS[15]やその後続の研究[16][17]にて，関連するデータブロックと inode をディスク上の近隣に配置することにより I/O 速度を向上させる方法が提案されている．これらの手法は動的なアクセスパターンを考慮しないため，性能向上の程度が制限されてしまうことが指摘されている[8][10]．また文献

[18]にて，参照の局所性ではなく，微小ファイルの距離を近づけることに着目して性能を上げる方法が提案されている．

ログ構造化ファイルシステム[19]では，大幅な書き込みの性能の向上が実現されている．また，アクセス頻度の高いファイルをディスクの外周に配置することによりログ構造化ファイルシステムをさらに高速化する研究がなされている[20]．

HFS[21]の Hot File Clustering や Smart File System[22]では，ファイルシステムが実行時アクセスパターンを観察し，高頻度アクセスデータを予約領域に移動を行っている．

FS2[3]では，ファイルの複製を用いて連続アクセスされるファイル，あるいはその複製を近隣に配置することによりさらなる高速化を実現している．

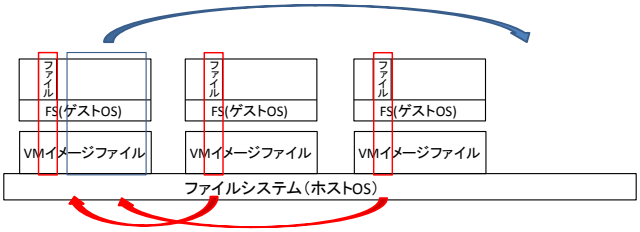

図 7　VM 環境および読み込みを考慮した手法

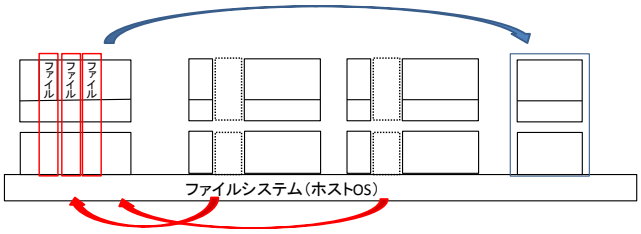

図 8　VM 環境および読み込みを考慮した手法

二重ファイルシステムで構成される VM 環境を考慮した手法として山田らによる手法[1]がある．VM 環境では，ホスト OS のファイルシステム上に巨大なイメージファイルが作成され，そのイメージファイルの中にゲスト OS のファイルシステムが構築される．ホストファイルシステムのブロック使用状況管理においては，イメージファイルにより確保されている全領域が使用中としてみなされる．しかし，このイメージファイルの中にゲストファイルシステムが構築され，ゲストファイルシステムがこのイメージファイルの各ブロックを使用中あるいは未使用として分類する．すなわち，ホストファイルシステムにおいては使用中と扱われるがゲストファイルシステムにおいては未使用と扱われる領域がある．この領域は新規書き込み時以外にアクセスされることはなく，実質的に未使用領域である．当該手法では，この実質的な未使用領域にデータを移動し，VM 内の他のデータとの距離の低減を図っている．しかし，当該研究は読み込み処理のみを考慮しており，書き込みを含む処理での評価はされていない．

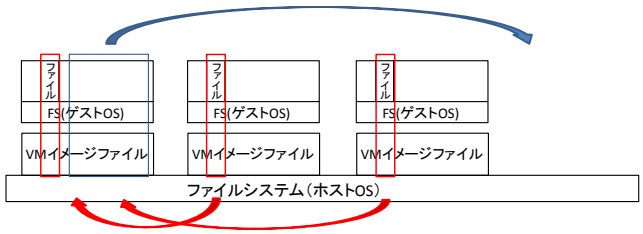

図 9　VM 環境および書き込みを考慮した手法

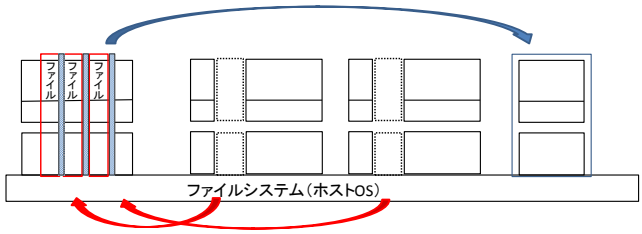

図 10　VM 環境および書き込みを考慮した手法

6. 提案手法

本章にて，二重ファイルシステムおよび書き込みを考慮した手法を提案する．既存手法[1)]が対象としている読み込み処理では未使用領域を使用することはないが，書き込みを行う際に未使用領域は使用されることがある．そこで本提案手法では図 9，図 10 のように，ゲストファイルシステムにおける未使用領域を後方に配置し，空いた領域にホットスポットと，書き込み用の未使用領域を移動する．よって，今後ホットスポットの論理的近辺(ファイルシステムブロック番号が近いブロック)に書き込み要求が生じてもこの「書き込み用未使用領域」に書き込まれるため，ホットスポットと新規書き込みデータが物理アドレス的な遠方(HDD セクタ番号が大きく異なる領域)に配置されず，性能が低下することがないと期待される．

7. 性能評価

7.1 実験環境

提案手法の有効性を確認するために性能評価実験を行った．1 台の物理マシン上に 3 台の VM を同時に稼動させ，各 VM 上でベンチマークソフト TPC-C を実行し再配置前，二重ファイルシステムおよび読み込みのみを考慮した既存手法，二重ファイルシステムおよび読み込みと書き込みを考慮した提案手法の 3 通りの手法で性能を測定した．物理マシンの環境は表 2，仮想マシンの環境は表 3 の通りである．

表 2　物理マシンの仕様

OS	CentOS6.4
CPU	Intel Celeron CPUG530 2.40GHz
Memory	2[GB]
HDD	2[TB]
	7200[rpm]
Kernel	2.6.32.57
仮想化システム	Xen-4.1.2
ファイルシステム	Ext2

表 3　仮想マシンの仕様

OS	CentOS 6.4
CPU	Intel Celeron CPUG530 2.40GHz
Memory	428[MB]
仮想 HDD	100[GB]
Kernel	2.6.32.57
ファイルシステム	Ext2

表 4　tpcc-mysql の設定

Warehouse	10
Connection	1
Ramp-up time[s]	300
Measurement interval[s]	1200

表 5　マイグレーションしたデータ量

ホットスポット[GB]	2
書き込み用未使用領域[GB]	2

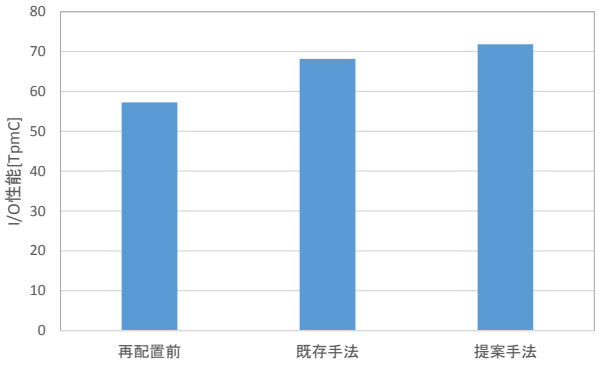

図 11　3VM の I/O 性能の比較

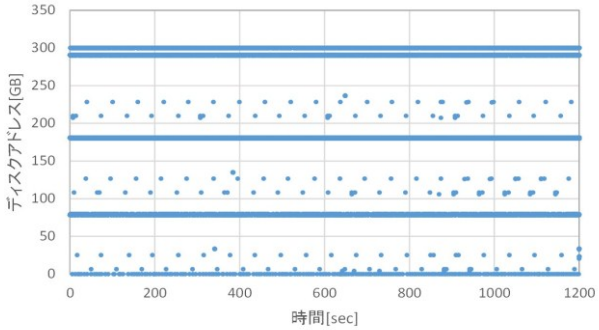

図 12　再配置前のアクセス分布

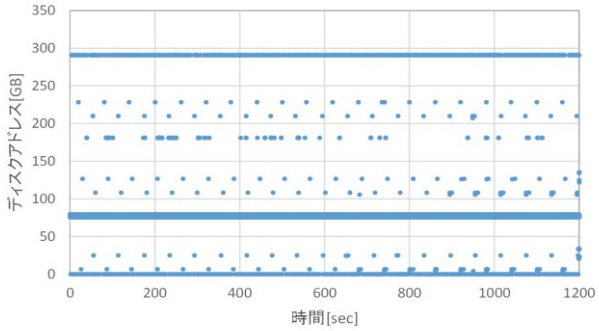

図 13　既存手法のアクセス分布

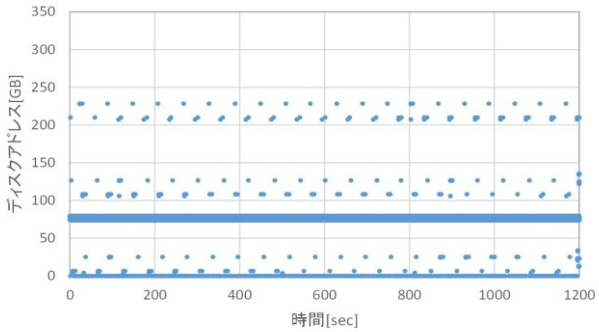

図 14　提案手法のアクセス分布

7.2　実験結果 (3VM)

測定結果を図 11 に示す．縦軸が I/O 性能を表している．図より，提案手法の VM の I/O 性能は再配置前よりも 25%高く，既存手法よりも 5%高いことが分かり，提案手法が有効であることが確認された．

各手法における HDD アクセスの時刻とアクセスアドレスを図 12 から図 14 に示す．既存手法と提案手法は再配置を行う前に 75[GB]，180[GB]，300[GB]付近にあったホットスポットを 75[GB]付近に移動し，アクセスを集中させている．これにより再配置前に比べシーク距離を削減することができ，性能が向上したものと考えられる．

既存手法は書き込み領域を確保していないため，ホットスポットを移動後も，180[GB]，300[GB]付近に新規書き込

みが発生し，そこへもアクセスが集中してしまい，提案手法よりもシーク距離を削減することができずに性能が提案手法よりも低くなってしまったものと考えられる．

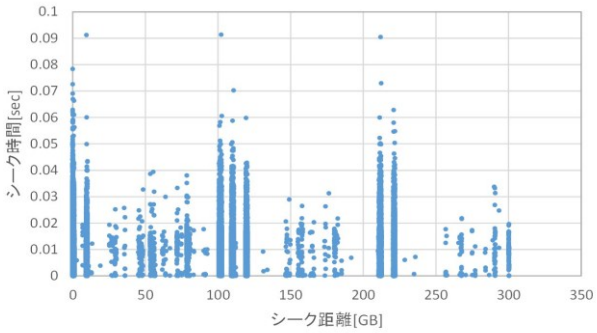

図 15　再配置前のシーク時間とシーク距離の関係

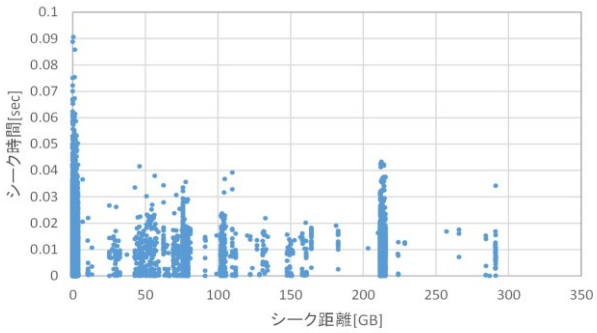

図 16　既存手法のシーク時間とシーク距離の関係

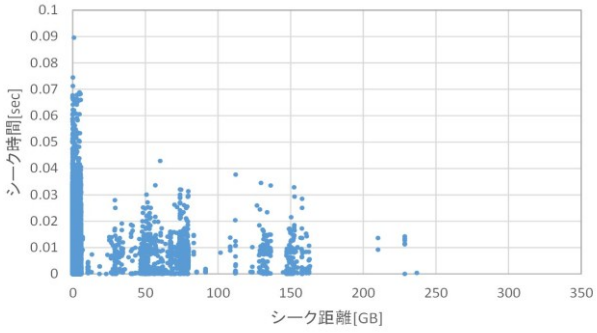

図 17　既存手法のシーク時間とシーク距離の関係

7.3 実験結果(2VM)

次に単一物理マシン上で 2 台の VM が稼動する環境における評価を行う．2VM 時は 3VM 時と比較してイメージファイル数と巨大イメージファイル間長距離シークの長さが減るため，提案手法の有効性が減少すると予想される．

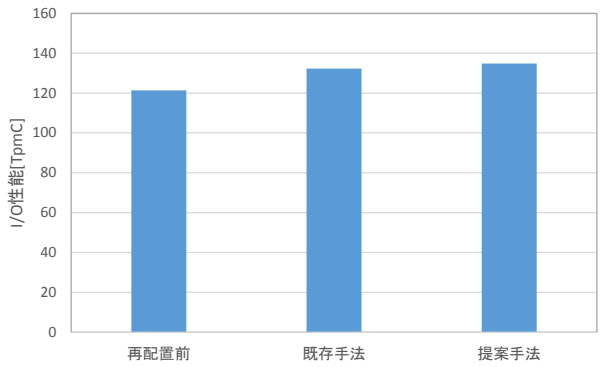

図 18　2VM の I/O 性能の比較

実験結果を図 18 に示す．実験環境は VM 数以外前節と同様である．図より 2VM という最小の複数 VM 環境でも提案手法は有効であることがわかる．

また，3VM 時と比較して，2VM 時の性能向上の程度は減少しており，このことから 4VM 以上の環境では提案手法によりさらなる性能向上が実現可能であると期待できる．

7.4 書き込み量が確保量を超える状況での評価

提案手法では書き込み用に確保した領域に書き込みを行うことでシーク距離の削減を行っている．よって，書き込み量が書き込み用領域のサイズを上回る状況では提案手法の有効性が減少することと予想される．本節では，書き込み量が書き込み用領域サイズを超える状況における評価を行う．TPC-C を 250 回実行し，書き込み用領域以外にも書き込みが発生する環境にて実験を行った．

性能測定結果を図 18 の提案手法（書き込み量超過）に，実 HDD へのアクセス分布を図 19 に，アクセスのシーク距離とシーク時間の関係を図 20 に示す．

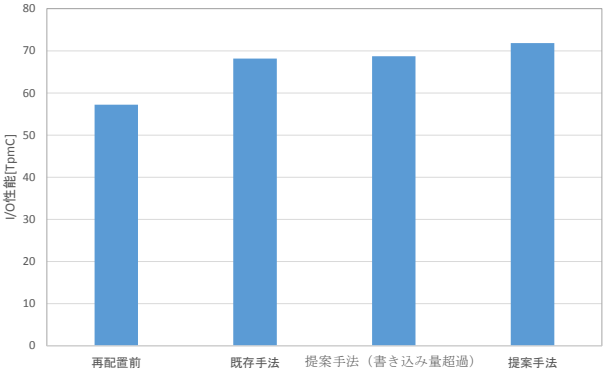

図 19　提案手法と提案手法（書き込み量超過）の I/O 性能の比較

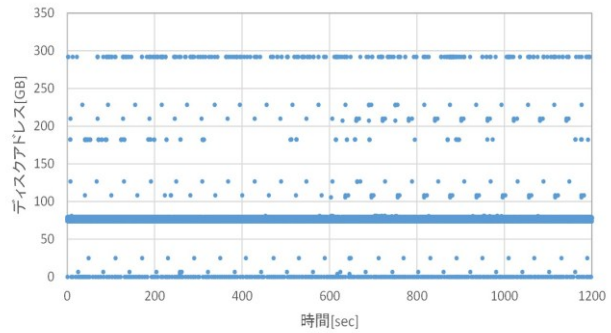

図 20　提案手法（書き込み量超過）のアクセス分布

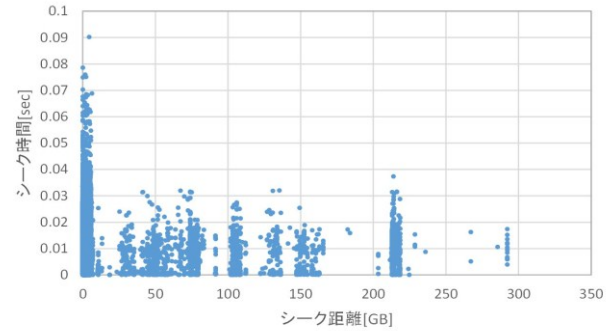

図 21　提案手法（書き込み量超過）のシーク時間とシーク距離の関係

図より，書き込み用領域サイズを超える書き込みが発生すると，提案手法は性能が低下することがわかるが，低下しても既存手法を下回らない結果となった．よって提案手法は書き込み量が書き込み用領域サイズを超えない状況であれば有効であり，書き込み量が書き込み用領域サイズを超えてしまっても既存手法と変わらない性能となり，デメリットはないものと考えられる．

8. おわりに

本稿では VM 環境および書き込みを考慮したデータ再配置手法の提案と評価を行った．評価の結果，書き込み量が書き込み用領域サイズを超えない状況であれば，再配置前や既存手法よりも高い性能を示すことが確認された．

今後は動的な再配置の実現について考察していく予定である．

謝辞

本研究は JSPS 科研費 24300034, 25280022, 26730040 の助成を受けたものである。

参考文献

1) Masaya Yamada, Saneyasu Yamaguchi, "Filesystem Layout Reorganization for Virtualized Environment with Two Filesytems", The 9th IEEE International Conference on Autonomic and Trusted Computing, September 2012
2) C.K.Wong, "Algorithmic Studies in Mass Storage Systems," Computer Sciences Press, 1983.
3) Hai Huang, Wanda Hung, Kang G. Shin "FS2: Dynamic Data Replication in Free Disk Space for Improving Disk Performance and Energy Consumption," SOSP'05, pp.263-276, October. 2005.
4) Brian Carrier，"File System Forensic Analysis，" Addison-Wesley Professional，2001
5) Moshe Bar，"Linux File Systems，" McGraw-Hill Companies，2001
6) Robert English and Stephnov Alexander, "Loge: A Self-Organizing Disk Controller," Proceedings of the Winter 1992 USENIX Conference, 1992.
7) David Musser, "Block Shuffling in Loge," HP Technical Report CSP-91-18, 1991.
8) C.Ruemmler and J. Wilkes, "Disk Shuffling," HP Technical Report, HPL-CSP-91-30, 1991.
9) P.Vongsathorn and S. D. Carson, "A System for Adaptive Disk Rearrangement," Software Practice Experience, 20(3): 225-242, 1990.
10) Sedat Akyurek and Kenneth Salem, "Adaptive Block Rearrangement, Computer Systems," 13(2): 89.121, 1995.
11) M.Aboutabl and A.Agrawala and J.Decotignie, "Temporally Determinate Disk Access: An Experimental Approach," Technical Report, CS-TR-3752, 1997.
12) Zoran Dimitrijevic et al., "Diskbench: User-level Disk Feature Extraction Tool," Technical Report, UCSB, 2004.
13) J.Schindler and G.Ganger, "Automated Disk Drive Characterization," Technical Report CMU-CS-99-176, 1999.
14) N.Talagala and R.Arpaci-Dusseau and D.Patterson, "Microbenchmark-based Extraction of Local and Global Disk Characteristics," Technical Report CSD-99-1063, 1999.
15) M.K.McKusick et al., "A Fast File System for UNIX," ACM Transactions on Computing Systems (TOCS), 2(3), 1984.
16) R.Card and T.Ts'o and S.Tweedle, "Design and Implementation of the Second Extended Filesystem," First Dutch International Symposium on Linux, 1994.
17) Stephen Tweedie, "Journaling the Linux ext2fs Filesystem," LinuxExpo, 1998.
18) Greg Ganger and Frans Kaashoek, "Embedded Inodes and Explicit Groups: Exploiting Disk Bandwidth for Small Files, "USENIX Annual Technical Conference, 1-17. 1997.
19) M.Rosenblum and J. Ousterhout, "The Design and Implementation of a Log-Structured File System," ACM Transactions on Computer Systems, 26-52, 1992.
20) J.Wang and Y.Hu, "PROFS.Performance-Oriented Data Reorganization for Log-structured File System on Multi-Zone Disks," The 9th International Symposium on Modeling, Analysis and Simulation on Computer and Telecommunication Systems, 285.293,
21) HFS Plus Volume Format, http://developer.apple.com/technotes/tn/tn1150.html
22) C.Staelin and H.Garcia-Molina, "Smart Filesystems," USENIX Winter, 45-52, 1991. 2001.