Vol.2017-OS-139 No.15
2017/3/2

不規則なIOアクセス集中に適用可能な高速メモリと低速フラッシュストレージを用いた階層ストレージシステムの提案

大江 和一[1,a)]　佐藤 充[1]　南里　豪志[2]

Abstract: 近年，複数の Solid State Drive (SSD) から構成される All-flash-array (AFA) システムが急速に普及してきている．しかしながら，Virtual Desktop Infrastructure (VDI)，in-memory database system などの一部のアプリケーションは，AFA システムより短い応答時間を必要としている．これらのワークロードを調査したところ，我々はストレージボリュームの数パーセントの範囲に全体の IO アクセスの半分以上が集まる IO アクセス集中が発生することを確認した．この IO アクセス集中はストレージボリュームの任意の位置に短時間継続する特徴を有し，さらにブロックレベルの観点で分析すると，規則性が低く LRU や FIFO などの従来のキャッシュ置換アルゴリズムでは捉えることが難しいことも分かった．AFA よりも短い応答時間を達成する目的で，本稿は SSD や AFA とメモリを用いた Hybrid storage system である automated tiered storage with fast memory and slow flash storage (ATSMF) 提案を行う．ATSMF は，ワークロードをリアルタイムに分析することで IO アクセス集中の発生を把握し，データ置換時の応答時間増加量とデータ置換後の応答時間削減量を比較し，削減量が上回る IO アクセス集中のみをメモリに置換することで平均応答時間の削減を達成した．ATSMF の評価では，フラッシュストレージ単体より 20%以上平均応答時間が削減され，メモリ領域のアクセス比が 50%以上となることを確かめた．

1. はじめに

近年，複数の NVMe[2]Solid State Drive (SSD) から構成される All-flash-array (AFA) システム [1] が急速に普及してきている．これらのシステムの中には 100 micro seconds 未満の平均応答時間を達成しているシステムも存在する．しかしながら，Virtual Desktop Infrastructure (VDI)[3]，in-memory database system [4] などの一部のアプリケーションは，AFA システムの平均応答時間より短い応答時間を必要としている．VDI は，boot storm[5] の様に短時間で急激に IO アクセス数が増加するケースが想定される．この様なケースにおいても出来るだけ IO アクセスの応答時間を悪化させないようにする目的で，VDI はより平均応答時間が小さいストレージシステムを必要としている．In-memory database system に関しては，我々は短い応答時間が求められる Online Transaction Processing (OLTP)[6], [7] で AFA より平均応答時間が小さいストレージシステムを必要としていることを把握した．これらの OLTP は，トランザクション当たりの平均応答時間を micro second order にするために，全てのテーブルが in memory に置かている．さらにこれらの OLTP は，ログやチェックポイント情報をストレージシステムに書き出す処理を併せて行う必要がある．従って，OLTP の平均応答時間をより小さくするには，ストレージシステムの write 応答時間を小さくすることが重要である．

そこで我々は，AFA システムや NVMe SSD より短い平均応答時間を必要とするアプリケーションのワークロードの調査を行い，空間的・時間的に連続した IO アクセス集中が発生していることを把握した．この IO アクセス集中は，ストレージボリュームの数%未満のごく狭い領域に最大 1 時間程度継続し，全 IO アクセスの少なくとも半分が含まれる．IO アクセス集中は，同じ領域に継続するケースと数分単位で隣接領域に移動するケースに分類され，大部分がストレージボリュームの何処に発生するのかを予測できない．さらに我々は，IO アクセス集中を含むワークロードを page レベルの観点で分析を行い，同一 page へ繰り返し IO が発生する割合が小さいことも把握した．この分析結果は，page レベルの規則性を利用してデータ置換を行う Caching [8], [9] を適用しても，十分な効果を上げられないことを意味する．

我々の以前の成果である on-the-fly automated storage tiering (OTF-AST) [10], [11], [12] は，SSD と HDD を対象にした Hybrid storage system である．OTF-AST は，HDD 上で発生した IO アクセス集中をリアルタイムに捉え，データ置換時の IO アクセスの応答時間増加よりデータ置換後の IO アクセスの応答時間削減が上回るかどうかを判断し，データ置換後の削減が上回る場合のみ IO アクセス集中が発生した範囲の HDD データを全て SSD に置換する．しかしながら OTF-AST は，シークや回転待ちの頻度に応じて性能が大きく変動する HDD を用いていたためデータ置換時の IO アクセス応答時間の増加量を予測することが難しく，ワークロードごとに予め求めておいた最大増加量を用いて置換の可否を

[1] (株) 富士通研究所
FUJITSU LABORATORIES LTD.
[2] 九州大学
KYUSHU UNIVERSITY.
[a)] ooe.kazuichi@jp.fujitsu.com

判断した．そのため，OTF-AST は特に IO アクセス集中が発生する領域が頻繁に移動するワークロードでは，IO アクセス集中を効果的に SSD へ置換することが出来ず，十分な効果を上げることが出来なかった．

本稿では，SSD や AFA とメモリを対象にした Hybrid storage system である automated tiered storage with fast memory and slow flash storage (ATSMF) 提案を行う．ATSMF は，アクセス時に物理的な動作を伴わずデータ置換時の IO アクセスの応答時間悪化量予測が容易な SSD とメモリを用いる．そこで ATSMF は，運用中にデータ置換時の IO アクセス応答時間悪化量とデータ置換後の IO アクセス応答時間削減量を常にモニタしておく．そして ATSMF は，検出した IO アクセス集中の継続時間を予測し，前述したモニタ値を用いてデータ置換時の IO アクセス応答時間悪化量がデータ置換後の IO アクセス応答時間削減量を下回るかど うかを判断し，下回る場合のみ IO アクセス集中が発生したデータ領域をメモリに置換する．ATSMF の評価を行ったところ，フラッシュストレージのみで動かしたときより少なくとも 20%平均応答時間が削減され，メモリ領域のアクセス比が 50% 以上となることが確かめられた．page レベルで置換判定を行うキャッシュとの比較においても，平均応答時間・メモリ領域のアクセス比共にキャッシュを大幅に上回る結果となり，ATSMF が提案する置換方法の有用性が実証された．

本研究の貢献は以下である．

- VDI や OLTP などのワークロードを分析し，IO アクセス集中が発生することを明らかにしたこと．さらにこれらワークロードは page レベルでの規則性が小さく，page レベルでデータ入れ替えを行う caching を用いても効果が期待できないことを明らかにしたこと．
- 上記ワークロードを効果的に扱うメモリと AFA/SSD を用いた hybrid storage system として，ATSMF を提案したこと．ATSMF は，メモリに置換することで平均応答時間削減に効果を期待できる IO アクセス集中のみを選択し，即座にメモリへ置換を行う．
- ATSMF の評価を行い，大部分のケースで SSD や caching より平均応答時間が小さくなることを示したこと．さらに，メモリアクセス率に関して caching と比較し，大部分のケースで圧倒的に上回ることを示したこと．

2. フラッシュストレージとメモリの定義

2.1 フラッシュストレージの定義

本稿でのフラッシュストレージは，商用の AFA[1] に加えて NVMe SSD[13] も含む．これらフラッシュストレージの大半は，100 micro seconds を下回るが 1 桁の micro second オーダーには遥かに及ばない平均応答時間である．我々は intel P3700 PCIe SSD [13] を用いて評価を行った．本稿の評価における P3700 の平均応答時間は，40 から 80 micro seconds であった．read が中心のワークロードを実行したときに 40 micro seconds 前後の平均応答時間となり，read-write が混在したワークロードを実行したときに 80 micro seconds 前後の平均応答時間であった．本稿では，フラッシュストレージのことをフラッシュもしくは SSD と呼ぶ．

2.2 メモリの定義

本稿でのメモリは，SSD よりも平均応答時間が十分に小さく，不揮発であることが求められる．近年，non-volatile memory (NVM) が複数のベンダーで開発が進められている．AGIGARAM[14] は，Dual-Inline Memory Module (DIMM) 上に DRAM とフラッシュの両方が実装され，DIMM の電源が遮断されたときに DRAM の内容をフラッシュにバックアップすることで不揮発を実現している．AGIGARAM の平均応答時間は，DRAM と同等であり SSD と比較して十分に小さい．

intel 3D-Xpoint[15] は，不揮発なメモリ素子を用いた開発中の NVM である．Subramanya[29] らの報告によると，intel 3D-Xpoint のメモリアクセスレイテンシは DRAM の 2 倍から 4 倍の範囲である．DRAM の CAS latency (read response time)[17] が 15 nano-seconds であるので，intel 3D-Xpoint のメモリアクセスレイテンシは SSD と比較して十分に小さいと推定される．

なお，AGIGARAM は入手が容易ではなくメモリアクセス性能が DRAM と同等であるので，本稿の評価は DRAM をメモリとして使用した．

3. IO アクセス集中の特徴

3.1 用語の定義

本稿で使用する用語の定義を行う．まず，OS から認識出来るストレージの論理的な単位を Logical Unit Number(LUN) と定義する．LUN を 1GB 前後の適当な大きさで区切った部分を sub-LUN と定義する．sub-LUN は複数の block から構成されており，block の大きさは通常 512 bytes である．この各 block ごとに Logical Unit Address (LBA) が割り当てられ，LUN の各部分への IO アクセスはこの LBA を指定することで行われる．また，本稿のワークロードとは，時間経過ごとに LUN を構成する各 block に発生する IO アクセスのことである．

3.2 分析を行ったワークロード

本稿では，IO アクセス集中の特徴を抽出する目的で，VDI, DBT-2, 共有ファイルサーバ，web サーバ，及び mail サーバのワークロードを分析した．

VDI は社内の研究開発部門で運用中のシステムからワークロードを採取した [18]．この VDI は約 300 ユーザが常時アクセスし，CITRIX Xen- Desktop[19] で運用されている．

DBT-2[20] は OLTP transactional performance test であり，TPC-C[21] 相当のテストが行われる．本稿では，DBT-2 実行時に採取したワークロードを分析した．DBT-2 実行時に使用したパラメータは，warehouse 数=500，rampup time=50 分，execution time=30 分，connection 数=10，database size=50 GB，ボリュームサイズ=800 GB である．

共有ファイルサーバ，web サーバ，及び mail サーバのワークロードは，以前に分析を行った Samba[22]，MSR Cambridge [23], [24]，Microsoft Exchange Server (MS EX)[18] の結果を引用した．

3.3 分析方法

我々の分析の目的は，IO アクセスの空間的・時間的偏りを巨視的な観点と微視的な観点の両方で明確にすることである．最初に巨視的な観点での分析方法を説明する．巨視的な観点での分析は，3.2 節のワークロードを 1 GB sub-LUN 単位で分割し，各 sub-LUN の IO アクセス数を 1 分間隔で集計したデータを用いて行った．

微視的な観点の分析は，3.2 節のワークロードを Facebook

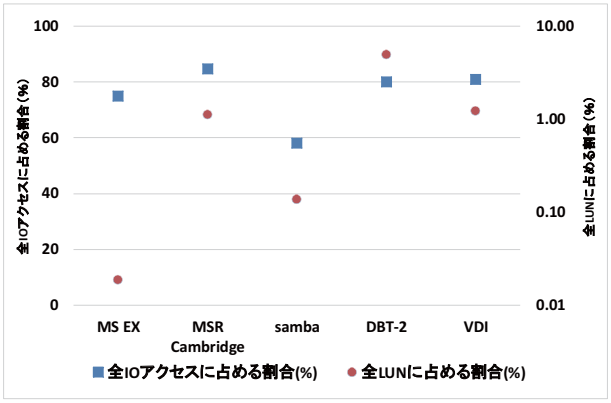

Fig. 1 IO アクセス集中の巨視的分析結果

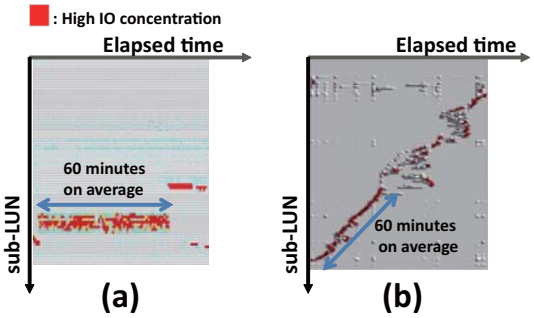

Fig. 2 IO アクセス集中の例

FlashCache[8] 上で再現し，その時のキャッシュヒット率を比較することで行った．FlashCache は，4 KB page 単位でデータ入れ替えを行い，デフォルトの置換アルゴリズムは FIFO である．FlashCache 評価に用いたシステム構成やメモリのサイズは 5.2 節と同一である．

3.4 IO アクセス集中の特徴

図 1 は，IO アクセス集中に含まれる IO アクセス数が全 IO アクセス数に占める割合と IO アクセス集中となった領域の全 LUN に占める割合を示したものである．この分析結果より，最大でも全 LUN の数%の範囲に全 IO アクセスの 60%以上が集中していることが分かる．なお MSR Cambridge は，分析を行った src1_0, src1_1, proj1, proj2, proj4, usr1, usr2, web2 ワークロードの平均値である．

表 1 は，IO アクセス集中の平均継続時間，任意の LBA に発生した割合，Write 比，Caching に適用した場合のヒット率である．この分析結果より，IO アクセス集中は 3 分から最大 87 分継続することが分かる．図 2 は，IO アクセス集中がどの様に継続するのかを抜粋した図である．図の様に，同じ sub-LUN に継続するケース (a) と隣接する sub-LUN に短時間で移動しながら継続するケース (b) にわかれる．(a) は samba や VDI に多く見られ，(b) は MSR Cambridge や MS EX に多く見られた．IO アクセス集中が任意の LBA に発生した割合は 66%以上であり，IO アクセス集中の発生を予測することが困難であることが分かる．なお，任意の LBA に発生した割合の計算方法は文献 [18] と同様の方法を用いた．

最後に Caching ヒット率に関して説明する．表 1 の Caching ヒット率は，ワークロードごとにバラバラであることが分かる．Caching の置換アルゴリズムは page レベルの規則性を利用しており，各ワークロードに含まれる page レベルの規則性が一意ではないため，このような結果となった．よって，IO アクセス集中を含むワークロードに Caching を適用しても，そのワークロードに含まれる page レベルの規則性に応じて Caching の効果が増減することが分かる．

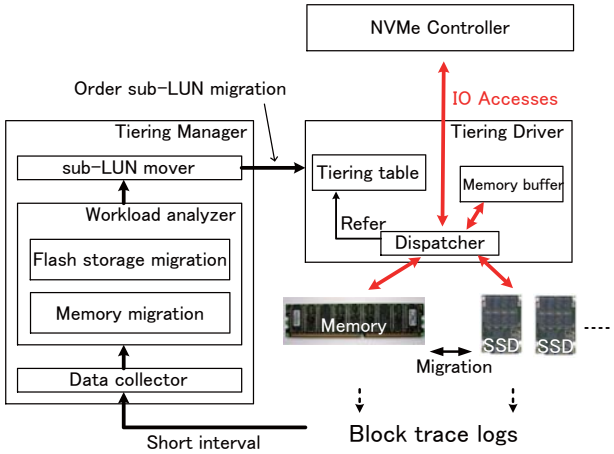

Fig. 3 Block diagram of ATSMF.

4. ATSMF

4.1 概要

3 章の分析結果を受け，我々は IO アクセス集中を適切に扱うシステムとして'automated tiered storage with fast memory and slow flash storage (ATSMF)' の提案を行う．

ATSMF は，LUN を sub-LUN の単位に分割し，sub-LUN 単位で IO アクセス集中の発生を監視する．ATSMF が IO アクセス集中となった sub-LUN 群を検出すると，ATSMF は検出した IO アクセス集中の継続時間を予測し，予測した継続時間を用いてデータ置換時の IO アクセス応答時間増加量とデータ置換後の IO アクセス応答時間削減量を比較する．もしデータ置換後の IO アクセス応答時間削減量が上回る場合は，ATSMF はこれらの sub-LUN を即座に SSD からメモリに置換する．同時に ATSMF は，この IO アクセス集中が時間経過と共に隣接する sub-LUN に移動するかどうかをチェックする．もし IO アクセス集中が隣接 sub-LUN に移動する場合，ATSMF はその移動速度を測定し，近い将来に IO アクセス集中が移動する sub-LUN を計算し，そしてそれら sub-LUN を即座に SSD からメモリに置換する．また，ATSMF は IO アクセス集中が終息した sub-LUN を検出すると，それら sub-LUN をメモリから SSD へ即座に置換する．

図 3 は ATSMF のブロック図である．ATSMF は tiering manager と tiering driver から構成される．tiering manager は data collector, workload analyzer, sub-LUN mover から構成される．data collector は短い時間間隔で block trace を実行し，block trace の log を収集し，それら log から sub-LUN 単位の IO アクセス数を計算して保存する．workload analyzer は，短い時間間隔で保存した sub-LUN 単位の IO アクセス数情報を取り出し，前節で説明したメモリへの置換と SSD への置換を実行する．本稿では，4.2 節と 4.3 節にてメモリへの置換と SSD への置換方法を詳細に説明する．workload analyzer の出力は，メモリと SSD 間で置換が必要な sub-LUN の情報である．sub-LUN mover は，この workload analyzer の出力を用いて tiering driver に sub-LUN の置換を指示する．

tiering driver は，tiering table，dispatcher，memory buffer から構成される．tiering table は，メモリに置換した sub-LUN

Table 1 IO アクセス集中の分析結果

Workloads	samba	proj1	proj2	proj4	src1_0	src1_1	usr1	usr2	web2	MS EX	DBT-2	VDI
平均継続時間 (分)	30	60	54	40	87	66	86	32	58	20	—	3
任意の LBA に発生した割合 (%)	71	75	88	75	90	89	66	84	100	92	—	80
Write 比 (%)	70	11	12	2	44	5	9	19	1	50	100	20
Caching ヒット率 (%)	51.81	0.01	10.46	11.55	10.56	31.11	98.42	0.01	0.00	—	71.20	0.01

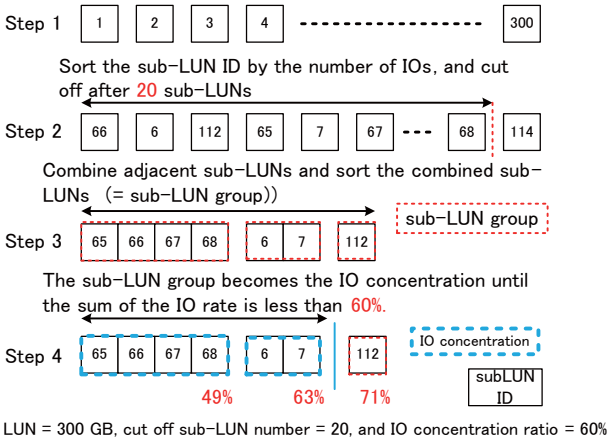

Fig. 4 How to choose IO concentration sub-LUNs.

情報を管理する．dispatcher は，tiering table を参照することで IO アクセスをメモリ，SSD，又は memory buffer に振り分ける．dispatcher は，置換中の sub-LUN に入る LBA の IO アクセスのデータを memory buffer に書き込む．さらに，dispatcher は memory buffer に書き込んだ領域と一致する read IO アクセスは，memory buffer から読み込む．sub-LUN 置換が完了すると，dispatcher は memory buffer に書き込んだ全ての領域とメモリもしくは SSD を同期する．

4.2 メモリへの置換

workload analyzer は，data collector から保存してある sub-LUN 単位の IO アクセス数情報を取り出し，その情報を用いてメモリへの置換を実行する．最初に，メモリへの置換は IO アクセス集中が発生した sub-LUN を抽出する．図 4 は，300 GB の LUN で sub-LUN の抽出方法を示している．Step 1 では，メモリへの置換は sub-LUN 単位の IO アクセス数を抽出する．Step 2 では，メモリへの置換は IO アクセス数で sub-LUN ID をソートし，N 以上となった sub-LUN をカットする．3 章の分析結果より，大部分の IO アクセス集中は LUN の数%の範囲に発生する．それ故，N は全 sub-LUN の数の数%となるように設定する．図 4 の N は 20 である．Step 3 では，メモリへの置換は隣接した sub-LUN を結合し，結合した sub-LUN の IO アクセス数を合計し，IO アクセス数順に結合した sub-LUN をソートする．本稿では，この結合した sub-LUN のことを sub-LUN group と呼ぶ．Step 4 では，メモリへの置換は IO アクセス数順に sub-LUN group を選択し，選択した sub-LUN group の IO アクセス数を合計し，合計した IO アクセス数と全 IO アクセス数のとの IO アクセス比を求める．もし IO アクセス比が M 以上なら，メモリへの置換は sub-LUN group の選択を中止する．選択した sub-LUN group の集合が新たな IO アクセス集中である．3 章の分析結果より，大部分の IO アクセス集中は全 IO アクセスの半分以上を含む．従って，M は 50 以上に設定する．図 4 の M は 60 である．

次に，メモリへの置換は前節で抽出した IO アクセス集中をメモリに置換するかどうかを判断する．そこで，メモリへの置換は IO アクセス集中の継続時間を測定する．図 5 は，IO アクセス集中の継続時間測定方法を示している．各 period は図 4 の結果抽出された IO アクセス集中であり，メモリへの置換は各 IO アクセス集中に対して period 数をカウントする．例えば，IO conc.-1 は，period 1 では 1 period 継続，period 2 では 2 period 継続したことになる．なお period とは，図 3 で説明した data collector が block trace を実行する間隔である．本稿の例では，block trace を 20 秒実行し後処理に 4 秒使っているので，24 秒間隔で period が進む．さらに，メモリへの置換は表 2 を用いて IO アクセス集中の残り継続時間を推定する．もし IO アクセス集中が 2 period 継続したとすると，残り継続時間は 32.88 秒である．もし period 数が増加すると，残り継続時間もまた増加する．この理由は IO アクセス集中の大部分は短時間しか継続しないためである．メモリへの置換は，ストレージシステム運用時に取得した直近のワークロードデータを用いて各 IO アクセス集中の継続時間を測定し，その測定データを用いて表 2 を生成，更新する．ここで表の生成方法を説明する．最初に我々は，半日等適当な期間で各 IO アクセス集中の継続時間を収集する．次に我々は，収集したデータから A period 以上継続した IO アクセス集中を取り出し，平均継続時間を計算する．最後に我々は，平均応答時間から A を引くことで残り継続時間を得る．例えば，A が 2 又は 48 秒の時，残り継続時間は 32.88 秒である．

次に，メモリへの置換は置換後の応答時間削減が置換時の応答時間増加を上回るかどうかを判断する．図 6 は，メモリへの置換を行うかどうかを判断する方法を示している．Y はメモリへの置換後の応答時間削減量であり，X はメモリへの置換時の応答時間増加量である．X と Y を計算する目的で，メモリへの置換は tiering driver から $Avef$, $AvefWithMg$, $Avem$, and $AveMgTime$ の値を計算時に回収する．$Avef$ は，SSD 側の平均応答時間である．$AvefWithMg$ は，データ置換実行中の SSD 側の平均応答時間である．$Avem$ は，メモリ側の平均応答時間である．$AveMgTime$ は，sub-LUN を SSD とメモリ間で置換するのにかかる平均時間である．tiering driver は，各 IO アクセスの応答時間と各データ置換にかかった時間を集計し，この集計した値を用いてこれらパラメータを計算する．よって，メモリへの置換は，以下の式で計算することが出来る．これらの式は，予測した期間に発生する IO アクセス数が変化しないことを前提にしている．

$$Xduration = AveMgTime * subLUNs \quad (1)$$

$$Yduration = Pduration - Xduration \quad (2)$$

$$X = (AvefWithMg - Avef) * Xduration \quad (3)$$

$$Y = (Avef - Avem) * Yduration \quad (4)$$

$subLUNs$ は IO アクセス集中を含む sub-LUN 数である．図

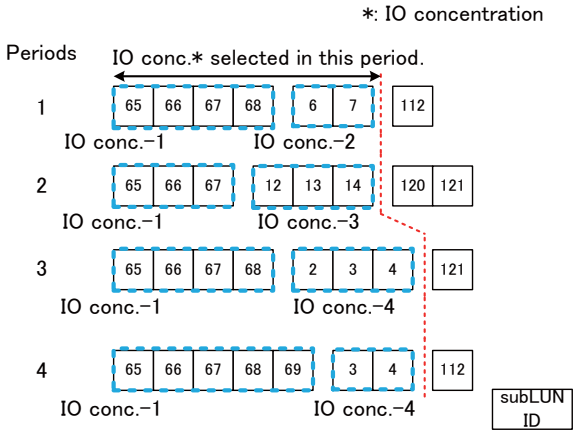

Fig. 5 How to measure the duration of IO concentrations.

Table 2 Prediction table for the duration of IO concentration sub-LUNs (example).

Periods (A)	average duration (B) (periods)	rest duration (B-A) (periods)	rest duration (B-A) (seconds)
1	1.73	0.73	17.58
2	3.37	1.37	32.88
3	4.83	1.83	43.90
4	6.37	2.37	56.89
5	8.00	3.00	72.00

1 period = 24 second (block trace の実行時間=20 秒，後処理の実行時間=4 秒).

5 の Period 2, IO conc.-1 だと，*subLUNs* は 3 になる．*Pduration* は，表 2 から求めた残り継続時間である．*Xduration* はデータ置換に必要な時間であり，*Yduration* はデータ置換後の IO アクセス集中継続時間を示している．*X* はデータ置換にかかる時間とデータ置換に伴う応答時間の増加量を掛け合わせた値である．*Y* はデータ置換後の IO アクセス集中継続時間とデータ置換後の応答時間の減少量を掛け合わせた値である．もし *Y* が *X* を上回ると，メモリへの置換はその IO アクセス集中の置換を実行する．もし *Y* が *X* を上回らないと，メモリへの置換はその IO アクセス集中の置換を保留する．もしその保留した IO アクセス集中が次の period も継続していたら，メモリへの置換は再度その IO アクセス集中を置換すべきかどうかを判断する．

図 2 は IO アクセス集中の例を示している．IO アクセス集中 (a) はほぼ同じ複数の sub-LUN 上に発生しているが，IO アクセス集中 (b) は同一 sub-LUN に数分継続し，隣接する sub-LUN に移動する．(b) の IO アクセス集中の平均応答時間を削減する目的で，メモリへの置換は data collector が蓄積した情報を用いて直近の IO アクセス集中の *moving speed* を常に計算しておく．もしメモリへの置換が IO アクセス集中の置換を決めると，メモリへの置換は *moving speed* と表 2 の残り継続時間を使って近い将来 IO アクセス集中が移動する sub-LUN を計算する．そして，それら sub-LUN も同時にメモリに置換する．

4.3 SSD への置換

SSD への置換は，IO アクセス集中が既に終息した sub-LUN をメモリから SSD へ置換する．IO アクセスの応答時間の増加を避ける目的で，SSD への置換は *AvefWithMg* の応答時間が予め定めておいた閾値を超えると SSD への置換を保留する．

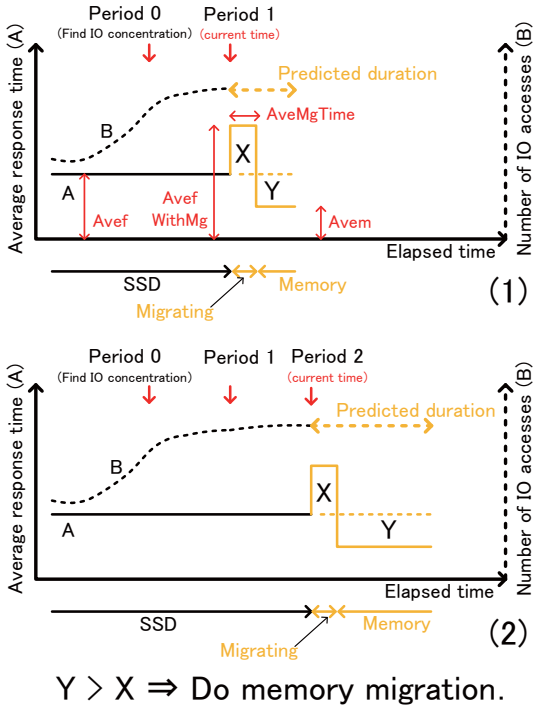

Fig. 6 How to judge whether memory migration is done or not

Table 3 Specifications of experimental system.

PC	Fujitsu PRIMERGY TX300S8, Intel Xeon E5-2660 x2, mem= 256 GB, CentOS 7.1 (64 bits)
SSD	Intel P3700 (800 GB, PCIe attached)

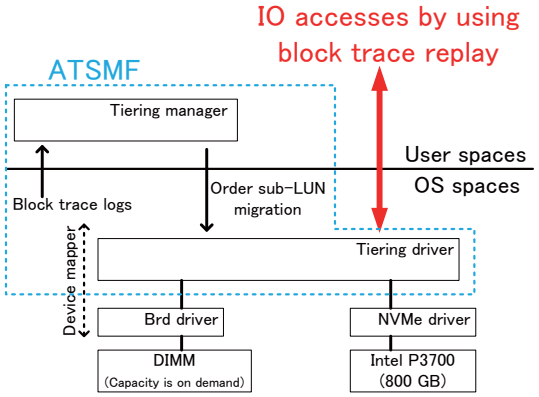

Fig. 7 Experimental system.

5. 評価

5.1 評価システムの概要

図 7 は本稿の実験システムである．本稿の実験システムは，CentOS 7.1 を前提にしており，block trace log は Linux blktrace command を用いて採取する．tiering driver は，Linux device mapper framework [25] 上に実装した．メモリデバイス層は Linux brd driver[26] を用いた．表 3 は本稿の実験システムの仕様である．この実験システムの sub-LUN サイズは 1 GB である．tiering manager の実行間隔は，blktrace を 20 秒実行し，その後処理を 4 秒実行したため，24 秒である．

5.2 評価方法

本稿の評価の目的は，ATSMF が IO アクセス集中を大量に

Table 4 Workloads used in experiments.

Workload	GB	Comments
vdi (VDI)	800	
dbt2 (OLTP)	800	
src1_0 (MSR)	293	retrieved between 1920 and 2020 (No.1)
src1_1 (MSR)	293	retrieved between 480 and 600 (No.1)
proj1 (MSR)	800	retrieved between 230 and 550 (No.1)
proj2 (MSR)	800	retrieved between 6760 and 6890 (No.1)
proj4 (MSR)	236	retrieved between 7390 and 7600 (No.1)
usr1 (MSR)	800	retrieved between 2090 and 2350 (No.1)
usr2 (MSR)	569	retrieved between 5970 and 6130 (No.1)
web2 (MSR)	182	retrieved between 5870 and 5930 (No.1)
src1_0+ (samba)	293	created on src1_0 basis
src1_1+ (samba)	293	created on src1_1 basis

No.1: Elapsed time (minutes) from first trace logs.

含むワークロードを適切に処理できるかどうかを明確にすることである．そこで本稿では，平均応答時間とメモリアクセス比の観点で ATSMF の性能を評価した．平均応答時間はデータ置換に伴うオーバーヘッドを含んだケースでの ATSMF の効果を判断でき，メモリアクセス比はオーバーヘッド分を除いた ATSMF の効果を判断できる．本稿では ATSMF の性能を SSD のみの構成ととメモリと SSD の caching 構成で比較した．評価に用いたベンチマークは，3.4 章で紹介した VDI，DBT-2，及び複数の共有ファイルサーバのワークロードの replay である．この複数のファイルサーバとは，MSR Cambridge [23], [24] と Samba [22] である．さらに本稿では，caching の実装として Facebook FlashCache [8] を用いた．FlashCache のデフォルトの置換アルゴリズムは FIFO であり，デフォルトのブロックサイズは 512 bytes である．本稿では，ブロックサイズを 4 KB に設定した以外は，デフォルトの設定を用いた．

VDI ワークロード [3] は，約 1 週間分のトレースログから構成され，採取したワークロードのボリュームサイズは 64 TB である．このワークロードを replay するために，本稿では平日夕方の 1 時間分のログを取り出した．この時間帯は多くの IO アクセスが発生する．さらに，本稿で用いた実験システムの SSD サイズが 800 GB であるので，64 TB の中から 800 GB 分を取り出した．この 800 GB の領域は，64 TB ボリュームの中で最も IO アクセスが集中していた領域である．MSR Cambridge は，src1_0, src1_1, proj1, proj2, proj4, usr1, usr2, web2 ワークロードの一部分を選択して replay した．その理由は，これらワークロードのトレースログは約 1 週間分あり，IO アクセス集中が発生していたタイムスロットを選んだためである．Samba は，ワークロードのトレースログが 1-GB 1-分単位の統計情報の集合体であるため，そのままでは block レベルで IO アクセスを再現出来ない．そこで本稿では，MSR Cambridge src1_0, src1_1 ワークロードを用いて，Samba 相当のトレースログを生成した．このトレースログの生成方法は，文献 [10], [12] に説明がある．表 4 は，本稿の評価で用いたワークロードの詳細情報である．表 4 の GB は，各ワークロードの LUN の大きさを示している．MSR の Comments は 1 週間分のトレースログの中から replay 用に切り出した範囲を示しいる．

本稿では，表 2 で例示した prediction table を表 4 の各ワークロードのトレースログを用いて生成した．我々の過去のこれらワークロードのトレースログ分析結果 [18] より，これらワークロード内で発生する IO アクセス集中の継続時間の分布はワークロード全区間でほぼ同じである．prediction table

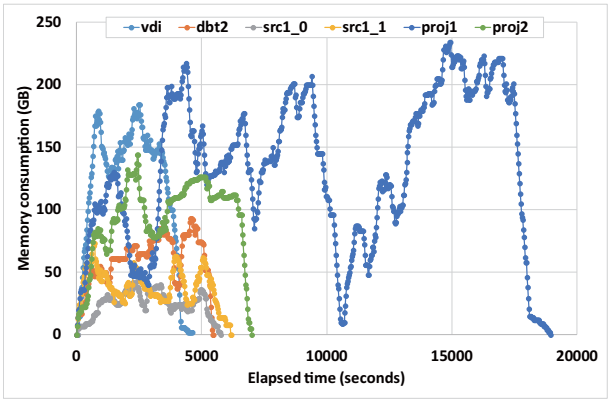

Fig. 8 Memory consumption of ATSMF (1).

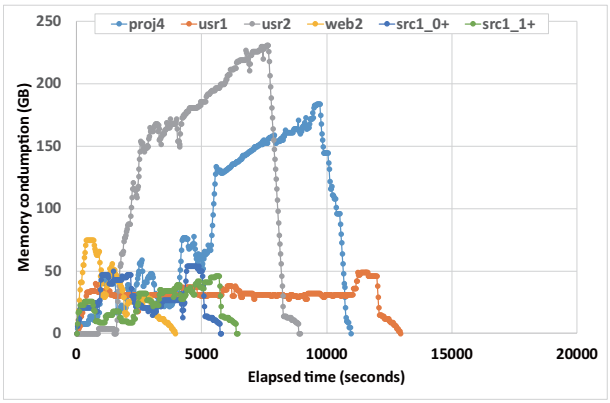

Fig. 9 Memory consumption of ATSMF (2).

は ATSMF を運用する直前の IO アクセス集中の継続時間分布に基づいて生成する必要があるが，過去の分析結果 [18] より，replay するトレースログの IO アクセス集中の継続時間分布と replay する直前の分布はほぼ同じと判断できる．

図 4 の cut off する sub-LUN 数は 30 であり，IO アクセス集中比は 60%とした．各トレースログは，Linux btreplay command を用いて実行する．オプションは，*-X 1 -W 1* を指定する．

5.3 評価結果

表 5 は，intel P3700 (SSD)，FlashCache，ATSMF の平均応答時間と intel P3700 の平均応答時間に対する FlashCache と ATSMF の平均応答時間比を示している．表 6 は，各ワークロードごとの LUN の大きさ (GB)，write 比 (%)，FlashCache を適用した場合のメモリアクセス比 (%)，ATSMF を適用した場合のメモリアクセス比 (%)，ATSMF を適用した場合の最大メモリ消費量を示している．この最大メモリ消費量は，図 8，9 の最大値であり，パーセント指標は LUN の大きさ対する割合である．

ATSMF は 4.2 節で説明した方法で IO アクセス集中を検出した時にメモリを割り当て，4.3 節で説明した IO アクセス集中の終息を検出するとメモリを解放する．従って，適用するワークロードごとにメモリの消費量が異なる．図 8，9 は，時間経過と共に ATSMF のメモリ消費量がどの様に変化するのかを示している．そこで，FlashCache のメモリ容量は，各ワークロードごとに ATSMF のメモリ消費量の最大値になるように設定した．例えば，VDI の場合は，FlashCache のメモリ容量は 184 GB である．

Table 5 Average response time (milli seconds).

Workloads	Intel P3700 (A)	FlashCache (B)	B/A	ATSMF (C)	C/A
vdi	222.3463	290.2623	1.31	177.3156	0.80
dbt2	0.0610	1.6662	27.32	0.0344	0.56
src1_0	0.0695	2.8704	41.29	0.0254	0.37
src1_1	0.0419	0.2009	4.80	0.0250	0.60
proj1	0.0419	3.1369	74.82	0.0524	1.25
proj2	0.0527	2.7640	52.47	0.0309	0.59
proj4	0.0458	0.9575	20.92	0.0300	0.65
usr1	0.0401	0.0169	0.42	0.0150	0.38
usr2	0.0413	1.6239	39.32	0.0373	0.90
web2	0.0620	1.0250	16.53	0.0427	0.69
src1_0+	0.647	3.3216	51.36	0.0207	0.32
src1_1+	0.0475	0.1425	3.00	0.0185	0.39

Table 6 Evaluation results.

Workloads	Capacity (GB)	Write ratio (%)	memory access ratio (FlashCache) (%)	memory access ratio (ATSMF) (%)	Maximum memory consumption (GB)	Maximum memory consumption (%)
vdi	800	22.98	0.01	49.80	184	23.00
dbt2	800	100.00	71.20	89.33	93	11.63
src1_0	293	84.24	10.56	82.66	43	14.68
src1_1	293	0.91	31.11	86.43	74	25.26
proj1	800	1.16	0.01	74.71	220	27.50
proj2	800	42.49	10.46	82.77	144	18.00
proj4	236	0.22	11.55	86.55	184	77.97
usr1	800	0.25	98.42	85.43	49	6.13
usr2	569	0.22	0.01	84.25	231	40.60
web2	182	0.03	0.00	81.65	75	41.21
src1_0+	293	82.84	51.81	88.44	54	18.43
src1_1+	293	0.87	63.45	87.90	46	15.70

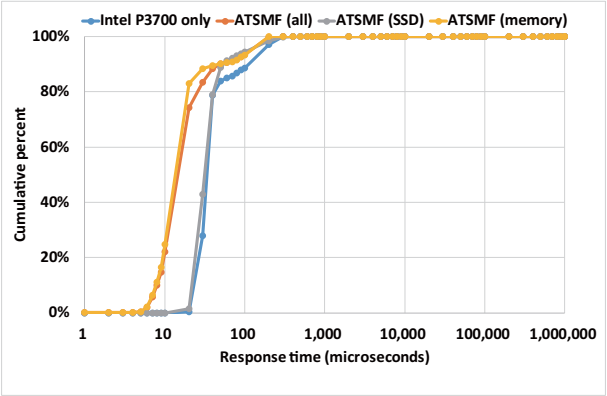

Fig. 10 CDFs of DBT2 response time (1).

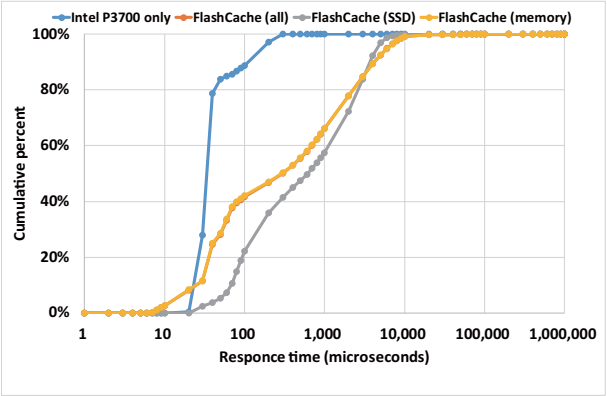

Fig. 11 CDFs of DBT2 response time (2).

5.3.1 VDI

SSD のみ (intel P3700) の平均応答時間が 222 milli second に達しており，このワークロードは本稿の実験システムに対してオーバーロードであった．しかしながら，ATSMF の平均応答時間は SSD のみより 20%低く，且つメモリアクセス比は 49.80%である．

一方，FlashCache のメモリアクセス比は 0.01%に過ぎず，さらにワークロードの write 比が 22.98%であるので，FlashCache 運用は write-back 負荷を伴うことになる．それ故，FlashCache の平均応答時間が SSD のみより 31%も大きくなったと我々は判断した．

さらに，FlashCache の低メモリアクセス比は，FlashCache の置換アルゴリズムが VDI の IO アクセス集中を適切に処理出来ないことを意味している．言い替えると，VDI の IO アクセス集中は，page レベルの規則性が極めて小さいといえる．

5.3.2 DBT2(OLTP)

ATSMF の平均応答時間は，SSD のみより 44%低く，且つメモリアクセス比は 89.33%である．図 10 は DBT-2 応答時間の cumulative distribution functions (CDFs) である．グラフの読み方は，曲線が左上にいくほど応答時間が小さいことになる．*intel P3700 only* は SSD のみで実行したときの応答時間の推移である．*ATSMF (all)* は，ATSMF の SSD part と memory part を合計した応答時間の推移である．*ATSMF (SSD)* は，ATSMF の SSD part の応答時間推移であり，*ATSMF (memory)* は，ATSMF の memory part の応答時間推移である．*ATSMF (all)* の応答時間推移がほぼ *ATSMF (memory)* と一致することが分かる．これは，メモリアクセス比が 89.33%に達するためである．

一方，FlashCache の平均応答時間は，SSD のみより 27 倍大きく，メモリアクセス比は 71.2%である．図 11 は，FlashCache の DBT-2 応答時間の CDFs である．*FlashCache (SSD)* と *FlashCache (memory)* の両方の大部分の応答時間が SSD のみを下回っていることが分かる．それ故，メモリアクセス比が 71.2%に達するにもかかわらず，FlashCache の平均応答時間が SSD のみの平均応答時間を大幅に上回った．この理由は，FlashCache をメモリと SSD で構成するとなんらかの実装上の問題が顕在化したため，と我々は判断している．

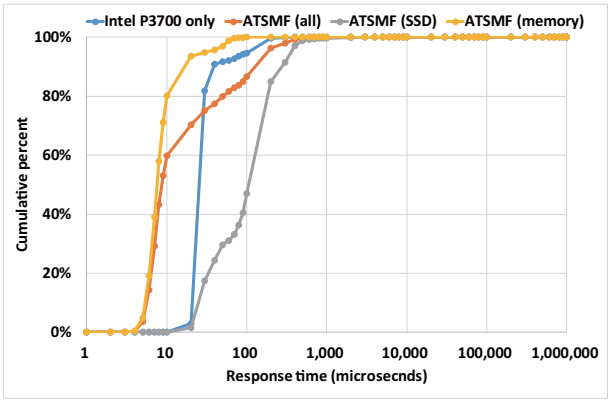

Fig. 12 CDFs of proj1 response time (1).

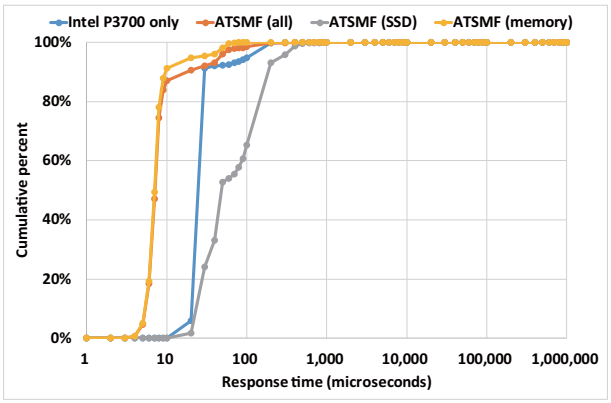

Fig. 13 CDFs of usr1 response time (1).

5.3.3 残りのワークロード

本節では，表 5，6 の src1_0 から src1_1+ワークロードまでの評価結果を議論する．ATSMF の平均応答時間は，proj1 ワークロードを除いた SSD のみと usr1 ワークロードを除いた FlashCache の平均応答時間を上回った．さらに，ATSMF のメモリアクセス比は，usr1 ワークロードの除くと FlashCache のメモリアクセス比を大幅に上回った．この理由に関して我々は，これらのワークロードの IO アクセス集中は，page レベルの規則性が極めて小さいため，と判断している．

図 12 は，proj1 ワークロードの応答時間 CDF である．*ATSMF (all)* は，*ATSMF (SSD)* と *ATSMF (memory)* の中間値となっている．さらに，この図の *ATSMF (SSD)* は，ATSMF でより高頻度にデータ置換が発生したため，図 10 の *ATSMF (SSD)* より右寄りになっている．深青色の線で示された図 8 の proj1 は，proj1 のメモリ消費量が激しく増減したことを示しており，高頻度にデータ置換が発生した証拠である．proj1 ワークロード実行時の平均応答時間を削減するには，4.3 節で示した SSD への置換方法の更新が有効であると我々は考えている．Observational migration [11] は，小さなサイズのデータ置換を実行し，その時の IO アクセス応答時間をチェックすることでデバイスが busy かどうかを把握し，データ置換の可否を判断する方法である．よって，現在の SSD への置換を observational migration に置き換えれば，ATSMF の平均応答時間は SSD のみの平均応答時間より小さくなると判断している．

最後に，本稿では usr1 ワークロードの結果に関して議論する．ATSMF の平均応答時間は FlashCache の平均応答時間とほぼ同じであり，ATSMF のメモリアクセス比は FlashCache より小さいことが分かる．これは，大部分の IO アクセスが全ボリュームの中の約 10 GB の範囲に発生し，FlashCache が大部分の IO アクセスをメモリで捉えるためである．図 13，14 は usr1 応答時間の CDFs である．*ATSMF (SSD)* の推移は *FlashCache (SSD)* より左寄りとなっているが，FlashCache のメモリアクセス比が上回るため，両者の平均応答時間はほぼ同じとなった．

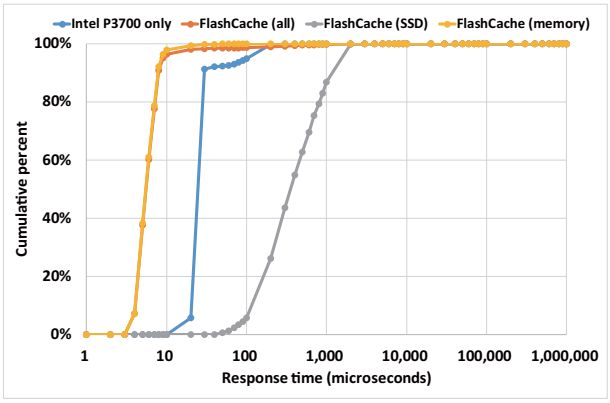

Fig. 14 CDFs of usr1 response time (2).

6. 議論

6.1 ARC 置換アルゴリズムを用いた場合の見積り

Adaptive Replacement Cache (ARC) [27] は，FIFO や LRU などの従来のキャッシュ置換アルゴリズムよりキャッシュヒット率が上回ることが知られている．そこで我々は，ミネソタ大学で開発されたキャッシュシミュレータである sim-ideal [28] を用いて，表 4 の VDI と src1_0 ワークロードのメモリアクセス比の評価を行った．我々は sim-ideal が標準で準備している ARC の機能を用いて評価した．sim-ideal のキャッシュサイズは，表 6 の ATSMF の最大メモリ消費量を上回るように，VDI は 256 GB に src1_0 は 64 GB に設定した．その結果，VDI のメモリアクセス比は約 19%，src1_0 のメモリアクセス比は約 18%となり，いずれも表 6 の ATSMF のメモリアクセス比を大きく下回る結果となった．この結果より，少なくとも VDI と src1_0 ワークロードに関しては，FlashCache の置換アルゴリズムを ARC に入れ替えて実験を行っても，ATSMF を上回らないと判断した．

6.2 ATSMF のメモリ消費量に関する議論

3 章で説明したように，IO アクセス集中は LUN の大きさの数%以下の範囲に発生する．しかしながら，表 6 の Maximum memory consumption (%) の結果より，ATSMF は IO アクセス集中が発生する範囲より遥かに大きなメモリ容量を消費している．これは 4.2 節の最終パラグラフで説明した IO アクセス集中が発生する領域をプリフェッチする機能や IO アクセス集中が終息した sub-LUN をすぐに SSD に戻さない機能が働くためと我々は判断している．これら機能を見直すことで，より少ないメモリ消費量で同等の効果が上げられるよう検討していきたい．

7. 関連研究

最初に NVM を用いた研究報告を紹介する．Subramanya らは [29]NVM と DRAM 間の data tiering を提案している．この研究は，アプリケーションが割り当てるメモリ領域のごく

一部のみが DRAM の速度を必要としていることをまず示している．そこで彼らの提案は，X-Mem profiler を用いて DRAM 性能が必要なメモリ領域を判定し，その結果を用いて提案システムが DRAM 領域と NVM 領域の割り当てを行う．この研究では，NVM を少し遅くて安いメモリとして扱っている．

Jiazin らは [30]High performance file system for non-volatile main memory (HiNFS) を提案している．HiNFS は NVM の遅い write latency を隠蔽する目的で，2 つのアクセスモードを採用している．最初のモードは，NVMM の遅い write latency を隠蔽する目的で lazy-persistent なファイルライトを DRAM にバッファリングし，遅れて DRAM にバッファリングしたデータを NVM に書き込む NVMM-aware な write buffer policy である．もう 1 つのモードでは，HiNFS はすぐにファイルライトが必要なデータを直接 NVMM に書き込む．そして read に関しては，NVMM と DRAM の性能に違いがないため，NVMM と DRAM のどちらかから直接読み込む．

Jian らは [31]NVM と DRAM を用いた hybrid memory を前提に性能と一貫性制御を両立した NOVA ファイルシステムを提案した．この目標を達成するために，NOVA は log-structured ファイルシステムを採用し，頻繁なアクセスが発生する複雑なメタデータ構成部を DRAM に置き，ログデータを NVM に配置した．

Takahiro らは [34]STT-MRAM と DRAM の hybrid memory system (RAMinate) を提案している．STT-MRAM は DRAM 相当の性能が得られるが，書き込み時の消費電力は DRAM より 2 桁大きいと予想されている．そこで RAMinate は，メモリへの書き込みが多いページを DRAM に集めることで，DRAM のみのシステムと比較して 50%の消費電力削減を達成した．

ATSMF の提案の中に prefetching に関する技術が含まれているので，我々は次に prefetching に関する研究報告を紹介する．Gokul らは [32] アプリケーションのトランザクションや query などのコンテキストをキャプチャしそれら情報を活用する context-aware prefetching を提案している．しかしながら，この提案はブロックレベルのアクセスパターンにフォーカスしており，本稿で取り上げた IO アクセス集中を検出することは出来ない．

Mingji らは [33]Table-based Prefetching (TaP) を提案している．TaP は，最適な先読みリードヒット率を実現する目的で，ワークロードごとに適したプリフェッチキャッシュサイズを定義する．しかしながら，この TaP もブロックレベルのシーケンシャルアクセスにフォーカスしており，IO アクセス集中を検出することは出来ない．

8. まとめ

近年，all-flash-array (AFA) システムが急速に普及してきている．これらのシステムの中には 100 micro seconds 未満の平均応答時間を達成しているシステムも存在する．しかしながら，Virtual Desktop Infrastructure (VDI)，in-memory database system などの一部のアプリケーションは AFA システムの平均応答時間より短い応答時間を必要としている．

本論文はこの課題を解決する自動階層制御ストレージシステムソフトウエアに関して記述しており，我々は提案システムを automated tiered storage with fast memory and slow flash storage (ATSMF) と命名した．ATSMF は，アクセス時に物理的な動作を伴わずデータ置換時の IO アクセスの応答時間悪化量予測が容易な SSD とメモリを用いる．そこで ATSMF は，運用中にデータ置換時の IO アクセス応答時間増加量とデータ置換後の IO アクセス応答時間減少量を常にモニタしておく．そして ATSMF は，検出した IO アクセス集中の継続時間を予測し，前述したモニタ値を用いてデータ置換時の IO アクセス応答時間増加量がデータ置換後の IO アクセス応答時間減少量を下回るかどうかを判断し，下回る場合のみ IO アクセス集中が発生したデータ領域をメモリに置換する．

ATSMF の評価を行ったところ，フラッシュストレージのみで動かしたときより少なくとも 20%平均応答時間が削減され，メモリ領域のアクセス比が 50% 以上となることが確かめられた．ブロックレベルで置換判定を行うキャッシュとの比較においても，平均応答時間・メモリ領域のアクセス比共にキャッシュを大幅に上回る結果となり，ATSMF が提案する置換方法の有用性が実証された．

9. 謝辞

sim-ideal のシミュレーション結果を提供頂いた株式会社富士通研究所　コンピュータシステム研究所の五木田研究員に感謝いたします．

References

[1] DELL EMC DSSD D5, https://www.emc.com/collateral/data-sheet/h14828-ds-dssd-d5-rack -scale-flash-appliance.pdf
[2] NVM Express, http://www.nvmexpress.org/
[3] Desktop virtualization, https://en.wikipedia.org/wiki/Desktop_virtualization
[4] List of in-memory databases, https://en.wikipedia.org/wiki/List_of_in-memory_databases
[5] Vijayaraghavan Soundararajan and Jennifer M.Anderson, 'The Impact of Management Operations on the Virtualized Datacenter', Proc. of the ACM IEEE International Symposium Computer Architecture (ISCA'10) (2016.6).
[6] Oracle TimesTen In-Memory Database, http://www.oracle.com/technetwork/database/data-technologies/ timesten/overview/ds-timesten-imdb-129255.pdf
[7] Cristian Diaconu, Craig Freedman, Erik Ismert, and Per-Ake Larson, 'Hekaton: SQL Server's Memory-Optimized OLTP Engine', Proc. of the 2013 ACM SIGMOD/PODS Conference (SIGMOD'13) (2013.6).
[8] Facebook FLACHCACHE, https://github.com/facebook/flashcache
[9] Fusion IO Directcache, http://www.fusionio.com/data-sheets/directcache/
[10] Kazuichi Oe, Satoshi Iwata, Takeo Honda, Motoyuki Kawaba, and Koji Okamura, 'On-The-Fly Automated Storage Tiering (OTF-AST)', Proc. of The Third Asian Conference on Information Systems – Special Session on Information Storage (ACIS-IS 2014), Nha Trang, Viet Nam (2014.12)
[11] Kazuichi Oe, Takeshi Nanri, and Koji Okamura, 'On-The-Fly Automated Storage Tiering with Proactive and Observational Migration', IEICE CPSY, Oita, Japan (SWoPP 2015) (2015.8)
[12] Kazuichi Oe, Satoshi Iwata, Takeshi Nanri, and Koji Okamura, 'Proposal and Evaluation for hybrid storage system which can choose and migrate IO concentration areas to SSD automatically when their duration and number of IO access are long enough to surpass the migration overhead', IPSJ Transactions on Advanced Computing Systems (No.53) (in japanese) (2016.3)
[13] Intel SSD DC P3700 Series, http://www.intel.com/content/www/us/en/solid-state-drives/ ssd-dc-p3700-spec.html
[14] AGIGARAM DDR4 NVDIMM, http://www.agigatech.com/
[15] intel 3D-Xpoint, https://en.wikipedia.org/wiki/3D_XPoint
[16] Subramanya R Dulloor, Amitabha Roy, Zhenguang Zhao, Narayanan Sundaram, Nadathur Satish, Rajesh Sankaran,

Jeff Jackson, and Karsten Schwan, 'Data Tiering in Heterogeneous Memory Systems', Proc. the 11th ACM European conference on Computer systems (EuroSys 2016) (2016.4)
[17] DDR4 SDRAM, https://en.wikipedia.org/wiki/DDR4_SDRAM
[18] Kazuichi Oe, Takeshi Nanri, Koji Okamura, 'Analysis of storage workloads of input-output access locality and designing of hybrid storage system', Proc. of the 5th IIAI International Congress on Advanced Applied Informatics (2016.7)
[19] CITRIX XenDesktop, https://www.citrix.com/products/xendesktop/overview.html
[20] Database Test Suite, http://osdldbt.sourceforge.net/
[21] TPC Benchmark C, http://www.tpc.org/tpcc/
[22] Kazuichi Oe, Takeo Honda, and Motoyuki Kawaba, 'Samba workload analysis and consideration for hybrid storage system', IPSJ SIGOS, Tokyo, Japan (in Japanese) (2012.12)
[23] Dushyanth Narayanan, Austin Donnelly, and Antony Rowstron, 'Write Off-Loading: Practical Power Management for Enterprise Storage', in Proc. of 6th USENIX Conf. on File and Storage Tech (2008.2)
[24] SNIA trace data MSR Cambridge, http://iotta.snia.org/traces/388
[25] Device mapper, https://en.wikipedia.org/wiki/Device_mapper
[26] brd driver, https://github.com/spotify/linux/blob/master/drivers/block/brd.c
[27] Nimrod Megiddo and Dharmendra S. Modha, 'ARC: A SELF-TUNING,LOW OVERHEAD REPLACEMENT CACHE', Proc. 2th USENIX conference on File and Storage Technorogies (FAST2003) (2003.3)
[28] sim-ideal, https://github.com/arh/sim-ideal
[29] Subramanya R Dulloor, Amitabha Roy, Zhenguang Zhao, Narayanan Sundaram, Nadathur Satish, Rajesh Sankaran, Jeff Jackson, and Karsten Schwan, 'Data Tiering in Heterogeneous Memory Systems', Proc. the 11th ACM European conference on Computer systems (EuroSys 2016) (2016.4)
[30] Jiaxin Ou and Jiwu Shu and Youyou Lu, 'A High Performance File System for Non-Volatile Main Memory', Proc. the 11th ACM European conference on Computer systems (EuroSys 2016) (2016.4)
[31] Jian Xu and Steven Swanson, 'NOVA: A Log-structured File System for Hibrid Volatile/Non-volatile Main Memories', Proc. of the 14th USENIX Conf. on File and Storage Tech. (FAST'16) (2016.2)
[32] Gokul Soundararajan, Madalin Mihailescu, Cristiana Amza, 'Context-Aware Prefetching at the Storage Server', The 2008 USENIX Annual Technical Conference (ATC 2008) (2008.6)
[33] Mingju Li, Elizabeth Varki, Swapnil Bhatia, and Arif Merchant, 'TaP: Table-based Prefetching for Storage Caches', In Proc. of 6th USENIX Conf. on File and Storage Tech (2008.2)
[34] Takahiro Hirofuchi and Ryousei Takano, RAMinate: Hypervisor-based Virtualization for Hybrid Main Memory Systems, In Proc. of SoCC'16 (2016.10)