spike 領域をリアルタイムに高速ストレージに移動することが可能な階層ストレージシステムの提案

大江和一[†] 荻原一隆[†]
野口泰生[†] 小沢年弘[†]

長時間にわたり特定の領域への負荷集中が発生するワークロードを対象に，負荷が集中する領域を動的に抽出し，その領域をリアルタイムに高速ストレージ (SSD) に移動する階層ストレージシステムの提案を行う．MSR Cambridge ワークロード[2] を用いて評価を行ったところ，最大 77%性能向上することを確認した．負荷の分析に使用する blktrace,iostat がユーザからの IO に与える影響が軽微であることも確認した．

Proposal for a hierarchical storage system that can move a spike to high-speed storage in real time

Kazuichi Oe,[†] Kazutaka Ogihara,[†] Yasuo Noguchi[†] and Toshihiro Ozawa[†]

For storage workloads including a significant spike, we propose a hierarchical storage system that can detect a spike area dynamically and move the area into high-speed storage like SSD in real-time. Our evaluation using MSR Cambridge workload revealed that our storage system can response quickly to spikes and improve IOPS performance by up to 77% compared with conventional HDD-only systems. Additionally, it was confirmed that workload monitoring for spike detection had negligible performance influence on user IO.

1. はじめに

ストレージワークロードの分析を行うと，一部のデータ領域に負荷集中するものが少なからず存在する．例えば，インターネットから利用されるサーバの負荷集中の調査を行った Workload Spikes 論文[1] によると，全体の 1%のデータに 32-88%の負荷が集まり，その負荷が数時間から最大数日継続する．そして，そのデータを支える各ストレージボリュームにおいても同様な負荷集中が発生する報告が行われている．また，samba ワークロード[3] や MSR Cambridge ワークロード[2],[4] のようなより規模が小さなものにおいても，同様な傾向にあることが分かった．この任意のデータ領域に一定時間発生する負荷集中を例えば RAID 装置だけで負荷設計を行うと，最大負荷に対応した構成となり非常に高価なシステムとなってしまう．

そこでこのような負荷集中が発生するワークロードを対象に，コストを上げることなくボリューム全体の性能向上とピーク性能の改善を目指して，負荷が集中する領域を動的に抽出し，抽出した領域をリアルタイムに高速ストレージ (SSD) 側に移動することを特徴とする HDD との階層ストレージシステムを提案する．提案方式を利用することで，ごく一部の領域を SSD に移動するだけでストレージシステム全体の性能を大幅に向上し，HDD 単体では耐えられない負荷を処理することが出来る．さらに，負荷が集中する領域が SSD 側に移ることで HDD 側の seek などのオーバーヘッドが小さくなり，HDD 側から取り出せる IO 数が向上する効果も得られる．

商用の多くの自動階層制御方式は，毎日決まった時間にそこまでに蓄積した統計情報を用いて負荷が高いデータ領域の抽出& 移動を行うものであり，少なくとも数日間以上負荷が継続しないと効果が得られない．本提案方式では，負荷が高いデータ領域に変化があった場合リアルタイムに検出& 移動を行うものであり，数時間しか継続しない負荷集中に対しても効果が期待できる．

† (株) 富士通研究所
Fujitsu Laboratories Ltd.

なお，本稿ではこれ以降負荷が長時間継続して発生する領域を spike 領域，短時間で負荷が収束する領域を short spike 領域と呼ぶことにする．

以下に本稿の構成を示す．**2 章**で関連研究を紹介する．**3 章**で spike 領域が発生するストレージワークロードに関する考察に関して説明する．**4 章**で提案する階層ストレージシステムの説明を行う．**5 章**で提案方法の評価に関して説明し，**6 章**をまとめとする．

2. 関 連 研 究

SSD などの高速ストレージと HDD などの低速ストレージを組み合わせた階層ストレージシステムは，製品レベルまで達したシステムも存在する．EMC FAST では自動的に階層間移動を行うことが可能になる．文献 6) によると，1GB slice でボリュームを分割し，この単位で統計情報を収集し，階層間移動をおこなう．さらに，毎日決まった時間に階層間移動を行うと説明されている．これらより，少なくとも数日以上負荷が高いデータ領域を，負荷が低い深夜時間帯などを利用して SSD に移動する方式であると考えられる．本稿で提案するリアルタイムに spike 領域を抽出し，SSD に移動する様な機能の実装はない．

性能だけではなくコストや消費電力などの指標を用いて階層制御を行う提案も行われている．Dushyanth らの論文[7] では，使用するストレージの価格性能比や容量性能比を用いてデータ配置を決める提案が行われている．

分散ストレージシステムを対象にワークロードに応じて動的にストレージ構成を変更する提案として文献 10) が上げられる．ここでは 10-100MB で分割した bin という単位でリクエスト遅延の監視を行い，この値があらかじめ定められた Service-Level Objectives(SLOs) を満たせない場合にあらかじめ定められた性能モデルに従って bin の再配置を行う提案である．

SSD に代表される高速ストレージをキャッシュとして利用する研究として facebook flashcache[5] や Directcache[11] などが存在する．これらは全て OSS としてソースが公開されていたり，製品として販売されていたりする．このうち flashcache は IO offset が 4KB 境界の IO のみしかキャッシュ出来ないので，IO offset が 4KB 境界とならない IO の割合が多いワークロードでは効果が期待できない．また，キャッシュ方式で実現した場合，SSD から HDD へのライトバックがユーザ IO とは別に発生する．本稿の提案でも階層移動時にユーザ IO とは別の IO が発生するが，IO がシーケンシャル IO 1 回で済むので，本提案の方がユーザ IO への影響は小さいと判断している．

spike 検出という観点でストレージワークロードの分析を行った研究紹介も行っておく．文献 1) はインターネットから利用されるサーバの負荷集中により発生する spike の分析とモデル化を行っている．本稿でもこの分析結果を参考に，より小さなワークロードにおける spike の分析を行った．文献 9) は Windows Server 2008 上に構築した 12 の業務サーバに関する ETW ☆ 分析結果がまとめられている．この分析結果においても Exchange server などで spike 発生の報告が行われている．

3. spike 領域が発生するストレージワークロードに関する考察

インターネットから利用されるサーバの負荷集中の調査を行った Workload Spikes 論文[1] では，data spike と volume spike の存在が明記されており，data spike に関して以下のような報告が行われている．

- Top 1%データへの負荷：全アクセスの 32-88%
- Top 10%データへの負荷：全アクセスの 67-99%
- peak 到達まで 40-130 分．継続時間は数時間から数日．

さらに，企業内で運用される samba ワークロードを 1ヶ月間分析した報告[3] においても，spike 領域☆☆が発生していることが分かる．この領域は全データ領域の 0.9%にしか過ぎず，さらに日中の時間帯に最大 480 分程度負荷が継続する．

そこで文献 1) の様なインターネットから利用されるサーバの負荷集中だけではなく，より小さいワークロードにおいても同等な結果が得られるのではないかと考え，一般に公開されている MSR Cambridge ワークロード[2] における負荷集中の調査を行った．MSR Cambridge ワークロードはケンブリッジにある Microsoft Research サーバに関するブロック I/O トレースデータである．このトレースデータを用いた論文[4] の Table 1 にトレースデータの概要がまとめられている．このトレースデータは 13 servers, 36 volumes, 179 disks から収集しており，User home directories や Web/SQL server など様々なサーバの IO のトレースとなっている．この中から最大 50iops 以上の負荷が発生していた 13 volumes ☆☆☆ を対象に spike 領域発生に関する調査を行い，以下の結果が得られた．

☆ Event Tracing for Windows
☆☆ 1iops 以上の負荷が発生する領域
☆☆☆ mds_1,proj_1,proj_2,proj_4,prxy_1,src1_0,src1_1, src2_1,stg_1,usr_1,usr_2,web_0,web_2

- Top 1%ボリュームへの負荷:全アクセスの 8-95%
- Top 10%ボリュームへの負荷:全アクセスの 48-100%
- peak 到達まで 10-60 分．継続時間は 10-200 分．

文献 1)，3) に掲載された調査結果，及び一般に公開されている MSR Cambridge ワークロードに対する分析結果より，インターネットで利用されるサーバのワークロードだけではなく，企業内の samba や MSR Cambridge の様な規模の小さなワークロードにおいても全領域の 10%程度に spike 領域が発生していることが分かった．さらに，spike の継続時間は数時間で終わっているものも多く，これらの短時間で終わる spike に対して効果的に性能改善する方法が必要なことが判明した．

4. 提案方法の説明

4.1 概　　要

本稿では，spike 領域を動的に捉え，その領域をリアルタイムに高速ストレージに移動することを特徴とする SSD と HDD との階層ストレージシステムの提案を行う．提案方式を利用することで，ごく一部の領域を SSD に移動するだけで spike 領域で発生する IO を受け止めることが可能になり，安価で高性能なストレージシステム構築することが出来る．必要とする SSD 容量は，全ボリューム容量の最大 10%程度を目安とし，数時間程度 spike 領域が継続するワークロードを主なターゲットとする．移動に伴うコストを性能向上効果が上回れば，トータルで性能向上を期待できる．

spike 領域を動的に捉え，SSD へ移動する方法に関して述べる．ストレージボリュームを segment という固定の大きさに分割して扱う．この segment 単位に負荷の大小の判断を行い，負荷の高い segment を SSD へ移動する．segment の大きさに関しては，大きくとると spike 領域以外も含んでしまう確率が上昇するが spike 領域の揺らぎには強くなる．一方小さくとると，spike 以外の領域を減らすことが出来るが，spike 領域の揺らぎにより SSD へ移動した領域から外れる確率が上昇する．segment の大きさに関して定量的な指針は確立していないため，本稿では 1GB として評価を行いその妥当性に関して評価時に議論することにした．

図 1 は，本稿の評価で使用した CentOS 5.4 上での実装例である．この図を用いて提案システムの説明を行う．提案システムは，アプリケーションとして動作する分析･構成変更エンジンと OS ドライバとして動作する Tiering driver から構成される．ストレージワークロードの分析に用いるデータとしては，各 segment ごとの情報が採取できる blktrace と各デバイスの情報が採取できる iostat の実行結果を用いる．

4.1.1 分析・構成変更エンジン

分析・構成変更エンジンは，Log pool，ワークロード分析，segment 移動指示，の 3 コンポーネントから構成される．以下の節で各コンポーネントに関して説明する．

4.1.1.1 Log pool

Log pool では，各 segment ごとの統計情報と階層制御に使用する各デバイスの統計情報を収集する．これら情報を用いて，高負荷 segment の抽出を行う．各 segment ごとに収集する統計情報は，一定時間間隔ごとの IO 数，IO size ごとの分布，read/write の割合，レスポンスのヒストグラム，である．各デバイスごとに収集する統計情報は，一定時間間隔ごとの iops，IO busy 率，などである．

実装では，各 segment ごとの統計情報は blktrace を用いて抽出する．各デバイスの統計情報は iostat を用いる．実行間隔は 1 分間である．

blktrace ではボリューム全体に発生する IO のトレースを，階層制御で IO を SSD と HDD に分離する前の段階である Tiering driver のレベルで取得する．**図 1** を例に取ると，/dev/mapper/tierdev に対して行う（**図 2** 参照）．

階層制御に使用する各デバイスの統計情報は，1 分間隔で実行した SSD，HDD 各デバイスに対応する iostat -x の実行結果を収集する．これらのデータを用いて，各デバイスの IO busy 率を把握する．**図 1** を例に取ると，/dev/sdb と /dev/sdc の iostat 実行結果の蓄積を行う．

測定間隔を 1 分間とした理由に関して説明する．実ワークロードでは均一に io が出ることはまれであり，測定間隔を短くしていくと 1 つのデータだけではワークロードの特性を正確に把握出来ない可能性が高まる．read と write が混在するワークロードを例に取ると，測定間隔を短くすることで read と write の混在比がデータごとに振動する場合があり，何らかの平準化処理が必要になる．今回は比較的大きな測定間隔を用いることでこの問題を回避した．

4.1.1.2 ワークロード分析

ワークロード分析は，負荷の高い segment を抽出しその中から SSD に移動する segment を決めることがその目的である．分析には，Log pool に収集した複数のデータを用いて行う．継続して負荷が高い segment を抽出するために，Log pool より直前の n データを取り出しその平均値を求めておく．次に HDD 側の %util

☆があらかじめ決められた閾値 ($\%util^h_{up}$) を超えた場合に SSD 側に高負荷 segment を移動する．今回の実装では n=2 とした．詳細説明は **4.2 章**で行う．

4.1.1.3 segment 移動指示

segment 移動指示は，ワークロード分析の結果を受け，Tiering driver に対して segment の移動を指示する．今回の実装では，/proc に Tiering driver に対して segment 移動を行う API を準備した．複数の segment 移動を行う場合，ユーザからの IO への影響を少なくする目的で，HDD 側の負荷に応じて移動を実施する時間間隔を変更する実装を入れた．segment 移動を同時に複数依頼すると，ユーザからの IO が長時間ブロックされるためである．今回の評価に使用した環境 (**表 1**) では，ユーザからの IO がない状態で 10 秒程度で segment 移動が可能である．さらにユーザからまとまった IO が発生した状態で評価を行ったところ，少なくとも 60 秒程度間隔をおいて segment 移動を行うとユーザ IO への影響が目立たなくなることが分かった．そこで，現在の%util が$\%util^h_{up}$ を上回る場合は 60 秒間隔で実行する様にした．

4.1.2 Tiering driver

Tiering driver は，Tiering table の内容に従って SSD もしくは HDD のいずれかに IO の送出を行う．**Tiering table** には，SSD に移動した segment 情報のみが掲載され，SSD 側の先頭 offset，HDD 側の先頭 offset を保持している．HDD の先頭 offset がボリューム全体の offset と一致する．登録してある HDD 側の先頭 offset から segment の大きさの範囲に収まる IO 要求が来た場合，SSD 側へ IO 要求を出す．Tiering table は segment 移動指示による SSD〜HDD 間のデータ移動後に更新する．移動方法の詳細は **4.3 章**で行う．今回の実装では，Tiering driver は Linux device-mapper framework 上に実装した．

4.2 高負荷 segment の抽出方法

制御の基本方針は，HDD 側の%util を$\%util^h_{up}$〜$\%util^h_{lo}$ の範囲に留めることで HDD 側が過負荷にならないようにすることである．さらに$\%util^h_{up}$ と$\%util^h_{lo}$ の中間値として$\%util^h_{md}$ を設ける．%util が$\%util^h_{up}$〜$\%util^h_{lo}$ から出た場合，$\%util^h_{md}$ になるように制御する（**図 3** 参照）．なお，**図 3** の%utl_hdd_upper は$\%util^h_{up}$ である．%utl_hdd_middle は$\%util^h_{md}$ である．%utl_hdd_lower は$\%util^h_{lo}$ である．

segment の移動は文献 8) の方法を参考にした．最初に HDD 側の%util が$\%util^h_{up}$ を上回るケースの説明を行う．まず，HDD 側に存在する各 segment を iops の大きい順にソートしておく．次に，HDD 側の%util と iops より iops 当たりの%util を求めておく．そして，ソートした順に各 segment の iops を iops 当たりの%util に掛け合わせ，得られた結果を加算していく．k 番目の segment の iops を hdd_iops(k) とすると，以下の計算を行うことになる．

$$\%util\_del = \sum_{k=1}^{n} hdd\_iops(k) * (\%util\_per\_iops)$$

現在の HDD 側の%util から%util_del を引いた値が$\%util^h_{md}$ を下回る n の値を求め，その segment までを SSD へ移動する．

次に HDD 側の%util が$\%util^h_{lo}$ を下回るケースの説明を行う．このケースでは，負荷の軽い SSD 側の segment を HDD 側に移動することになる．まず，SSD 側に存在する各 segment を iops の小さい順にソートしておく．次に，HDD 側の%util と iops より iops 当たりの%util を求めておく．そして，ソートした順に各 segment の iops を iops 当たりの%util に掛け合わせ，得られた結果を加算していく．k 番目の segment の iops を ssd_iops(k) とすると，以下の計算を行うことになる．

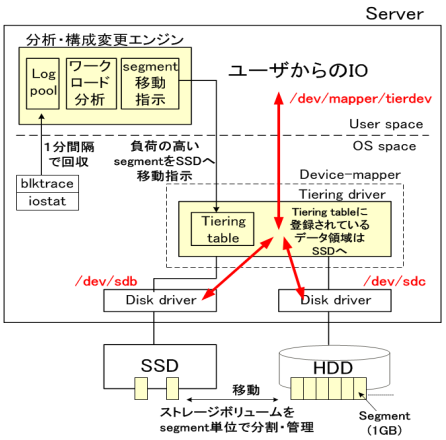

図 1 提案方法の Linux(CentOS 5.4) での実装例

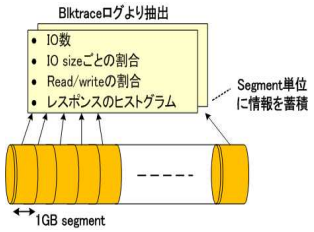

図 2 blktrace データの分析方法

☆ iostat の 1 パラメータで IO busy 率を表す

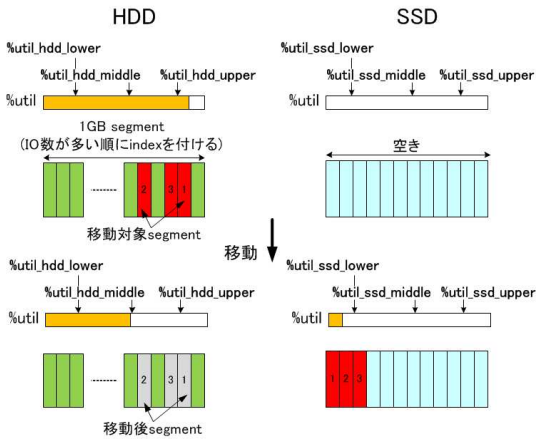

図 3　HDD から SSD に segment を移動する方法

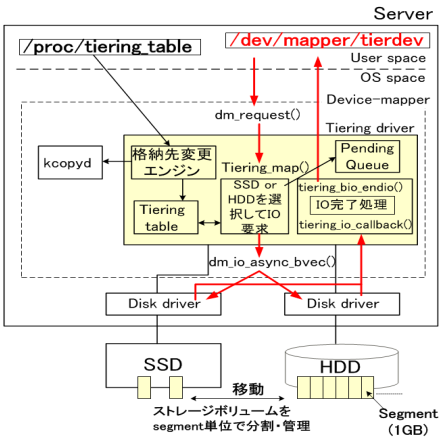

図 4　tiering driver の Linux(CentOS 5.4) での実装例

$$\%util\_add = \sum_{k=1}^{n} ssd\_iops(k) * (\%util\_per\_iops)$$

現在の HDD 側の%util から%util_add を加算した値が$\%util^h_{md}$ を上回る n の値を求め，その segment までを HDD へ移動する．

今回の評価では，$\%util^h_{up}$=70，$\%util^h_{md}$=50，$\%util^h_{lo}$=10，とした．

4.3　SSD への移動方法

ユーザからの IO が Tiering driver に入ってくると，IO 処理を行う Tiering_map() で受け取った IO 要求の offset 値と Tiering table の比較を行う．比較の結果，Tiering table に載っていれば SSD へ，そうでなければ HDD へ IO 要求を送出する．分析・構成変更エンジンからの移動指示は/proc/tiering_table を介して行う．Tiering driver は，移動指示を受け取ると device-mapper 内共通モジュールである kcopyd を用いて segment 移動を行う．データ移動中の segment へのユーザ IO は Pending Queue に蓄積しておき，segment 移動完了を契機に Pending Queue に蓄えられた IO を実行する．(図 4 参照)

5.　提案方法の評価

5.1　概　　要

評価は以下の 3 つの観点で行った．

- HDD 単体との比較
- spike 領域と short spike 領域の構成比変更実験
- blktrace,iostat オーバーヘッド調査

評価に使用した環境を表 1 にまとめた．評価に使用したワークロードは，**3** 章の調査に使用した MSR Cambridge ワークロードである．このデータの中から負荷集中が長時間継続していた**表 2** に示す 4 ワークロードを選択した．いずれも 1 時間以上継続する高負荷状態が繰り返し発生するものである．**表 3** はこれらワークロードの負荷の偏りを整理したものである．いずれも全ボリュームの 10%に半分以上の負荷 (iops) が発生している．

ワークロードの再現は blktrace で取得したデータの再生を行う btreplay を使用した．btreplay が解釈可能な形式に変換する filter を通して実行した．評価の目的は，HDD 単体で負荷を処理出来ないワークロードに対して，僅かな SSD 容量に本稿の提案方式を組み合わせることで負荷を処理出来ることを示すことにある．そこで，そのままでは HDD 側を過負荷に出来ないワークロードに関して，btreplay を倍速実行☆することで過負荷状態をつくり出した．なお，本稿で過負荷とは，iostat %util が継続してほぼ 100%となってしまう状態を指す．

5.2　HDD 単体との比較

HDD から SSD への segment 移動オーバーヘッドまで含めた提案方法の効果を確認する目的で，HDD のみで表 2 のワークロードを実行した場合との比較を行った．比較方法は表 2 の負荷集中が発生した開始時間からワークロードを起動し，1 時間後までの平均 iops の比較である．btreplay は，usr1, web2 は 8 倍速で実行し，proj2，proj4 は 2 倍速で実行した．**表 4**

表 1　ストレージ装置性能調査環境

PC	FUJITSU PRIMERGY RX200S6 Xeon E5640 2.67 GHz,Mem=32GB,CentOS 5.4
HBA	LSI SAS 9200-8e
SSD	intel X25-E(SSDSA2SH064G1GC) (64GB)
HDD	SEAGATE ST9600204SS (600GB)

☆ -x ⟨num⟩ で何倍速にするのかを指定する

が実行結果である．usr1，web2，proj2 で性能向上が見られ，最大 77%向上しているが，proj4 ではほとんど効果が得られない．

表 5 は提案方法における SSD rw の割合☆，及び提案方法と HDD 単体における HDD 側の svctm ☆☆である．usr1 は SSD 6GB の範囲に 89%の IO が発生しており，提案方法により spike 領域を SSD に移動でき，その結果として HDD 単体と比較して 77%性能向上した．全体の 1%の領域 (SSD 6GB) を移動するだけで性能向上出来たことが分かる．

web2 は SSD に移動した 27GB の領域に 55%の IO が発生している．usr1 より効果が小さい主な理由は移動領域が多いことである．今回の実装では，**4.1.1.3 章**にて説明したように複数の segment 移動を同時に行う場合は 60 秒間隔で行う．27GB だと移動だけでも 27 分必要としてしまい，segment 移動による十分な効果が得られるまで時間がかかるためである．

proj2 は SSD に移動した 23GB の領域に 11%の IO しか発生していないが 45%性能向上していることが分かる．表 5 の提案方法と HDD 単体における HDD 側の svctm を比較すると，提案方法の HDD 側 svctm が 0.39ms 向上していることが分かる．これは SSD への一部 segment の移動により HDD 側のオーバーヘッドが小さくなり，HDD 側から取り出せる性能が大きくなったことを示している．提案方式と HDD 単体の %util の比較を行ったところ，両者とも 1 時間の平均値がほぼ 100%であった．このことより，svctm の差がそのまま性能差となったと言える．

proj4 は SSD に移動した 23GB の領域に 1%の IO しか発生せず，向上率も 1%であることが分かる．**図 5** より，ベンチマーク終了時には提案方法の HDD 側の%util は 50%程度でそれほど負荷がかかっていないが HDD 単体は常に 100%であることが分かる．**図 6** は提案方法と HDD 単体における 1 分間隔の iops の推移である．提案方法では，3000iops 超の負荷が 3 回発生し，一度に高負荷状態を解消し，その後負荷が下がっていることが分かる．一方，HDD 単体は提案方法の負荷が下がり始めた 31 分後も 1000 iops 前後の性能を維持し，結果として提案方式との差がなくなった．proj4 は平均 iops 向上には効果がなかったが，peak iops 向上には効果があった．SSD へ 23GB 移動したのにアクセス量が 1%に留まった理由は，負荷集中する segment が移動していき，継続時間が短かったためと考えられる．

まとめると，usr1 の様に spike 領域が狭いワークロードでは，提案方法によりすぐに性能向上する．web2 の様に spike 領域が広いワークロードでは，segment 移動に時間が必要なため，十分な効果が出るまで時間が必要となる．proj2 では，spike 領域が SSD に移動することで HDD 側オーバーヘッドが小さくなり，SSD から得られる性能以上に向上する場合がある．proj4 の様に spike 領域の負荷が極めて小さいワークロードでも，peak iops 向上に効果がある場合がある．

最後に segment の大きさに関する議論を行っておく．**表 6** は評価に用いた各ワークロードの spike 領域の大きさである．usr1，proj2 は平均値が 0.3-0.4GB であるので 1GB segment にすると無駄な領域が生じるが，proj4 は 1GB が適しており，web2 は segment size を 12GB 程度にするのが適している．一方，segment size を大きくすると HDD から SSD への転送時間が増加してしまい，ユーザ IO に影響を与える可能性が出る．本稿の検証環境では 1GB 転送するのに 10 秒程度かかっていた．SCSI timeout が 20-30 秒であることを鑑みると，本稿の検証環境では segment size を最大 2GB 程度に留めるべきであることが分かる．まとめると，本方式を適用する環境ごとに，HDD から SSD への転送時間により上限 segment size を設定した上で，ワークロードに応じて segment の大きさを変更出来る様に実装するのが SSD 領域をより有効に利用できる．

表 2 評価に使用したワークロード

ワークロード	処理内容	全容量 (GB)	peak iops	負荷集中が発生した時間帯 (分)*
usr1	ユーザのホームディレクトリ	600	237	2090-2340
web2	web と SQL サーバ	170	231	5870-5930
proj2	プロジェクトディレクトリ	600	1070	6760-6880
proj4	プロジェクトディレクトリ	236	794	7390-7590

*: トレース先頭からの経過時間

表 3 評価ワークロードの負荷 (iops) 集中度

全ボリュームに占める割合	usr1	web2	proj2	proj4
1%	95%	8%	17%	43%
10%	100%	48%	75%	70%

☆ =(SSD への IO 数*100)/(全 IO 数)
☆☆ iostat の 1 パラメータ．1IO の平均実行時間

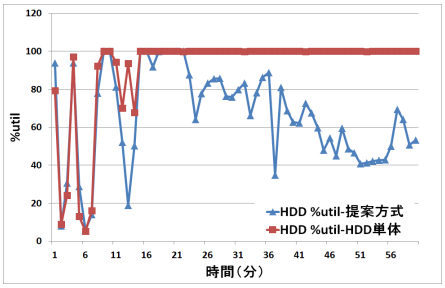

図 5　proj4 の 1 分間隔の%util(HDD 単体との比較)

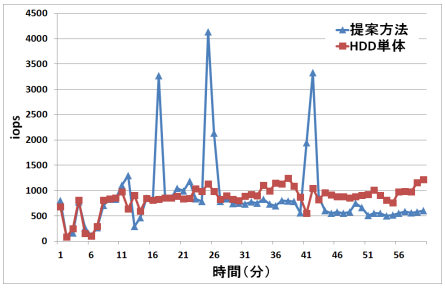

図 6　proj4 の 1 分間隔の iops(HDD 単体との比較)

5.3　spike 領域と short spike 領域の構成比変更実験

本実験の目的は，同じ領域に長時間継続して負荷が発生する spike 領域と負荷が短時間に収束& 移動する short spike 領域に発生する負荷の割合を変更した実験を行うことで，提案方式が効果的なワークロードを明確にすることである．

はじめに本実験環境における short spike 領域の負荷継続時間に関して議論する．short spike 領域を本稿の実装に照らしあわせると，少なくとも spike が発生した segment を検知し，その segment を SSD へ移動が終わったときには，spike が収束している領域，となる．提案システムは直前の 2 data を用いて判定を行っており，segment 移動は少なくとも segment 当たり 10 秒必要とする．この事実より，本実験環境における short spike 領域の負荷継続時間は，少なく見積もっても 3 分程度であり，segment 数に応じて増加する．

実験には表 2 の proj4 を用いた．proj4 を調査したところ 4GB の spike 領域と 187GB の short spike 領域から構成されることが分かった．そこで spike 領域と short spike 領域の負荷割合を変更できるようにするために，proj4 を spike 領域と short spike 領域に分離し，それぞれを重複して btreplay の実行を可能にした．spike 領域の btreplay 倍速度は 4 で固定し，short spike 領域の倍速度を 1 から 5 に変化させることで実験を行った．btreplay の実行時間は 20 分とした．事前の調査で short spike 領域の負荷を 4 倍速以上で実行すると HDD 側%util がほぼ 100%となることも把握した．

表 7 が実験結果である．SSD 使用量には，SSD へ移動が完了していない segment は含まない．増加 iops(期待値) は short spike x1 実測値が倍速加算された場合の値である．実験結果より，ピーク性能が得られるのは short spike 領域を 3 倍速で実行した場合である．

表 4　1 時間経過後の平均 iops(HDD 単体との比較)

ワークロード	平均 iops (提案方法)	平均 iops (HDD 単体)	SSD 使用量 (GB)	向上率 (%)
usr1	1483	836	6	77
web2	1439	1174	27	22
proj2	635	436	23	45
proj4	865	854	23	1

表 5　SSD rw の割合と svctm(HDD 単体との比較)

ワークロード	SSD rw の割合 (%)	HDD の svctm (提案方法) (ms)	HDD の svctm (HDD 単体) (ms)
usr1	89	0.39	1.20
web2	55	0.60	0.86
proj2	11	1.93	2.32
proj4	1	0.90	1.07

表 6　spike 領域の大きさ (単位: 1GB)

	usr1	web2	proj2	proj4
最小値	0.1	0.1	0.1	0.1
平均値	0.4	11.9	0.3	1.0
最大値	1.0	34.3	1.0	4.8

表 7　spike 領域と short spike 領域の構成比変更実験結果

	平均 iops	SSD 使用量 (GB)	増加 iops (実測値)	増加 iops (期待値)
spike x4	1498	2		
short spike x1	261			
s x4 + ss x1	1676	2	178	261
s x4 + ss x2	1967	6	469	523
s x4 + ss x3	2262	8	765	784
s x4 + ss x4	1699	10	201	1046
s x4 + ss x5	1284		-213	1307

s: spike, ss: short spike

表 8　blktrace,iostat オーバーヘッド調査結果

	分析・構成変更エンジン有り	分析・構成変更エンジン無し
iops	1481	1466
CPU %idle	92.5	92.5
w/s*	8.1	0.5

*: システムディスクに対する write sector per sec.

short spike 領域を 4 倍速以上で実行すると，short spike 領域に関する SSD 移動オーバーヘッドが大きくなり，SSD による性能向上効果が薄れてしまう．

この実験結果より，提案方法は short spike 領域への負荷が HDD 側の性能を大幅に上回らないワークロードで効果的であることが分かる．

5.4 blktrace,iostat オーバーヘッド調査

4.1 章で説明したように，提案方法では blktrace と iostat を常に実行することで，ワークロードのリアルタイム分析を実現した．そこで blktrace と iostat を常に実行することによるユーザ IO への影響調査を行った．調査は usr1 を起動し，6GB segment の SSD への移動をまず行っておく．その後，usr1 を 10 分間づつ再実行し，分析・構成変更エンジンの有無で iops や cpu 使用率がどのように変化するのかを調査した．

表 8 は実験結果である．ほぼ同じ iops が得られている状態で CPU %idle がほぼ同じ値となることが確認できた．システムディスク側の write 負荷が若干増加するが，ユーザ性能に影響を与えるほどではないことが確認できた．今回実験に使用した環境では，blktrace,iostat のユーザ IO への影響はほとんどないと判断できる．

6. ま と め

特定の領域への負荷集中が数時間発生するワークロードを対象に，コストを上げることなくボリューム全体の iops 性能向上と，ピーク iops 性能の改善を目指して，負荷が集中する領域を動的に抽出し，その領域をリアルタイムに高速ストレージ (SSD) に移動する階層ストレージシステムの提案を行った．提案方式を利用することで，最大でも全ボリューム容量の 10%未満の SSD 容量を追加するだけでストレージシステム全体と，ピーク性能を大幅に向上 (最大 77%) することが出来る．

spike 領域が狭い場合は，すぐに大きな性能向上が得られること，広い場合には効果が出るまで時間がかかるので spike の長時間の継続が必要なことが分かった．この点を改善するためには，SSD への segment 移動方法を改良する必要がある．また，spike 以外の領域の負荷についても，spike 領域が SSD に移ることで HDD のオーバーヘッドが小さくなり，SSD から得られる性能以上に向上するケースがあることを確認した．負荷の分析に使用する blktrace,iostat がユーザからの IO に与える影響が軽微であることも確認した．

今後の課題は以下である．

- ユーザ IO への影響を最小にしつつ，迅速に SSD への segment 移動する方法の追求
- ワークロードの特徴に応じた高負荷 segment 抽出方法
- 負荷が下がってきたときに HDD へ segment を戻す方法の検証

参 考 文 献

1) Peter Bodik, Armando Fox, Michael J.Franklin, Michael l.Jordan, and David A.Patterson. Characterizing, Modeling, and Generating Workload Spikes for Stateful Services. ACM Symposium on Cloud Computing (SOCC 2010), June 2010.
2) http://iotta.snia.org/traces/388
3) 大江和一，熊野達夫，野口泰生: iops 当たりの IO Busy 率を用いた IO 性能設計方法の提案，第 8 回先進的計算基盤システムシンポジウム SACSIS2010, May 2010.
4) Dushvanth Narayanan, Austin Donnelly, and Antony Rowstron, Microsoft Research Ltd. Write Off-Loading: Practical Power Management for Enterprise Storage, Proc. 6th USENIX Conference on File and Storage Technologies (FAST 08)
5) https://github.com/facebook/flashcache
6) EMC White Paper, EMC FAST VP for Unified Storage Systems A Detailed Review, March 2011
7) Dushyanth Narayanan, Eno Thereska, Austin Donnelly, Sameh Elnikety, and Antony Rowstron, Microsoft Research Cambridge, UK. Migrating Server Storage to SSDs: Analysis of Tradeoffs. the 4th ACM European conference on Computer systems, April 2009.
8) 大江和一: iops 当たりの busy 率を用いた IO 性能保証方法の提案，第 23 回コンピュータシステムシンポジウム Comsys2011, Dec., 2011.
9) Swaroop Kavalanekar, Bruce Worthington, Qi Zhang, and Vishal Sharda. Characterization of Storage Workload Traces from Production Windows Servers, the 7th International Smantic Web Conference(ISWC2008), Octorber,2008
10) Beth Trushkowsky, Peter Bodik, Armando Fox, Michael J.Franklin, Michael I.Jordan, and David A.Patterson. The SCADS Director: Scaling a Distributed Storage System Under Stringent Performance Requirements. 9th USENIX Conference on File and Storage Technologies, February,2011.
11) http://www.fusionio.com/data-sheets/directcache/