Vol.2012-OS-123 No.1
Vol.2012-EMB-27 No.1
2012/12/5

階層ストレージ方式検討に向けた商用Sambaワークロード分析と考察

大江 和一[1,a)]　本田　岳夫[2]　河場 基行[1]

概要：階層ストレージ制御の 2 方式である cache 方式と tiering 方式の使いかたを明確にする目的で，商用環境の Samba ワークロードを用いてストレージアクセスの空間的局所性とその継続時間に依存した特徴を分析した．その結果，全体の 1%の領域に 81%の負荷が発生し，その 71%が任意の offset に数分から 10 分程度以上発生し，別の offset に移動することが分かった．さらに，write 比が 88%に達することも分かった．この結果を用いた階層制御方法の検討を行い，例えば 10 分前後以上継続する負荷のみをリアルタイムに tiering し，残りの負荷は cache を用いるなど，負荷の継続時間に応じてリアルタイムに tiering と cache を使い分ける方法の有効性を示した．この方法はファイル共有サーバ一般に適用出来ると判断している．

キーワード：ストレージ，階層，SSD，cache，tiering，リアルタイム

Samba workload analysis and consideration for hybrid storage system

Kazuichi Oe[1,a)]　Takeo Honda[2]　Motoyuki Kawaba[1]

Abstract: We investigated spatial locality and duration-dependent characteristics of storage accesses using commercial workload on Samba in order to build a cost-effective hybrid storage system, which can harness caching and tiering appropriately. The experimental result unveils that 81% of the loads arose on 1% of the storage area, and 71% of the concentrated loads were migratory, that is, the loads tend to hop to different areas after they continue for a couple of minutes to 10 minutes at arbitrary offsets. Additionally, we discovered that the ratio of r/w was quite high (88%).
Based upon the above results, this paper discussed the effectiveness of a control technique that utilizes caching and tiering appropriately according to load duration. For instance, it may handle only loads that continue for more than 10 minutes with tiering and the others with caching. We believe that this technique can widely be used for hybrid storage systems.

Keywords: storage, hybrid storage system, SSD, cache, tiering, realtime

1. はじめに

近年，SSD の様な高速なデバイスがストレージデバイスとして用いられる様になってきた．SSD は HDD との比較で高速であるが高価となる．そこでコストパフォーマンスを向上する目的で，SSD と HDD を組み合わせた階層ストレージシステムが多数が提案されている．これらシステムは，アクセス頻度が高いデータを SSD に置くことで高速化を図る仕組みである．この階層ストレージシステムの主な方式としては，cache 方式と tiering 方式があげられる．この両方式は従来，日単位の負荷変動に tiering 方式，1 日の中の短い時間単位の負荷変動に cache 方式が用いられてきた．しかし cache 方式を用いると，特に write 比が大きいワークロードでは writeback 負荷が発生するため十分に性能を引き出せない場合がある．そこで 1 日の中の短い時間単位の負荷変動に対しても，負荷変動を常時モニタリン

[1] (株) 富士通研究所
FUJITSU LABORATORIES LTD.
[2] (株) 富士通ソフトウエアテクノロジーズ
FUJITSU SOFTWARE TECHNOLOGIES LIMITED.
[a)] ooe.kazuichi@jp.fujitsu.com

グし，その結果を用いてリアルタイムに tiering する提案が幾つか行われている．Hystor[3] では，負荷を常時モニタリングし，アクセス頻度が多いブロックを tiering する提案が行われている．文献 [4] では，モニタリング結果を元に spike が発生したブロックを抽出し，tiering する提案が行われている．tiering 方式では tier 間移動時間が発生するのでこの移動時間を鑑みた上で方式選択を行う必要があるが，これら提案では移動時間などのワークロードの特徴を詳細に分析した上での評価が十分に行われていない．

そこで本稿では，1）商用環境で採取した Samba ワークロード半年分を用いて空間的局所性とその継続時間に依存した特徴の観点で特徴抽出を行い，2）抽出結果より分析を行ったワークロードをモデル化し，3）最後にモデル化したワークロードの制御方法に関する議論を行う．

分析を行った Samba ワークロードは，社内で実際に運用を行ったサーバのログであり，4.4TB のボリュームへ常時 3000 user 前後からのアクセスが記録されている．このワークロード半年分の分析を行ったところ spike 領域[*2]が定常的に発生し以下の特徴があることが分かった．

- 負荷の集中度
  - 全容量の 0.1%(6GB)：全 IO の 58%
  - 全容量の 1%(53GB)： 全 IO の 81%
- spike 継続時間： 10 分以上が 50-66%
- spike 発生 offset の任意性
  - 広範囲へ分散：71-79%
  - 同一 offset への繰り返し：21-29%
- write 比： 77-88% (spike のみ)

この様なワークロードを対象にした制御方法の検討を行い，spike 継続時間が 10 分前後以上となる負荷は tiering 方式，それ以外は cache 方式を用いる提案を行った．さらに，課題として spike 継続時間の予想することが必要になることを示した．

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 cache 方式と tiering 方式の特徴に関して説明する．4 章でワークロードの分析結果に関して説明し，5 章で分析結果を用いたワークロードのモデル化と制御方法の提案を行う．6 章でまとめを行い，7 章で今後の課題を説明する．

2. 関連研究

ストレージワークロードの分析を行った研究紹介を最初に行う．文献 [5] 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．この分析では，負荷の大きさとその継続時間，全データ量に占める spike の割合までの分析は行われている．しかし，同じ offset に繰り返し spike が発生するのか等の空間軸方向の分析は行われていない．文献 [6] は Windows Server 2008 上に構築した 12 の業務サーバに関する ETW[*3]分析結果がまとめられている．この分析結果に関しても空間軸方向の分析は行われていない．

負荷変動を常時にモニタリングし，その結果を用いてリアルタイムに tiering する研究紹介も行っておく．Hystor[3] は，SSD を remap area と write-back area に分離し，critical ブロックを remap area に置く制御を行う．critical ブロックはアクセス頻度が高いブロックとファイルシステムのメタデータブロックであり，統計情報モニタリングで抽出する．ワークロード上で発生する spike を dynamic に tiering する提案 [4] も行われている．この研究では，統計情報のモニタリング結果より spike 領域を動的に抽出&移動するシステムの提案を行い，Facebook Flashcache との比較で効果的なワークロードと効果的でないワークロードを示し，其々の原因分析も行っている．どちらの提案も負荷するワークロードの空間的局所性とその継続時間の特徴に応じて効果が変動するため，本稿で扱うワークロード分析が重要であることが分かる．

3. cache 方式と tiering 方式の特徴比較

表 1 は cache 方式と tiering 方式の実装例である．cache 方式は Facebook Flashcache[1] のパラメータを用いた．tiering 方式はリアルタイムに構成変更する方式を前提とし，文献 [4] のパラメータを用いた．

cache 方式は，負荷が発生する領域がストレージボリューム上の広範囲に分散し，その領域が一度に大幅に変わらないワークロードで特に効果的である．負荷が発生する領域が頻繁に入れ替わる write 比が高いワークロードでは，キャッシュブロック入れ替えに伴う writeback が大量に発生し性能遅延を引き起こす可能性がある．この事実は，文献 [4] の Facebook Flashcache との比較評価でも示されている．ブロックサイズは，4KB など比較的小さなサイズを用いる場合が多い．これは，負荷が広範囲に分散し，且つ分散した各負荷の offset 方向の大きさも様々なサイズを想定しているためである．

tiering 方式は，負荷が狭い範囲に定常的に集中するワークロードに特に効果的である．一旦 tiering してしまえば，一時的に負荷が下がっても cache 方式の様に SSD から追い出されることはなく，キャパシティミスによる writeback などの負荷は発生しない．しかし，tiering では tier 間移動時間が必要であるため，負荷が広範囲に分散し短時間で収束するワークロードでは，移動時間に見合う効果が得られないことになる．この事実も，文献 [4] の Facebook Flashcache との比較評価で示されている．ブロックサイズは，負荷が狭い範囲に集中するケースを前提にしているため，EMC FAST[2] など製品レベルにおいても 1GB など比

[*2] なお本稿では，全容量の 1%以下の領域に 50%以上の IO が集まるケースを spike 領域と定義する．

[*3] Event Tracing for Windows

表 1　cache 方式と tiering 方式の実装例

cache or tiering の単位	4KB	1GB
SSD-HDD 転送性能* (MB/sec)	0.8	100
単位当たりの転送時間	5 ms	10 sec
spike 検知遅延	—	60 sec
制御方法	LRU など	**

*: 単位サイズ当たりのランダムアクセスを前提
**: ワークロードの特徴抽出をリアルタイムに行う

較的大きなサイズを用いる場合が多い．大きなブロックサイズを用いると，HDD のシーケンシャルアクセス相当で tier 間移動が出来，負荷が発生・収束した領域入れ替えを迅速に行うことが可能になる．

このため，図 1 の様な比較的短い時間間隔の負荷集中に関しても，tier 間移動時間以上継続する負荷であれば，負荷の変化をその都度とらえて dynamic に tiering することで cache 方式より高性能となる可能性があることが分かる．

4. ワークロードの分析

本章では，最初に分析を行ったワークロードの概要とその収集方法に関して説明し，その後ワークロード分析を行う．ワークロード分析は，事前の調査結果より負荷が集中する平日の 12:00〜17:59 までのデータを用いた (**表 2** 参照)．分析方法は時間的局所性と負荷の継続時間の観点で其々行い，結果を統合する．

4.1 分析を行ったワークロードの概要とその収集方法

4.1.1 ワークロードの概要

今回の分析に用いたデータは，社内で運用しサービス提供していた 4.4TB のストレージ装置に発生したワークロード半年分の蓄積ログである．このワークロードは Samba を用いた情報共有サーバ上で採取されたものであり，平日昼間の 3000 user 前後からのアクセスを記録している．(表 2 参照)．分析を行うワークロードは不特定ユーザからアクセスを長期間蓄積したものとなっており，分析結果は Samba によるファイル共有サーバに広く適用出来ると我々は判断する．

我々は経験的に，ファイル共有サーバの負荷には特定の offset に一定時間負荷が集中し，その後別の領域に負荷が移動する特徴と，負荷の継続時間や offset 幅が一意でない特徴があると考えていた．図 1 は，ワークロードデータより一部を抜粋した経過時間毎の負荷の偏り例である．この特徴の一般性を検証するために Samba ワークロード分析を行った．

このワークロードを負荷したストレージシステムは，AP (Access Processor) と複数の DP (Disk Processor) から構成される分散ストレージシステムである（図 2 参照)．AP 上で 4.4TB の仮想ボリュームを構成し，その上に Samba などのサービスを構築している．仮想ボリュームは 1GB

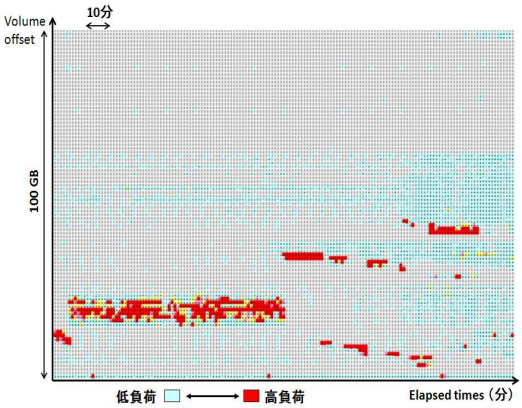

図 1　経過時間毎の負荷の偏り（実データより一部を抜粋）

表 2　分析を行ったワークロードの概要

ボリュームの大きさ	4.4 TB
論理ボリューム数	27
収集期間	2009.9.1 - 2010.3.31
システム構成	Linux+VxFS+Samba
Linux version	RedHat EL 4.4
Samba version	3.0.21b-2
VxFS version	Veritas Storage Foundation 4.1 MP4RP2_HF4
運用方法	情報共有サーバ
上位アプリ	Windows 系が中心
平均ユーザ数	3000

単位で分割し，各 DP にサイクリックに割り当てられる．ワークロード収集は AP-DP 間を流れるパケットを GbE Switch の mirroring 機能を利用して Packet analyzer に収集することで行う．

4.1.2 収集方法

図 2 及び前節の説明のように，Packet analyzer に AP-DP 間を流れるパケットが含まれるストリームが送られてくる．まず、このストリームを tcpdump[7] を用いて分析対象のストレージシステムのパケットのみにフィルタリングして analyzer に渡す．analyzer は，受け取ったパケットのうち read/write に関係するもののみを抽出し，1 分間隔で統計処理したデータをファイルに保存する．統計処理では 4.4TB の仮想ボリュームを 1GB 単位に分割し，この 1GB ごとの IO 数と io size ごとの割合，rw 比，レスポンスなどの情報の集計を行っている．

4.2 負荷の継続時間の観点における分析

4.2.1 分析方法

図 1 で示した様に，このワークロードでは特定の offset に一定時間集中し，その後負荷が移動する特徴があり，さらに負荷の大きさ，継続時間，offset 幅もその都度毎に変化することが分かっている．そこで，負荷が集中した offset 方向と経過時間方向を 1 つの**エリア**としてとらえ，この単位で分析を行うことにした（図 3 参照)．事前分析で一旦

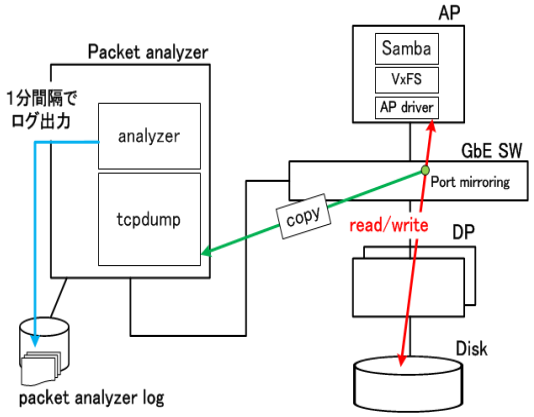

図 2　ワークロード収集を行ったストレージシステムと収集方法の概要

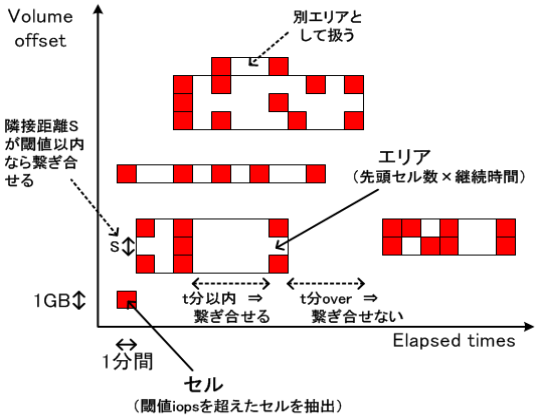

図 3　ワークロード分析方法

表 3　エリア抽出パラメータ

iopm (io per minute)	1, 6, 60, 600
隣接セル間距離 s (GB)	2
隣接セル間時間 t (分)	5

負荷が収束した offset が短い時間間隔ですぐに負荷が回復するケースや負荷が発生した offset の近隣 offset に数 time slice 後に負荷が発生するケースがあることが分かっている．エリアを用いるとこれらを 1 単位で扱うことが出来，HDD-SSD 間の無用なデータ移動を防いだり，近い将来負荷が発生する prefetch 効果も期待できる．本稿の分析の目的は，tiering の使いかたを明確にすることであり，tiering 向いた管理方法であるエリアを用いることにした．

エリアの定義方法を説明する．4.1 章で説明したように，分析に用いるデータは 1GB-1 分間の粒度である．本稿ではこの最小単位を「セル」と呼び，あらかじめ決めておいた IO が発生したセルの抽出をまず行う．エリアはこのセルを offset 方向と継続時間方向につなぎ合わせた領域となる．あらかじめ，offset 方向のセル間距離 (s) と継続時間方向のセル間時間 (t) を定義しておき，このｓとｔの範囲内に入るセルを結合することでエリアを決めていく．

表 3 はエリア抽出に用いたパラメータである．io per minute は，事前の調査で 600 io per minute でセル抽出すると spike 領域を取り出せることが分かり，1, 6, 60, 600 の 4 段階の設定とした．なお，本稿では以後 io per minute を *iopm* と表記することにする．例えば，600 io per minute は 600 iopm とかく．ｓとｔは事前調査で最もセルの充填率[*4]高い値を選択した．

3 章より，tiering 方式と cache 方式の選択にエリア継続時間が必要であることが分かる．そこでさらに，エリアを long/middle/short の 3 種類に分類して分析することにした．long は継続時間 10 分以上，middle は継続時間 3 分以上 10 分未満，short は継続時間 3 分未満である．目安として，long は tiering を選んでよいエリア，middle はエリアのサイズや SSD/HDD 性能に応じて tiering or cache の判断が分かれるエリア，short は cache に負荷したほうがよいエリアである．

[*4] エリア内の抽出セルの割合

4.2.2　分析結果

表 4 はエリアに発生する IO 量に関する分析結果である．分析結果より 60iopm 以上と 600iopm 以上が spike 領域であることが分かる．両者とも 4.4TB の 1%以内に全体の 50%以上の負荷が発生しており，1 章での spike 定義を満たしている．

600iopm に対応するエリアに平均で全体の 58%の負荷が集まっており，且つその平均使用容量が 6 GB（全容量の 0.1%）であることが分かる．図 4 はこの時の使用容量の分布である．最大 44GB で 12GB までが全体の 90%を占めることが分かる．

60iopm に対応するエリアに関しては，平均で全体の 81%の負荷が集まっており，その平均使用容量は平均 53 GB（全容量の 1%）である．図 5 はこの時の使用容量の分布である．最大 192GB で 85GB までが全体の 90%を占めることが分かる．このケースは，エリアの大きさ（継続時間と offset 幅）や SSD-HDD 間の転送性能によって cache 方式と tiering 方式の選択が分かれるところである．

write 比に注目すると，全体平均は 70%であるが負荷が高いエリアほど write 比が大きくなることが分かる．600iopm に絞ると 88%に達する．

平均エリア数と平均使用容量の関係についても説明する．600iopm では平均エリア数と平均使用容量は一致するが，iopm が小さくなると平均使用容量が上回る．これは spike 領域はほぼ 1 点となるが，その周囲の比較的近い範囲に負荷が発生していることを意味する．

表 5 は，表 4 をさらに long/middle/short のエリア継続時間の観点で分析した結果である．高負荷エリアの制御方法検討が分析の目的であるので，1iopm は削除した．

まず，全ての iopm 閾値（6, 60, 600）で long の平均エリア数/容量が大きいことが分かる．ほぼ，long ＞ middle ＞

表 4　エリアに発生する IO 量

iopm*	1	6	60	600
平均エリア数**	—	246	42	6
平均使用容量 (GB)*	654	431	53	6
全 io 数比 (%)	100	97	81	58
write 比 (%)	70	71	77	88
平均 iops	254	246	207	148

*: この値以上のセルを抽出
**: 1 time slice 当たり

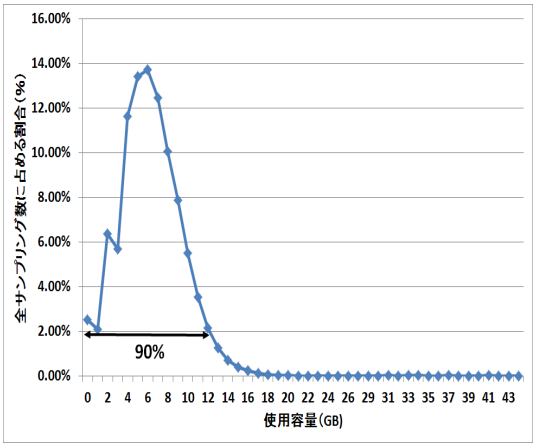

図 4　使用容量の分布 (600 iopm)

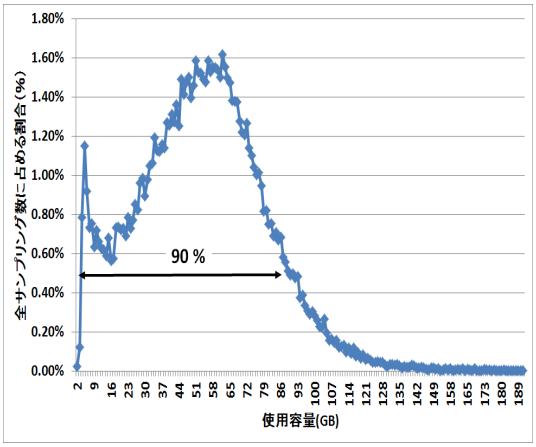

図 5　使用容量の分布 (60 iopm)

= short の傾向となっており，tiering 候補となる spike 領域 (60/600iopm)，且つ long/middle となるエリアが全体の 70-80%のエリア数/ 容量に達することが分かる．600iopm のエリアに限って考察すると，この場合の負荷が全体の 58%であることより，long+middle は全負荷の 48%を占める．よって，600iopm long+ middle エリアのみの性能向上でも全体の性能向上に大きく貢献することが分かる．

4.3　空間的局所性観点における分析

4.3.1　分析方法

空間的局所性を把握するために，4.2.1 節で説明した分析方法に加え以下の観点で分析を行った（図 6 参照）．

- 1GB offset 単位に延べでエリアに属した time slice 数

表 5　long/middle/short ごとの内訳

	平均エリア数	平均使用容量 (GB)
long-6iopm	163	312
middle-6iopm	55	81
short-6iopm	28	38
long-60iopm	21	28
middle-60iopm	11	14
short-60iopm	10	12
long-600iopm	4	4
middle-600iopm	1	1
short-600iopm	1	1

全て 1 time slice 当たりの値

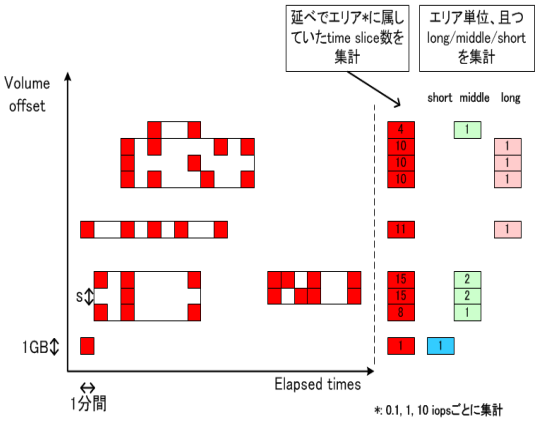

図 6　ワークロード分析方法（空間的局所性）

- 1GB offset 単位に延べで long/middle/short に属した個数

1GB offset 単位に延べでエリアに属した time slice 数を分析することで特定の offset にエリアが集中して発生しているのか，それとも不特定 offset に分散するのかを把握出来る．また，time slice 単位の平均使用容量と組み合わせることでエリアの分散割合を把握できる．さらに，1GB offset 単位に延べで long/middle/short に属した個数を分析することで，long/middle/short ごとの特徴抽出が可能になる．なお，分析結果を用いて主に spike 領域の制御方法を議論することになるため，spike 領域を含まない 6iopm は分析対象から外した．

4.3.2　分析結果

図 7，図 8 は，1GB offset 単位にエリアの発生割合を 27 個の仮想ボリュームごとにまとめたものである．5%を閾値とし 5%を超える場合に割合の高い順に topX (X=1,2,..) の順に掲載し，5%以下は全て REST に統合した．

図 7 は 60iopm 以上のエリアに関する分析結果である．仮想ボリューム ID=1,2,5,27 を除くと，REST の割合が 60%以上となることが確認できる．ID=9-12 など一部の仮想ボリュームは REST=100%となっており，ほぼ任意の 1GB offset にエリアが発生していることが分かる．27 仮想ボリューム全てを合計すると 79%が REST に属していることになる．

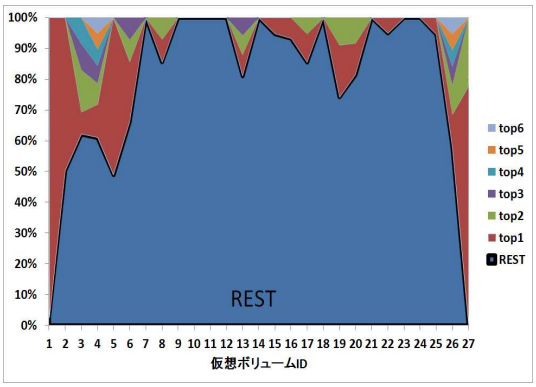

図 7　エリア発生 offset の割合 (60iopm)

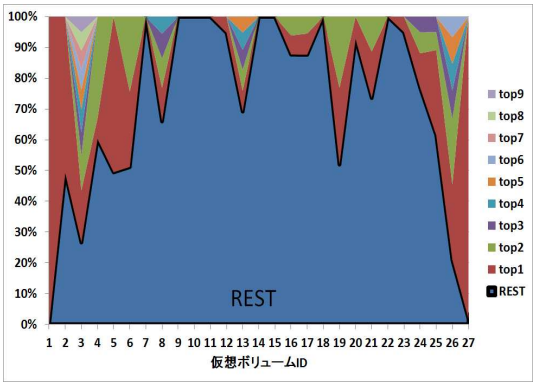

図 8　エリア発生 offset の割合 (600iopm)

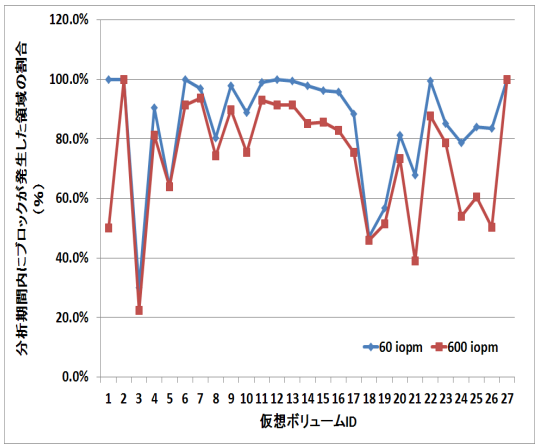

図 9　エリアが 1 回以上発生した領域の割合

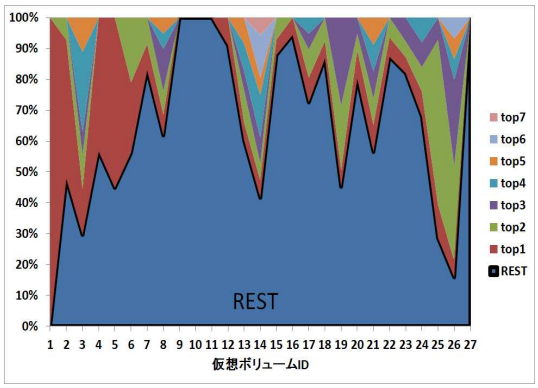

図 10　エリア発生 offset の割合 (long, 600iopm)

図 8 は 600iopm 以上のエリアに関する分析結果である．60iopm と比較すると幾つかの仮想ボリュームで REST の割合が減少していることが分かる．例えば ID=13 では，60iopm のとき REST=80%であったのが，600iopm だと REST=70%まで減少することが分かる．減少幅は仮想ボリュームごとにまちまちであるが，ほぼ全ての仮想ボリュームで減少していることが分かる．しかし，ID=1,3,26,27 を除くと REST が 50%以上となっており，少なくとも半数のエリアは任意の 1GB offset に発生していることが分かる．27 仮想ボリューム全てを合計すると 71%が REST に属していることになる．

図 9 は分析期間内に 1 回でもエリアが発生した領域の範囲を示したものである．仮想ボリューム ID=3 や 21 を除くと全領域の半分以上の領域でエリアが発生していることが分かる．27 仮想ボリュームの合計で 600iopm のケースで 74%，60iopm のケースで 85%範囲にエリアが発生することになる．一方，図 7，図 8 の分析結果より全エリアの少なくとも 71%は特定の 1GB offset に発生しないことが分かっており，図 9 と組み合わせると全容量の少なくとも 74%程度の offset の範囲に全エリアの少なくとも 71%が発生したことになる．

次に long/middle/short ごとの内訳分析を行う．図 7，図 8 より 600iopm と 60iopm でほぼ同じ傾向であったため，ここでは 600iopm のみの分析を行うことにした．図 10-

図 12 が分析結果である．仮想ボリューム ID=1,3,26 など一部の仮想ボリュームでは大部分のエリアが特定の 1GB offset に発生するが，これらを除くと少なくとも 50%は特定の 1GB offset 以外にエリアが発生していることが分かる．あと，一部の仮想ボリュームに関して，short，middle，long の順で特定の 1GB offset へエリアが発生する割合が高くなるものが存在することが分かる．例えば，ID=14 に注目すると，long では REST=40%であるが middle では REST=100%となり，継続時間が長くなると一部のエリアが特定 1GB offset に集まることが分かる．27 仮想ボリューム全てを合計すると，long は 66%，middle は 72%，short は 72%が REST に属する．

図 13，図 14 は分析期間内に 1 回でもエリアが発生した領域の範囲を示したものである．図 13 が 60iopm のケース，図 14 が 600iopm のケースである．まず図 13 を考察する．仮想ボリューム ID=3,21 等の一部の仮想ボリュームを除くと全領域の半分以上の領域でエリアが発生していることが分かる．27 仮想ボリューム全てを合計すると，全領域の long は 77%，middle は 79%，short は 79%にエリアが発生した．

次に図 14 を考察する．図より short,middle,long と継続時間が長くなるに従ってエリアが発生した領域の割合が狭

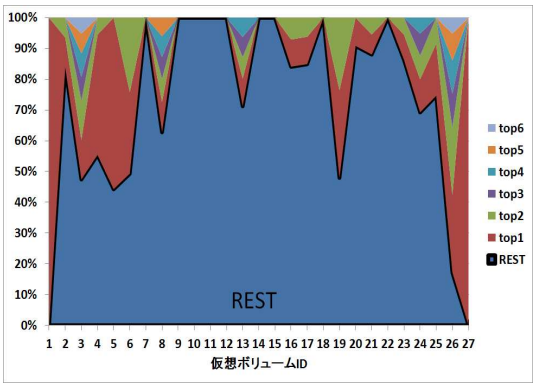

図 11　エリア発生 offset の割合 (middle, 600iopm)

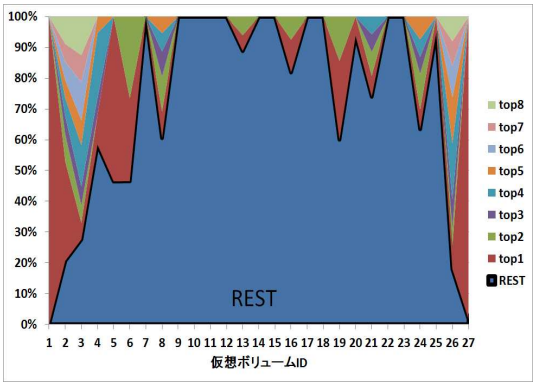

図 12　エリア発生 offset の割合 (short, 600iopm)

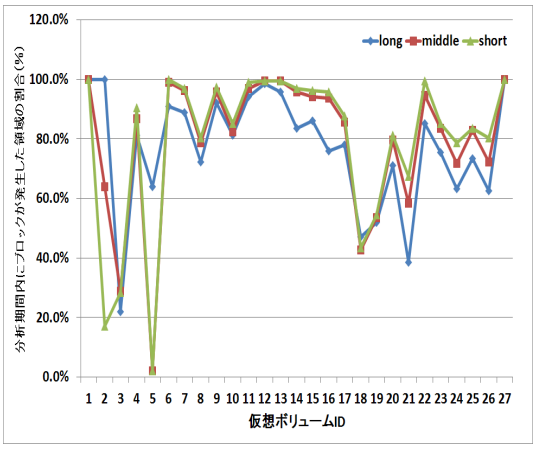

図 13　long/middle/short エリアが 1 回以上発生した領域の割合 (60iopm)

くなることが分かる．long に限ると全領域の最大 50% [*5]から 10%の範囲にしかエリアが発生しないことが分かる．27 仮想ボリューム全てを合計すると，全領域の long は 30%，middle は 49%，short は 66%にエリアが発生した．図 13 と比較すると，long に関してはエリアが発生する容量の範囲が狭くなったことが分かる．

5. 議論

表 6 は 4 章の分析結果をまとめたものである．図 15 は，

[*5] 仮想ボリューム ID=5,9,11

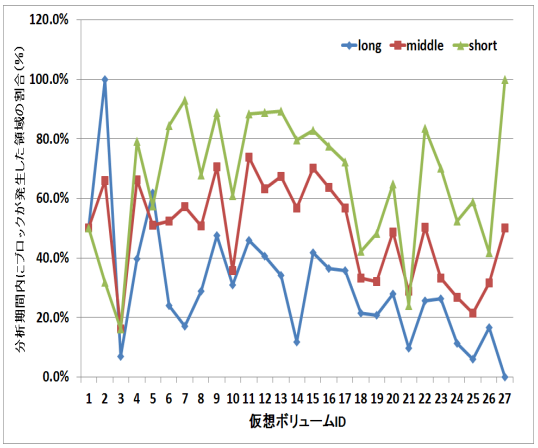

図 14　long/middle/short エリアが 1 回以上発生した領域の割合 (600iopm)

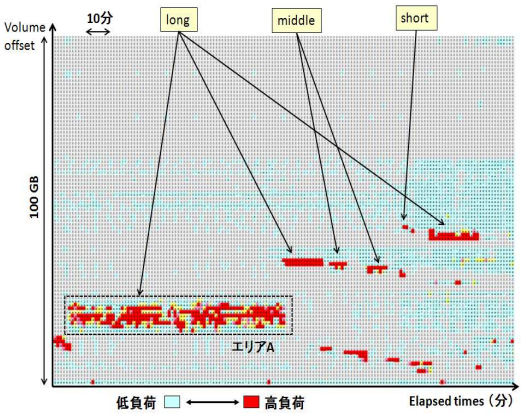

図 15　long/middle/short エリアの発生状況（実データより抜粋）

表 6　分析結果のまとめ

エリア	600iopm 以上	60iopm 以上
平均使用容量 (GB)	6	53
90%のエリアの使用容量 (GB)	12	85
最大使用容量 (GB)	44	192
全 IO に占める割合 (%)	58	81
write 比 (%)	88	77
long:middle:short(平均)	4:1:1	2:1:1
エリア発生範囲*(long)	30	77
エリア発生範囲*(middle)	49	79
エリア発生範囲*(short)	66	79
任意 offset へ発生割合 (%)**	71%	79%

*: 全容量 (4.4TB) に対する割合
**: エリア発生範囲内の任意の 1GB offset

実データの中からエリアが発生している箇所を抜粋したものである．本章では表 6 の様なワークロードを前提に，階層ストレージシステムの制御方法を議論する．議論の中で cache 方式と tiering 方式の比較を行うが，tiering は負荷の変動に応じてリアルタイムに構成変更する方式とし表 1 のパラメータを用いる．

文献 [4] 5.4 章に cache 方式 (Facebook Flashcache) と文献で提案が行われた tiering 方式の比較が行われている．

この中で write 比が高いワークロード (r:w=46:54) において，cache 方式は定常的に writeback 負荷が発生するために tiering 方式の方が性能向上する実験結果が示されてる．同時に tiering 方式では spike 領域の把握や tier 間移動に相応のコストが必要なことも示されている．さらに read 中心のワークロード (r:w=91:9) では cache 方式と tiering 方式に差がないことも示されている．そこで本稿の議論では，write 比が高く十分な継続時間を見込めるエリアに tiering 方式を使用し，残りのエリアは cache 方式を使用する方針で制御方法を議論する．

まず 600iopm 以上のエリアに関して議論する．表 6 より全 IO の 58%が平均 6GB の範囲に集まり，write 比が 88%に達することが分かる．90%のエリアが 12GB となり，最大使用容量は 44GB であることも分かる．表 1 の値を用いて 6GB の移動コストを見積もると，SSD から HDD へ書き戻す遅延まで含めて約 3 分[*6] となる．12GB では約 4 分である．よって，大部分は long と middle となるエリアに tiering を用い，short は cache を用いればよいことが分かる．最大使用容量を前提にすると，long エリアに限っても移動コストは 11 分に達してしまい，tiering を用いるのは long だけに絞る必要があることが分かる．12GB 超となるのは 600iopm 以上となるエリアの 10%以下であるが，エリアの容量方向の大きさに応じて middle まで tiering を用いるのかどうかを判断する必要があることが分かる．さらに，これらの制御を行うには，各エリアの継続時間の予測が必要となる．

次にエリアが発生する offset の観点で議論する．long と middle をあわせると表 6 より全体の 49%の範囲 (2156 GB) にエリアが発生し，その中の少なくとも 71%のエリアは 2156GB の範囲内の任意の 1GB offset に発生することになる．逆に一部のエリア (29%以下) は継続的に同じ offset に発生していることにもなる．継続的に同じ offset に発生するエリアに関しては，EMC FAST[2] などの様に日単位の統計情報を用いて 600iopm 以上のエリアが頻繁に発生する offset を抽出し，あらかじめ SSD へ移動しておく方法が考えられる．一方，任意の 1GB offset に発生するエリア (全エリアの 71%以上) に関しては，その都度エリアの発生を検出して制御しないと性能向上が見込めない．

次に 60iopm 以上 600iopm 未満となるエリアに関して議論する．表 6 より全 IO の 23%が平均 47GB の領域に集まり，write 比は約 37% [*7] となる．long だけでも 24GB に達し，tiering する場合の移動コストは 9 分となる．tiering を用いるのなら long エリアのみとなるが，write 比が 600iopm 以上との比較で小さくなることやここに属するエリアの IO 量が余り大きくないこともあり，long エリアの継続時間が十分に長くない限り全てのエリアは cache 方式でよいと判断する．

最後に図 15 に関して考察する．図のエリア A では空間軸方向・時間軸方向に比較的近い距離に IO が集中して発生することが分かる．これは図 3 で示したエリア抽出方法を用いることで空間軸方向・時間軸方向に近いセルを一体制御出来ることを示しており，近い将来負荷が発生する領域を prefetch したり，一旦負荷が収まってもすぐに負荷が復活する領域を SSD へとどめたままに出来ることを示している．

6. まとめ

商用環境で採取した Samba ワークロード 4.4TB 半年分を用いて空間的局所性とその継続時間の観点で分析し，空間軸方向では全体の 1%の領域 (約 44GB) に 81%の負荷が発生し，全体の 0.1%の領域 (約 6GB) に 58%の負荷が集まることを突き止めた．さらに，これら負荷の少なくとも 71%が任意の offset に数分から 10 分前後発生し，別の offset に移動することも分かった．write 比が 88%に達することも分かった．

この分析結果を用いた階層制御方法の検討を行い，10 分前後継続する負荷のみをリアルタイムに tiering し，残りの負荷は cache を用いる方法が有効である提案を行った．

7. 今後の予定

今後の課題は以下である．

- エリア継続時間求める方法
- cache 方式と tiering 方式を動的に切り替える方法

参考文献

[1] https://github.com/facebook/flashcache
[2] EMC White Paper, EMC FAST VP for Unified Storage Systems A Detailed Review, March 2011
[3] F.Chen, D.A. Koufaty, and X. Zhang, 'Hystor: Making the Best Use of Solid State Drivers in High Performance Storage Systems,' in ICS, 2011
[4] Kazuichi Oe, Kazutaka Ogihara, Yasuo Noguchi and Toshihiro Ozawa, Proposal for a hierarchical storage system that can move a spike to high-speed storage in real time, IPSJ Transactions on Advanced Computing Systems (No.40), Oct. 2012.
[5] Peter Bodik, Armando Fox, Michael J.Franklin, Michael I.Jordan, and David A.Patterson. Characterizing, Modeling, and Generating Workload Spikes for Stateful Services. ACM Symposium on Cloud Computing (SOCC 2010), June 2010.
[6] 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), October,2008
[7] http://www.tcpdump.org/

[*6] 60+6*10*2 (検知遅延+往復の移動時間)
[*7] 600iopm と 60iopm の差分計算で求めた